Mogens Blanke · Michel Kinnaert
Jan Lunze · Marcel Staroswiecki

Diagnosis and Fault-Tolerant Control

Third Edition

Springer

Diagnosis and Fault-Tolerant Control

Mogens Blanke · Michel Kinnaert
Jan Lunze · Marcel Staroswiecki

Diagnosis and Fault-Tolerant Control

Third Edition

With 218 Figures, 129 Examples, and 43 Exercises

Springer

Mogens Blanke
Department of Electrical Engineering,
Automation and Control Group
Technical University of Denmark
Kongens Lyngby
Denmark

Michel Kinnaert
Service d'Automatique et d'Analyse des
Systémes
Université Libre de Bruxelles
Brussels
Belgium

Jan Lunze
Ruhr-Universität Bochum
Bochum
Germany

Marcel Staroswiecki
Ecole Polytechnique Universitaire de Lille
Université Lille I
Villeneuve d'Ascq cedex
France

ISBN 978-3-662-47942-1 ISBN 978-3-662-47943-8 (eBook)
DOI 10.1007/978-3-662-47943-8

Library of Congress Control Number: 2015944440

Springer Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer-Verlag GmbH Berlin Heidelberg is part of Springer Science+Business Media
(www.springer.com)

Preface

Technological systems are vulnerable to faults. Actuator faults reduce the performance of control systems and may even cause a complete breakdown of the system. Erroneous sensor readings are the reason for operating points that are far from the optimal ones. Wear reduces the efficiency and quality of a production line. In many fault situations, the system operation has to be stopped to avoid damage to machinery and humans.

As a consequence, the detection and the handling of faults play an increasing role in modern technology, where many highly automated components interact in a complex way such that a fault in a single component may cause the malfunction of the whole system. Due to the simultaneously increasing economic demands and the numerous ecological and safety requirements to be met, high dependability of technological systems has become a dominant goal in industry.

This book introduces the main ideas of fault diagnosis and fault-tolerant control. It gives a thorough survey of new methods that have been developed in the recent years and demonstrates them with examples. To the knowledge of the authors, all major aspects of fault-tolerant control are treated for the first time in a single book from a common viewpoint.

Scope. Whereas fault diagnosis has been the subject of intensive research since the 1970s and there are several good books on this subject, systematic methods for fault handling is a new area of automatic control. The book considers both steps of fault-tolerant control together and shows how the information gained by model-based diagnosis can be used to find remedial actions that adapt the control algorithms to the faulty conditions in order to keep a system in operation. Basically, such actions can be classified as *fault accommodation*, which deals with the autonomous adaptation of the controller parameters to the faulty plant behaviour, and *control reconfiguration*, which includes the selection of a new control configuration and the online re-design of the controller.

The solution of these problems includes new analysis tasks like the test of the reconfigurability of the plant or the search for redundant sensors and actuators, which can replace faulty components. The aim is to close the control loop after a

breakdown of a component in the control loop has brought the controller out of operation. With respect to fault accommodation and control reconfiguration, the book presents the current state of the art.

The fault diagnostic parts of the book describe those methods and ideas which can be used to identify the fault with sufficient detail for fault accommodation or reconfiguration. The detection of a fault alone is not sufficient for fault-tolerant control, because the fault location and, possibly, the fault magnitude have to be known to activate appropriate remedial actions.

The design and implementation of fault-tolerant control necessitates a variety of techniques. The search for redundancies concerning the information and the possible control activities in a system, the selection of a reasonable control configuration, and the combination of diagnostic methods with controller design methods are some of the problems to be tackled. This set of different tasks cannot be dealt with by a single analytical model of the system under consideration, but different viewpoints have to be combined. For this reason, the book introduces a variety of models of dynamical systems and describes how these models can be used in fault-tolerant control. A component-oriented description of the system architecture is used to find the cause-effect chains from the primary faults towards the measured fault symptoms. A structural analysis based on bi-partite structure graphs is introduced to elaborate the analytical redundancies that can be used for fault diagnosis and fault-tolerant control actions. For the well-known continuous system representations like the state-space model and the transfer function, diagnostic methods and their extensions to fault-tolerant control are explained. With the presentation of diagnostic and reconfiguration methods for discrete-event systems, the book provides further novel material that has not yet been described in monographs or textbooks.

Structure of the book. This monograph consists of three parts:

- **Part I: Analysis based on components and system structure.** It is shown how abstract models of dynamical systems like component-oriented representations or structural graphs can be used to identify the connections between faults and symptoms and to find analytic redundancy relations for diagnosing faults.
- **Part II: Continuous systems.** Method for fault detection, fault identification and the re-design of the controller for a faulty system are described for continuous-variable systems that are represented by differential equations, difference equations or state-space models.
- **Part III: Discrete-event systems.** Methods for fault diagnosis and control reconfiguration are presented for discrete-event systems, whose behaviour is characterised by sequences of discrete signal changes and represented by deterministic, nondeterministic or stochastic automata.

As each of the models used requires its own mathematical background and the methods based on these models follow different lines of thinking, the book cannot present the methods in all details. The aim is to give the readers a broad view of the field and provide them with bibliographical notes for further reading. A further

reason for the different depth with which the chapters tackle the fault-tolerant control problems is given by the current status of research. Whereas for continuous-variable systems, fault diagnostic and fault-tolerant control methods have been developed for long, discrete-event systems became the subject of substantial research with respect to the topic of this book not before the 1990s. Hence, this field has not yet reached the same maturity as fault-tolerant control of continuous systems.

Many of the ideas are illustrated by **two running examples** that concern a simple tank system and a ship autopilot. The common use of these examples in several chapters makes a comparison of the alternative approaches very easy. It is the knowledge of the aims, models, ideas and methods used for different problems of fault diagnosis and fault-tolerant control that enables a control engineer to tackle practical problems under the circumstances given by the particular field of application. To introduce him to this knowledge is the primary aim of this book.

Level of the book. The intended readers of the book are graduate students of control, electrical, mechanical or process engineering with knowledge in dynamical systems, control design and filtering. The authors use the text in regular courses at the Université Libre de Bruxelles, the Ruhr-Universität Bochum, the Technical University of Denmark and the Norwegian University of Science and Technology.

In the introductory parts of all chapters the problems to be solved are posed in a framework that is familiar to practising engineers. They describe the new ideas and concepts of fault diagnosis and fault-tolerant control in an intuitive way, before these ideas are brought into a strict mathematical form. Examples illustrate the applicability of the methods. Bibliographical notes at the end of each chapter point to the origins of the presented ideas and the current research lines. The evaluation of the methods and the application studies should help the readers to assess the available methods and the limits of the present knowledge about fault-tolerant control with respect to their particular field of application.

The book is self-contained with a review of some basics in the appendices. Many figures illustrate the problems, methods and results in an intuitive way and make the interpretation of the rigorous mathematical treatment easier.

Common research. The large scope of the book was made possible by the close cooperation and by the common research of the four authors together with their Ph.D. students and colleagues. The introductory part (Chaps. 1 through 3) describe common ideas and results. The presentation of the methods for dealing with the system architecture (Chap. 4) is common work of the groups of Mogens Blanke in Aalborg and Lyngby (Denmark) and Marcel Staroswiecki in Lille (France). The part on structural analysis (Chap. 5) introduces the methods developed in Lille as they have been extended later on in Bochum and Lyngby. Diagnostic methods for continuous systems have been elaborated by many groups. The presentation of these ideas that can be used in fault-tolerant control (Chaps. 6 and 7) resulted from the common work and teaching experiences of Mogens Blanke, Michel Kinnaert (Brussels, Belgium) and Marcel Staroswiecki. Chapters 8–10 on fault

accommodation and control reconfiguration describe ideas of the four authors. The methods for dealing with discrete-event systems (Chaps. 11 and 12) have been elaborated by the group of Jan Lunze in Hamburg and Bochum (Germany).

Industrial applications. The methodologies presented in this book have been used in numerous industrial applications, among others in the automotive industry (fault-tolerant steering-by-wire, air system diagnosis), for aerospace (fault diagnosis of autonomous aircraft, fault-tolerant control of the Danish Ørsted satellite, detection of control surface vibrations, monitoring of the engine lubrication system), in the marine industry (fault-tolerant sensor fusion for navigation and for position mooring control), in offshore industry (prognosis and diagnosis of down-hole-drilling incidents), for wind turbines (pitch, load and yaw systems fault diagnosis, diagnosis of generator cooling), for electrical drives and in the process industry. The experiences gained by these applications are reflected in the selection and presentation of the material of this book.

Acknowledgements. The authors express their gratitude to the European Science Foundation and the European Union for financial support of the collaboration of the four groups in the COSY and the DAMADICS projects and to the national science funding organisations (Danish Research Council, Denmark; Région Wallonne, Belgium; Deutsche Forschungsgemeinschaft, Germany; Centre National de la Recherche Scientifique and Ministère de la Recherche, France; The Research Council of Norway) for supporting numerous projects in the field of fault diagnosis and fault-tolerant control.

Special thanks are due to our former and current PhD students and research associates, particularly to Roozbeh Izadi-Zamanabadi, Jakob Stoustrup and Jesper S. Thomsen (Aalborg), Henrik Niemann, Torsten Lorentzen, Ragnar I. Jónsson, Søren Hansen, Lidia Furno, Dimitrios Papageorgiou and Mikkel C. Nielsen (Lyngby), Manuel Gálvez Carrillo and Laurent Rakoto (Brussels), Frank Schiller and Jochen Schröder (Hamburg), Thomas Steffen (Hamburg/Bochum), Jörg Neidig, Jan Richter, Thorsten Schlage, Sven Bodenburg, Daniel Vey, Sebastian Pröll, Melanie Schuh and Markus Zgorzelski (Bochum), and Anne-Lise Géhin and Belkacem Ould Bouamama (Lille).

We are grateful for the valuable help of Ms. Andrea Marschall (Bochum) for drawing many of the figures and of Ms. Susanne Malow (Bochum) and Dr. Arben Cela (Paris) for technical assistance.

Third edition. After this book was used for a decade by several research groups, the third edition resulted from a major rewriting and restructuring of the material. In particular, Chap. 5 on structural analysis has been rewritten with more emphasis on the algorithms for finding analytical redundancy relations and the relation between structural and numerical properties of dynamical systems. Chapter 7 now includes more material on statistical change detection and isolation. In Chaps. 8 and 9, the reconfigurability analysis is presented separately from fault accommodation and reconfiguration methods and new methods have been inserted to extend this part towards the state of the art. Distributed diagnosis and distributed fault-tolerant

control have been included as a new topic for both continuous and discrete-event systems in Chaps. 10 and 12, respectively. Chapters 11 and 12 have been completely rewritten. The application chapter of the former editions has been moved to the book website.

Several new exercises should stimulate the readers to apply the methods presented to simple examples. The bibliographical notes have been updated and extended.[1]

Kongens Lyngby — Mogens Blanke
Brussels — Michel Kinnaert
Bochum — Jan Lunze
Paris — Marcel Staroswiecki
May 2015

[1]The book homepage at www.atp.rub.de/n/buch/ftcbook provides supplementary material for this book including lecture slides. A solutions manual can be made available for lecturers.

Contents

Part III Discrete-Event Systems

About the Authors

Mogens Blanke is professor in automation and control at the Technical University of Denmark and adjunct professor at Institute of Technical Cybernetics at the Norwegian University of Science and Technology. His research interests include autonomous and fault-tolerant systems, fault diagnosis and systems architecture design to obtain desired safety properties. Professor Blanke's experiences include the development of fault-tolerant control for the Danish Ørsted satellite, for marine automation and control, for mobile robots and for diagnosis for small aircraft. Professor Blanke is Technical Editor for Fault-Tolerant Systems for *IEEE Transactions on Aerospace and Electronic Systems* and he is Associate Editor for *Control Engineering Practice*.

Michel Kinnaert is professor in the Department of Control Engineering and System Analysis at the Université Libre de Bruxelles (Belgium). He has held a visiting professor position at the LAGEP at the Université Claude Bernard Lyon 1 and a postdoctoral position at the University of Newcastle (Australia). His research interests include fault diagnosis and fault-tolerant control for linear and nonlinear systems with applications in the process industry, in mechatronics and in wind farms. Professor Kinnaert has been the chairman of the IFAC Technical Committee SAFEPROCESS.

Jan Lunze is professor of automatic control and head of the Institute of Automation and Computer Control at the Ruhr-Universität Bochum (Germany). His research interests include fault diagnosis and reconfigurable control of discrete-event and hybrid systems, control theory with applications in the automotive and process industries, and networked control systems, where he has been the coordinator of a priority program of the German Research Foundation. He is author of numerous research papers and of monographs and textbooks on control theory, discrete-event systems and artificial intelligence with applications to dynamical systems.

Marcel Staroswiecki is Honorary Professor of automatic control at the Université des Sciences et Technologies de Lille (France). He is the former head of the

Laboratoire d'Automatique et d'Informatique Industrielle de Lille (LAIL-CNRS) and is currently with the Laboratoire SATIE-CNRS at Ecole Normale Supérieure de Cachan. Professor Staroswiecki and his group have been working on fault detection, isolation and recovery algorithms since 1986. His research addresses model, signal and data-based approaches to the supervision of complex and embedded systems with emphasis on structural analysis, intelligent instruments and components, and applications in the process industry and to transportation systems.

Chapter 1
Introduction to Diagnosis and Fault-Tolerant Control

Abstract This chapter introduces the aims, notions, concepts and ideas of fault diagnosis and fault-tolerant control and outlines the contents of the book.

1.1 Technological Processes Subject to Faults

Our modern society depends strongly upon the availability and correct function of complex technological processes as numerous examples show. Manufacturing systems consist of many different machine tools, robots and transportation systems all of which have to correctly satisfy their purpose in order to ensure an efficient and high-quality production. Economy and everyday life depend on the function of large power distribution networks and transportation systems, where faults in a single component have major effects on the availability and performance of the system as a whole. Mobile communication provides another example where networked components interact so heavily that component faults have far-reaching consequences. For automobiles strict legal regulations for protecting the environment claim that the engines have to be supervised and shut off in case of a fault.

In the general sense, a *fault* is something that changes the behaviour of a system such that the system no longer satisfies its purpose. It may be an internal event in the system, which stops the power supply, breaks an information link, or creates a leakage in a pipe. It may be a change in the environmental conditions that cause an ambient temperature increase and eventually stop a reaction or even destroy the reactor. It may be a wrong control action given by the human operator that brings the system out of the required operation point, or it may be an error in the design of the system, which remained undetected until the system moves into an operation point where this error reduces the performance considerably. In any case, the fault is the primary cause of changes in the system structure or parameters that eventually lead to a degraded system performance or even the loss of the system function.

In large systems, the overall system works satisfactorily only if all components provide the service they are designed for. Therefore, a fault in a single component usually changes the performance of the whole system.

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8_1

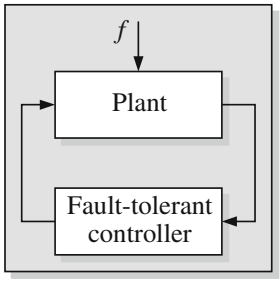


Fig. 1.1 Fault-tolerant system

In order to avoid production deteriorations or damage to machines and humans, faults have to be found as quickly as possible and decisions that stop the propagation of their effects have to be made. These measures should be carried out by the control equipment with the aim to make the system *fault tolerant*. If they are successful, the system function is satisfied also after the appearance of a fault, possibly after a short time of degraded performance in which the control algorithm adapts to the faulty plant.

From a systems-theoretic viewpoint, fault-tolerant control concerns the interaction between a given system (plant) and a controller (Fig. 1.1). The term "controller" is used here in a very general sense. It not only includes the usual feedback or feedforward control law, but also the decision-making layer that determines the control configuration. This layer analyses the behaviour of the plant in order to identify faults and changes the control law to hold the closed-loop system in a region of acceptable performance.

Controllers are usually designed for the faultless plant so that the closed loop meets the given performance specifications. Fault-tolerant control concerns the situation that the plant is subject to some fault f, which prevents the overall system from satisfying its goal in the future. A fault-tolerant controller has the ability to react to the existence of the fault by adjusting its activities to the faulty behaviour of the plant. Hence, for an observer who evaluates the function of the closed-loop system shown in Fig. 1.1, the system is fault-tolerant if it may be subject to some fault, but the fault is not "visible", because the system remains satisfying its designated goal.

Generally, the way to make a system fault-tolerant consists of two steps:

1. **Fault diagnosis**: The existence of faults has to be detected and the faults have to be identified.
2. **Control redesign**: The controller has to be adapted to the faulty situation so that the overall system continues to satisfy its goal.

These steps are not carried out by the usual feedback controller, but by a supervision system that prescribes the control structure and selects the algorithm and parameters of the feedback controller. As the supervision system reacts to the occurrence of a fault and changes the control loop, this two-step approach to fault-tolerant control is also referred to as *active fault-tolerant control*. As an alternative, it may be possible

for faults with small effects on the plant that the control loop tolerates the fault due to its robustness. Then, one speaks of *passive fault-tolerant control*.

Physical versus analytical redundancy. Engineers have been using this principle for a long time. Traditional methods for fault diagnosis include limit checking or spectral analysis of selected signals, which make the detection of specific faults possible. After a fault has been detected, the controller switches to a redundant component. For example, important elements of an aircraft use this principle with a threefold redundancy.

These traditional means for fault tolerance can only be applied to safety-critical systems. Indeed, for a more general use they are unnecessarily complicated and too expensive for two reasons. First, the traditional methods for fault diagnosis presuppose that for every fault to be detected there is a measurable signal that indicates the existence of the fault by, for example, the violation of a threshold or by changing its spectral properties. In complex systems with many possible faults, such a direct relation between a fault and an associated symptom does not exist or it is too expensive to measure all such signals. Second, this kind of fault tolerance is based on *physical redundancy*, where important components are implemented more than once. Industry cannot afford to use such kind of fault tolerance on a large scale.

The methods described in this book are based on *analytical redundancy*. An explicit mathematical model is used to perform the two steps of fault-tolerant control. The fault is diagnosed by using the information included in the model and in the online measurement signals. Then the model is adapted to the faulty situation and the controller is redesigned so that the closed-loop system including the faulty plant satisfies again the given specifications. Model-based fault-tolerant control is a cheaper way to enhance the dependability of systems than traditional methods based on physical redundancy.

The aim of the book is to describe the existing methods for model-based fault-tolerant control and to demonstrate their applicability by examples.

1.2 Faults and Fault Tolerance

1.2.1 Faults

A *fault* in a dynamical system is a deviation of the system structure or the system parameters from the nominal situation. Examples for structural changes are the blocking of an actuator, the loss of a sensor or the disconnection of a system component. In these situations, the set of interacting components of the plant or the interface between the plant and the controller are changed by the fault. Parametrical changes are brought about, for example, by wear or damage. All these faults yield deviations of the dynamical input/output (I/O) properties of the plant from the nominal ones and, hence, change the performance of the closed-loop system which further results in a degradation or even a loss of the system function.

System behaviour. For a more detailed analysis of the impact of faults consider the plant in Fig. 1.1 from the viewpoint of the controller. The fault is denoted by f. $\mathcal{F}$ is the set of all faults for which the function of the system should be retained. To simplify the presentation, the faultless case is also included in the fault set $\mathcal{F}$ and denoted by f_0. For the performance of the overall system it is important with which output $y(t)$ of the plant reacts if it gets the input $u(t)$. The pair (u, y) is called input/output pair (*I/O pair*) and the set of all possible pairs that may occur for a given plant define the *behaviour* $\mathcal{B}$. Note that for a single-input single-output system u and y denote the functions $u : |\mathcal{R} \to |\mathcal{R}$ and $y : |\mathcal{R} \to |\mathcal{R}$, which describe the input or output signals rather than the values of these functions for a specific time point.

Figure 1.2 gives a graphical interpretation. The behaviour $\mathcal{B}$ is a subset of the space $\mathcal{U} \times \mathcal{Y}$ of all possible combinations of input and output signals. The dot $A = (u_A, y_A)$ in the figure represents a specific I/O pair that may occur for the given system whereas $C = (u_C, y_C)$ represents a pair that is not consistent with the system dynamics. That is, for the input u_C the system produces an output $y \neq y_C$.

To illustrate the system behaviour in some more detail, consider a static linear system

$$y(t) = k_s\, u(t), \tag{1.1}$$

where k_s is the static gain. For static systems, the I/O pair can be considered for single time points t, for which the input and the output are elements of the set $|\mathcal{R}$ of real numbers. The set of all I/O pairs is given by

$$\mathcal{B} = \{(u,\ y)\ :\ y = k_s u\},$$

which can be graphically represented as a straight line in the u/y-coordinate system. Equation (1.1) describes, which values of u and y belong together. Faults are found, if this equation is violated, i.e. if the measured I/O pair $(u,\ y)$ does not belong to the behaviour $\mathcal{B}$ like the pair depicted by the point C in Fig. 1.2.

For a dynamical system the behaviour becomes more involved because the I/O pairs have to include the whole time functions $u(\cdot)$ and $y(\cdot)$ that represent the input and output signals. In a discrete-time setting, the input u is represented by the sequence

$$U = (u(0),\ u(1),\ u(2), \ldots,\ u(k_e))$$

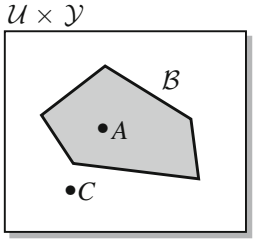


Fig. 1.2 Graphical illustration of the system behaviour

of input values that occur at the time instants $k = 0, 1,\ldots, k_e$, where k_e denotes the time horizon over which the sequence is considered. Often, k_e is the current time instant, until which the input sequence is stored. Likewise, the output is described by the sequence

$$Y = (y(0),\ y(1),\ y(2), \ldots,\ y(k_e)).$$

Consequently, the signal spaces $|\mathcal{R}$ used for the static system have to be replaced by $\mathcal{U} = |\mathcal{R}^{k_e}$ and $\mathcal{Y} = |\mathcal{R}^{k_e}$ for single-input single-output systems and by signal spaces of higher dimensions if the system has more than one input and one output. Then the behaviour $\mathcal{B}$ is a subset of the Cartesian product $\mathcal{U} \times \mathcal{Y} = |\mathcal{R}^{k_e} \times |\mathcal{R}^{k_e}$:

$$\mathcal{B} \subset |\mathcal{R}^{k_e} \times |\mathcal{R}^{k_e}$$

(Fig. 1.2). $\mathcal{B}$ includes all sequences U and Y that may occur for the faultless plant. For dynamical systems, the I/O pair is a pair $(U,\ Y)$ of sequences rather than a pair $(u,\ y)$ of current signal values.

Fault effects on the system behaviour. A fault changes the system behaviour as illustrated in Fig. 1.3. Instead of the white set, the system behaviour is moved by the fault towards the grey set. If a common input sequence U is applied to the faultless and the faulty system, then both systems answer with the output Y_A or Y_B, respectively. The points $A = (U,\ Y_A)$ and $B = (U,\ Y_B)$ differ and lie in the white or the grey set. This change in the system behaviour makes the detection and isolation of the fault possible, unless the faulty I/O pair lies in the intersection of $\mathcal{B}_0$ and $\mathcal{B}_f$.

In the strict sense, the fault is the primary cause of a malfunction. It has to be distinguished from the effects of the fault, which are described by the change of the I/O behaviour. Therefore, fault diagnosis has to trace back the cause–effect relations from the measured I/O pair, which is found to be different from the nominal one, to the primary cause of this change, which is the fault to be identified.

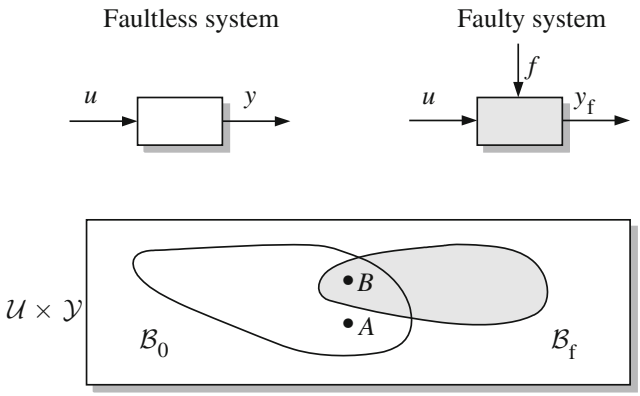


Fig. 1.3 System subject to faults

Modelling of faulty systems. For fault-tolerant control, dynamical models have to describe the plant subject to the faults $f \in \mathcal{F}$. These models will play a major role throughout this book. They describe the behaviour of the faultless and the faulty system, i.e. they restrict the possible I/O pairs to those that appear in the behaviour $\mathcal{B}_0$ or $\mathcal{B}_f$ in Fig. 1.3. Therefore, models represent *constraints* on the signals U and Y that appear at the plant. The notion of constraints will be used synonymously with the notion of model equations in this book.

In dependence upon the kind of systems considered, constraints can have the form of algebraic relations, differential or difference equations, automata tables or behavioural relations of automata. A set of such constraints constitutes a model, which can be used as a generator of the system behaviour. For a given input U the model yields the corresponding output Y. If the model is used for a specific fault, it shows how the system output Y is affected by this fault.

In fault diagnosis, the constraints are usually used to check the consistency of measured I/O pairs with the behaviour of the faultless or the faulty system. In this situation, not only the input U, but also the output Y is known and it is checked whether the pair (U, Y) belongs to the behaviour $\mathcal{B}_f$:

$$(U, Y) \stackrel{?}{\in} \mathcal{B}_f, \quad f \in \mathcal{F}.$$

Faults versus disturbances and model uncertainties. Like faults, disturbances and model uncertainties change the plant behaviour. In order to explain the distinction, consider a continuous-variable system that is described by an analytical model (e.g. differential equation). For this kind of systems, faults are usually represented as additional external signals or as parameter deviations. In the first case, the faults are called *additive faults*, because in the model the faults are represented by an unknown input that enters the model equation as addend. In the second case, the faults are called *multiplicative faults* because the system parameters depending on the fault size are multiplied with the input or system state.

In principle, disturbances and model uncertainties have similar effects on the system. Disturbances are usually represented by unknown input signals that have to be added up to the system output. Model uncertainties change the model parameters in a similar way as multiplicative faults. However, an important distinction between disturbances, model uncertainties and faults can be seen in the fact that disturbances and model uncertainties are always present, while faults may be present or not. Disturbances represent the action of the environment on the system, whereas uncertainties are a result of the modelling activities that end up with a model as an approximate representation of the system behaviour. Hence, both phenomena are nuisances whose effects on the system performance are handled by appropriate measures like filtering, feedback control or robust design. They do not call for fault-tolerant control, but for controllers designed so as to attenuate their effects.

On the other hand, improved maintenance and repair operations can remove existing faults or decrease the frequency of fault occurrence, but they will not

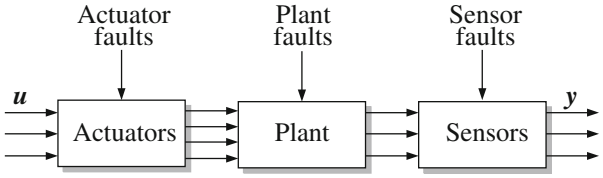


Fig. 1.4 Distinction between actuator faults, plant faults and sensor faults

suppress disturbances or model uncertainties. Since faults are changes whose effects on the plant behaviour cannot be suppressed by a fixed controller, fault-tolerant control must change the control law so as to cancel the effects of the faults or to attenuate them to an acceptable level.

Classification of faults. The faults are often classified as follows (Fig. 1.4):

- **Plant faults**: Such faults change the dynamical I/O properties of the system.
- **Sensor faults**: The plant properties are not affected, but the sensor readings have substantial errors.
- **Actuator faults**: The plant properties are not affected, but the influence of the controller on the plant is interrupted or modified.

Due to the "location" of sensor and actuator faults at the end or the beginning of the cause–effect-chain of the plant, there are specific methods for detecting and fighting against them. For example, several sections in Chaps. 8 and 9 deal with control reconfiguration for sensor or actuator failures, which open the control loop and can be overcome only by using an alternative sensor or actuator, respectively.

Faults can be distinguished concerning their size and temporal behaviour. Abrupt faults occur, for example, in a breakdown of the power supply whereas steadily increasing faults are brought about by wear, and intermittent faults by an intermitted electrical contact. All these different kinds of faults will be considered in this book, although not all methods are suitable to tackle all kinds of faults.

Fault versus failure. A short note is necessary concerning the distinction of the notions of fault and failure with respect to their current use in the engineering terminology. As explained above, a fault causes a change in the characteristics of a component such that the mode of operation or performance of the component is changed in an undesired way. Hence the required specifications on the system performance are no longer met. However, a fault can be "worked around" by fault-tolerant control so that the faulty system remains operational.

In contrast to this, the notion of a *failure* describes the inability of a system or component to accomplish its function. The system or a component has to be shut off, because the failure is an irrecoverable event. With these notions the idea of fault-tolerant control can be stated as follows:

Fault-tolerant control has to prevent a component fault from causing a failure at the system level.

Unfortunately, the notions of faults and failures are not clearly distinguished in the literature, but they are used precisely in the sense defined above all over this book.

1.2.2 Requirements and Properties of Systems Subject to Faults

As faults may cause substantial damage to the machinery, to the environment and risk for human life, engineers have investigated their appearance and impacts for decades. Different notions like safety, reliability, availability and dependability have been defined and investigated. In this section, the aims of fault-tolerant control are related to these notions, which result from different views on faulty systems.

- **Safety** describes the absence of danger. A safety system is a part of the control equipment that protects a technological system from permanent damage. It enables a controlled shutdown, which brings the technological process into a safe state. To do so, it evaluates the information about critical signals and activates dedicated actuators to stop the process if specified conditions are met. The overall system is then called a *fail-safe system*.
- **Reliability** is the probability that a system accomplishes its intended function for a specified period of time under normal conditions. Reliability studies evaluate the frequency with which the system is faulty, but they cannot say anything about the current fault status. Fault-tolerant control cannot change the reliability of the plant components, but it improves the reliability of the overall system, because with a fault-tolerant controller the overall system remains operational after the appearance of faults.
- **Availability** is the probability of a system to be operational when needed. Contrary to reliability it also depends on the maintenance policies, which are applied to the system components.
- **Dependability** lumps together the three properties of reliability, availability and safety. A dependable system is a fail-safe system with high availability and reliability.

As explained earlier, a fault-tolerant system has the property that faults do not develop into a failure of the closed-loop system. In the strict form, the performance remains the same. Then the system is said to be *fail-operational*. In a reduced form, the system remains in operation after faults have occurred, but the system has degraded performance. Then it is called to be *fail-graceful*.

Safety versus fault tolerance. Due to its importance, the relation between safety and fault tolerance is elaborated now in more detail. Assume that the system performance can be described by the two variables y_1 and y_2. Then Fig. 1.5 shows the different regions that have to be considered.

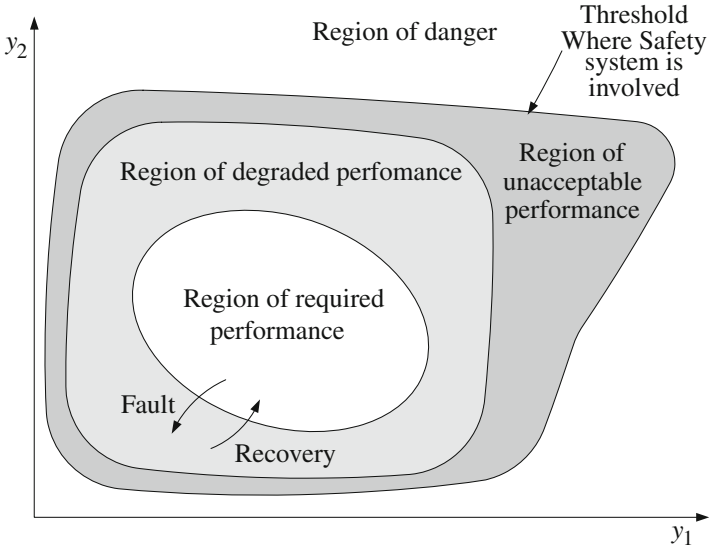


Fig. 1.5 Regions of required and degraded performance

In the region of required performance, the system satisfies its function. This is the region where the system should remain during its time of operation. The controller makes the nominal system remain in this region in spite of disturbances and uncertainties of the model used for the controller design. The controller may even hold the system in this region if small faults occur, although this is not its primary goal. In this case, the controller "hides" the effect of faults, which is not its intended purpose and makes the fault diagnostic task more difficult.

The region of degraded performance shows where the faulty system is allowed to remain, although in this region the performance does not satisfy the given requirements but may be considerably degraded. Faults bring the system from the region of the required performance into the region of degraded performance. The fault-tolerant controller should be able to initiate recovery actions that prevent a further degradation of the performance towards the unacceptable or dangerous regions and it should move the system back into the region of required performance. At the border between the two regions, the supervision system is invoked, which diagnoses the faults and adjusts the controller to the new situation.

The region of unacceptable performance should be avoided by means of fault-tolerant control. This region lies between the region of acceptable performance in which the system should remain and the region of danger, which the system should never reach.

A safety system interrupts the operation of the overall system to avoid danger for the system and its environment. It is invoked if the outer border of the region of unacceptable performance is exceeded. This shows that the safety system and the fault-tolerant controller work in separate regions of the signal space and satisfy

complementary aims. In many applications, they represent two separate parts of the control system. For example, in the process industry, safety systems and supervision systems are implemented in separate units. This separation makes it possible to design fault-tolerant controllers without the need to meet safety standards.

1.3 Elements of Fault-Tolerant Control

1.3.1 Structure of Fault-Tolerant Control Systems

The architecture of fault-tolerant control is depicted in Fig. 1.6. The two blocks "diagnosis" and "controller redesign" carry out the two steps of active fault-tolerant control introduced on p. 2.

1. The diagnostic block uses the measured input and output signals and tests their consistency with the plant model. Its result is a characterisation of the fault f with sufficient accuracy for the controller redesign.

2. The redesign block uses the fault information and adjusts the controller to the faulty situation.

To be successful in Step 2, there must exist a solution to the controller redesign problem in the faulty situation. If such a solution exists, the fault is said to be *recoverable*, otherwise it is non-recoverable. With respect to Fig. 1.5, a recoverable fault allows the controller to bring the system back into the region of required (or degraded) performance. For a non-recoverable fault, there does not exist any controller that is able to prevent the system from drifting into the region of unacceptable performance or even into the region of danger. Then, the supervision level has to make a decision about the system objectives (e.g. safe shutdown), since the current objectives can no longer be achieved. To decide which situation occurs with respect to the present fault is the aim of the recoverability test in Fig. 1.6.

Since the notion of the controller is used here in a very broad sense, the input u to the plant includes all signals that can be influenced by the control decision units. The aims and methods associated with both blocks will be discussed in more detail below.

In Fig. 1.6 all simple arrows represent signals. The connection between the controller redesign block and the controller is drawn by a double arrow in order to indicate that this connection represents an information link in a more general sense. The redesign of the controller may not only result in new controller parameters, but also in a new control configuration. Then the old and the new controllers differ with respect to the input and output signals that they use (Sect. 1.3.3).

The figure shows that fault-tolerant control extends the usual feedback controller by a supervisor, which includes the diagnosis and the controller redesign blocks. In the faultless case, the nominal controller attenuates the disturbance d and ensures set

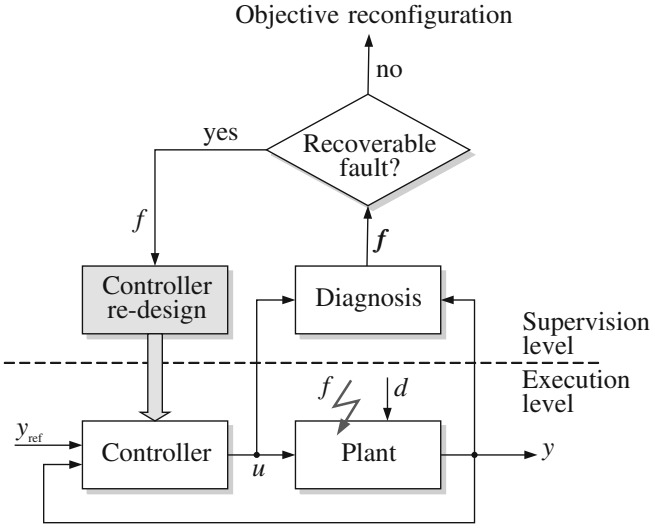


Fig. 1.6 Architecture of fault-tolerant control

point following and other requirements on the closed-loop system. The main control activities occur on the execution level. On the supervision level the diagnostic block simply recognises that the closed-loop system is faultless and no change of the control law is necessary.

If a fault f occurs, the supervision level makes the control loop fault-tolerant. The diagnostic block identifies the fault and the controller redesign block adjusts the control law to the new situation. Afterwards, the execution level alone continues to satisfy the control aims.

In Fig. 1.6 as well as in the next figures the diagnostic result f is assumed to be identical to the fault f occurring in the system. This reflects an idealised situation, because in many applications disturbances or model uncertainties bring about uncertainties of the diagnostic results so that instead of the fault f only an approximate fault $\hat{f}$ or a set $\mathcal{F}_c$ of fault candidates is obtained. This fact will be investigated in detail in all chapters of this book. Here, however, the idealised situation is considered in order to explain the basic ideas of fault diagnosis and fault-tolerant control.

Established methods for ensuring fault tolerance. To a certain extent, fault tolerance can also be accomplished without the structure given in Fig. 1.6 by means of well-established control methods. As this is possible only for a restricted class of faults, these methods will not be dealt with in more detail in this book, but they should be mentioned here.

- **Robust control**: A fixed controller is designed that tolerates changes of the plant dynamics. The controlled system satisfies its goals under all faulty conditions. Fault tolerance is obtained without changing the controller parameters. It is, therefore,

a method providing passive fault tolerance. However, the theory of robust control has shown that robust controllers exist only for a restricted class of changes of the plant behaviour that may be caused by faults. Further, a robust controller is not the best controller for the nominal plant because its parameters are fixed so as to get a trade-off between performance and robustness.

- **Adaptive control**: The controller parameters are adapted to changes of the plant parameters. If these changes are caused by some fault, adaptive control may provide active fault tolerance. However, the theory of adaptive control shows that this principle is particularly efficient only for plants that are described by linear models with slowly varying parameters. These restrictions are usually not met by systems under the influence of faults, which typically have a nonlinear behaviour with sudden and large parameter changes.

From a structural point of view, adaptive control has a similar structure as fault-tolerant control, if the diagnostic block is replaced by a block that identifies the current plant parameters and the controller redesign block adapts the controller parameters to the identification result (Fig. 1.6). However, in fault-tolerant control the size of the changes of the plant behaviour is larger and not restricted to parameter changes and to continuous-variable systems.

If the modifications of the plant dynamics brought about by faults satisfy the requirements that are necessary to apply robust or adaptive control schemes, then these schemes provide reasonable solutions to the fault-tolerant control problem. However, for severe or sudden faults, these methods are not applicable and the ideas presented in this book have to be used.

Fault-tolerant control at the component level and the overall system level. Modern technological systems consist of several, often many subsystems, which are strongly connected. The effect of a fault in a single component propagates through the overall system. In Fig. 1.7 the fault occurring in Component 2 influences all other components.

The effect of a fault in a single component may be of minor importance to this component. However, due to its propagation throughout the overall system, the fault

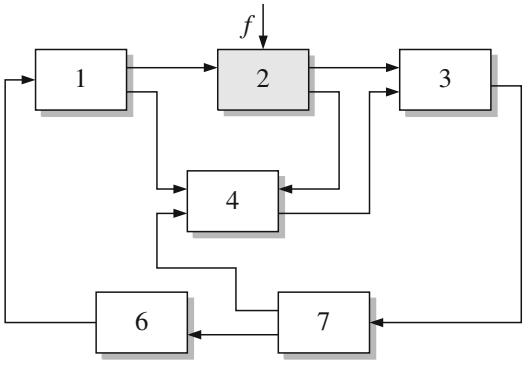


Fig. 1.7 Fault propagation in interconnected systems

may eventually initiate the safety system to shut off the whole system. In the terms defined above, the fault has then caused a system failure.

There are two possibilities to stop the propagation of the fault. Either the fault propagation is stopped inside the affected component by making the component fault-tolerant or the propagation of the fault among the components has to be stopped. As the propagation of the fault effects through the overall system usually takes time, the controller of the affected component has the chance to adjust its behaviour to the faulty situation and, hence, to keep the overall system in operation.

1.3.2 Main Ideas of Fault Diagnosis

The first task of fault-tolerant control concerns the detection and identification of existing faults. Figure 1.8 illustrates the diagnostic problem. A dynamical system with input u and output y is subjected to some fault f. The system behaviour depends on the fault $f \in \mathcal{F}$ where the element $f_0 \in \mathcal{F}$ symbolises the faultless case. The diagnostic system obtains the I/O pair $(U,\ Y)$, which consists of the sequences

$$\begin{aligned} U &= (u(0),\ u(1),\ u(2), \ldots,\ u(k_\mathrm{e})) \\ Y &= (y(0),\ y(1),\ y(2), \ldots,\ y(k_\mathrm{e})) \end{aligned}$$

of input and output values measured at discrete-time points $k = 0, 1, \ldots, k_\mathrm{e}$ within a given time horizon k_e. It has to solve the following problem:

Diagnostic Problem. *For a given I/O pair* $(U,\ Y)$, *find the fault* f.

If the unique result is f_0, the diagnostic system indicates that the system is faultless or that a non-detectable fault has occurred as explained below.

It should be emphasised that the problem considered here concerns online diagnosis based on the available measurement data. No inspection of the process is possible. The diagnostic problem has to be solved under real-time constraints by exploitation

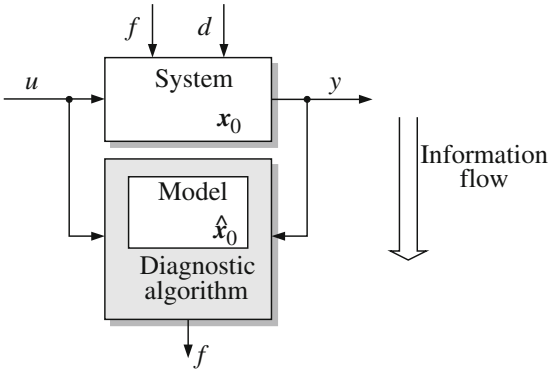


Fig. 1.8 Fault diagnosis

of the information included in a dynamical model and in the time evolution of the signals u and y. Therefore, the term *process diagnosis* is used if these aspects should be emphasised.

Diagnostic steps. For fault-tolerant control, the location and the magnitude of the fault have to be found. Different names are used to distinguish the diagnostic steps according to their "depth":

- **Fault detection**: Decide whether or not a fault has occurred. This step determines the time at which the system is subject to some fault.
- **Fault isolation**: Find in which component a fault has occurred. This step determines the location of the fault.
- **Fault identification and fault estimation**: Identify the fault and estimate its magnitude. This step determines the kind of fault and its severity.

Consistency-based diagnosis. Different diagnostic methods are explained throughout this book. Although they use different kinds of dynamical models and have different assumptions concerning the measurement information available, they follow a common principle, which can be explained by using the notion of the system behaviour.

In order to be able to detect a fault, the measurement information $(U,\ Y)$ alone is not sufficient, but a reference, which describes the nominal plant behaviour, is necessary. This reference is given by a plant model, which describes the relation between the possible input sequences and output signals. This model is a representation of the plant behaviour $\mathcal{B}$.

The idea of consistency-based diagnosis should be explained now by means of Fig. 1.2 on p. 4. Assume that the current I/O pair $(U,\ Y)$ is represented by point A in the figure. If the system is faultless (and the model is correct) then A lies in the set $\mathcal{B}$. However, if the system is faulty, it generates a different output $\hat{Y}$ for the given input U. If the new I/O pair $(U,\ \hat{Y})$ is represented by point C, which is outside of $\mathcal{B}$ then the fault is detectable. However, if the faulty system produces the I/O pair represented by point B in Fig. 1.3, no inconsistency occurs in spite of the fault. Hence, the fault is not detectable.

The principle of *consistency-based diagnosis* is to test whether or not the measurement $(U,\ Y)$ is consistent with the system behaviour. If the I/O pair is checked with respect to the nominal system behaviour, a fault is detected if $(U,\ Y) \notin \mathcal{B}$ holds. If the I/O pair is consistent with the behaviour $\mathcal{B}_f$ of the system subject to the fault f, the fault f may be present in the system. In this case, f is called a *fault candidate*. The diagnostic result is usually a set $\mathcal{F}_c \subseteq \mathcal{F}$ of fault candidates.

To illustrate this result, assume that the system behaviour is known for the faults f_0, f_1 and f_2. The corresponding behaviours $\mathcal{B}_0$, $\mathcal{B}_1$ and $\mathcal{B}_2$ are different, but they usually overlap, and there exist I/O pairs that may occur for more than one fault. If the I/O pair is represented by the points A, C or D in Fig. 1.9, the faults found are f_0, f_1 or f_2, respectively. If, however, the measurement sequences are represented by point B, the system may be subjected to one of the faults f_0 or f_1. The diagnostic

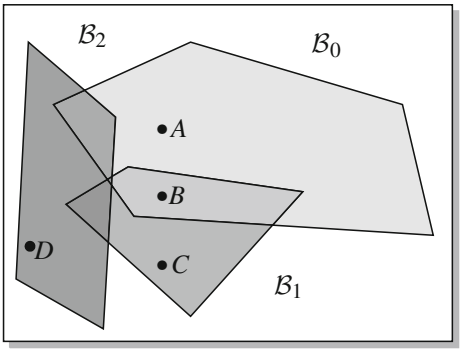


Fig. 1.9 Behaviour of the faultless and the faulty system

algorithm cannot distinguish between these faults because the measured I/O pair may occur for both faults. Hence, the ambiguity of the diagnostic result is caused by the system and not by the diagnoser, because the system generates the same information for both faults. No diagnostic method can remove this ambiguity by means of the given measurement information $(U,\ Y)$. This results in the set $\mathcal{F}_c = \{f_0,\ f_1\}$ of fault candidates.

The question of whether or not a certain fault can be detected concerns the *diagnosability* or *fault detectability* of the system, which are important system properties to be considered in several chapters of this book.

In summary, the diagnostic principle can be described as follows:

- **Consistency-based diagnosis**: For given models that describe the behaviour $\mathcal{B}_f$ of the system subject to the faults $f \in \mathcal{F}$, test whether the I/O pair $(U,\ Y)$ satisfies the relation
$$(U,\ Y) \in \mathcal{B}_f.$$
- **Fault detection**: If the I/O pair is inconsistent with the behaviour $\mathcal{B}_0$ of the faultless system
$$(U,\ Y) \notin \mathcal{B}_0$$
then a fault is known to have occurred.
- **Fault isolation and identification**: If the I/O pair is consistent with the behaviour $\mathcal{B}_f$
$$(U,\ Y) \in \mathcal{B}_f,$$
then the fault f may have occurred. f is a fault candidate.

To diagnose a system by testing the consistency of the measurements with a model is a general idea, which does not depend on the kind of model used.

Several direct consequences of this principle should be mentioned:

- Fault detection is possible without any information about the behaviour of the faulty plant. Fault detection algorithms use only a model of the nominal plant. The main idea is to identify deviations of the current system behaviour from the nominal behaviour, which is possible without a list of all possible faults and the corresponding plant models.
- Without information about the faults and about the way in which the faults affect the system, no fault isolation and identification is possible. In order to identify the fault, fault models have to be known.
- Consistency-based diagnosis *excludes* faults $f \in \mathcal{F}$ as fault candidates. There is no possibility to *prove* that a certain fault is present. This would necessitate further assumptions like the assumption that the present fault f is an element of a given fault set $\mathcal{F}$. For example, such an assumption holds true if the faults can be restricted to be a sensor fault.
- With a given measurement configuration, not all faults can be distinguished. Diagnosability considerations can be used to determine those faults that can be separately identified.

Consistency-based diagnosis concerns the comparison of the measured I/O pair with a plant model. For discrete-event systems this comparison is done in a direct way as described in Chaps. 11 and 12. For continuous-variable systems the usual way of comparison consists in using the difference between the measured system output and the model output in the way explained below.

Diagnosis of continuous-variable systems. Continuous-variable systems, which will be investigated in Chaps. 6 and 7, are usually described by differential equations or transfer functions. With these models, the principle of consistency-based diagnosis can be transformed into the scheme shown in Fig. 1.10. The model is used to determine, for the measured input sequence U, the model output sequence $\hat{Y}$. The consistency of the system with the model can be checked at every time t by determining the difference

$$r(t) = y(t) - \hat{y}(t),$$

which is called a *residual*. In the faultless case, the residual vanishes or is close to zero. A non-vanishing residual indicates the existence of a fault.

Diagnostic algorithms for continuous-variable systems generally consist of two components:

1. **Residual generation**: The model and the I/O pair are used to determine residuals, which describe the degree of consistency between the plant and the model behaviour.
2. **Residual evaluation**: The residuals are evaluated in order to detect, isolate and identify faults.

In both steps, model uncertainties, disturbances and measurement noise have to be taken into account.

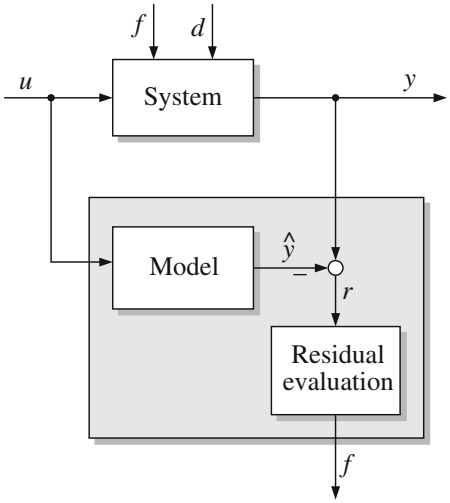


Fig. 1.10 Diagnosis of continuous-variable systems

Figure 1.10 shows the fact mentioned earlier that fault-tolerant control employs *analytical redundancy*. The model is an integral part of the diagnostic system. The residual is found by using more than one way for determining the variable y. The sensor value y is compared with the analytically computed value $\hat{y}$ of the same signal. This procedure avoids physical redundancy where at least three sensors are used to measure the same quantity in order to get fault indicators.

General properties of diagnostic algorithms. Some further general remarks should be made concerning practical problems encountered in process diagnosis. First, the behaviour of a dynamical system does not only depend on the input but also on the initial state. In Fig. 1.8 the initial state of the plant is denoted by $\boldsymbol{x}_0$ and that of the model by $\hat{\boldsymbol{x}}_0$. Inconsistencies may result from a deviation of both initial states. As the initial state of the system is usually immeasurable, every diagnostic problem includes a kind of state observation problem.

Second, the disturbance d that influences the plant is usually immeasurable. As it influences the plant behaviour, it has to be taken into account in the consistency check. For continuous-variable systems, this problem may be solved for certain classes of disturbances by including filters into the residual evaluation block.

Fault diagnosis for fault-tolerant control. In fault-tolerant control, the information obtained from the diagnostic algorithm should be used in the controller redesign. Hence, process diagnosis should not only indicate that some faults have occurred but it has to identify the fault locations and fault magnitudes with sufficient precision. This information will make it possible to set up a model of the faulty system, which can be used for the controller redesign.

Fault isolation and fault identification are essential for active fault-tolerant control. This contrasts with safety systems for which the information about the existence of some (unspecified) fault is sufficient. This fact shows another difference of the measures to be taken for fault tolerance or for safety, respectively.

1.3.3 Main Ideas of Controller Redesign

Controller redesign considers the problem of changing the control structure and the control law after a fault has occurred in the plant. The aim is to satisfy the requirements on the closed-loop system in spite of the faulty behaviour of the plant.

The necessity and aim of the controller redesign can be illustrated without reference to a particular class of systems by using again the notion of the system behaviour (Fig. 1.11). The faultless plant has the behaviour $\mathcal{B}_0$ and the controller has the behaviour $\mathcal{B}_C$. The set $\mathcal{B}_C$ describes the I/O pairs $(U,\ Y)$ that satisfy the control law. Since the I/O pairs of the closed-loop system are consistent with both the plant and the controller, the behaviour of the closed-loop system is given by the intersection $\mathcal{B}_0 \cap \mathcal{B}_C$, which is drawn in grey on the left-hand side of the figure. This behaviour satisfies the control specifications, which likewise can be formulated in the behavioural setting as the set $\mathcal{B}_{spec}$ of those I/O pairs that meet these requirements. Its border is drawn by the thick rectangle in the figure. As the grey set lies completely within the set $\mathcal{B}_{spec}$

$$\mathcal{B}_0 \cap \mathcal{B}_C \subset \mathcal{B}_{spec}$$

the closed-loop systems satisfies the performance specifications.

If the plant becomes faulty, it changes its behaviour, which is now given by the set $\mathcal{B}_f$. Hence, the closed-loop system behaviour changes to become $\mathcal{B}_f \cap \mathcal{B}_C$, which may no longer be a subset of $\mathcal{B}_{spec}$. On the right-hand side of the figure, this situation occurs because the grey set only partly overlaps with the set $\mathcal{B}_{spec}$. Hence, the controller has to be redesigned in order to restrict the behaviour of the faulty system to the set $\mathcal{B}_{spec}$. This explains the necessity of the controller redesign from the behavioural viewpoint.

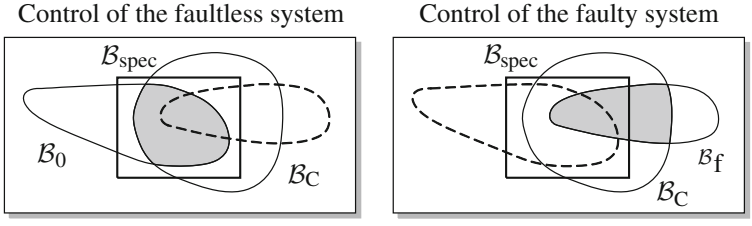


Fig. 1.11 Behaviour of the faultless and the faulty closed-loop system

The figure also shows that fault tolerance may or may not be possible depending on the properties of the faulty system. If the behaviour $\mathcal{B}_f$ overlaps with the specified behaviour $\mathcal{B}_{spec}$, a controller may be found that restricts this set to a new set $\mathcal{B}_f \cap \mathcal{B}_C$ which satisfies the relation

$$\mathcal{B}_f \cap \mathcal{B}_C \subset \mathcal{B}_{spec}$$

(Fig. 1.12). This controller makes it possible to hold the faulty system in operation. When adapting the controller parameter to the faulty plant, the set $\mathcal{B}_C$ cannot be chosen arbitrarily because restrictions concerning the realisability of the control law have to be satisfied. These restrictions bring about further difficulties into the fault-tolerant control problem, which will be discussed later.

There may be faults, for which the behaviour $\mathcal{B}_f$ does not overlap with $\mathcal{B}_{spec}$. Then a new control configuration has to be chosen, which changes the signals under consideration and, hence, the behaviour of the plant severely. There may even be faults for which no controller can make the closed-loop system satisfy the specification and the system has to be shut off. Hence, the question whether a fault-tolerant controller exists is not a property of the controller or the control redesign method, but a property of the plant subject to faults. Faults for which redesigned controllers exist, are called *recoverable*, otherwise unrecoverable. An illustrative example for an unsolvable fault-tolerant control problem is to consider a plant whose unstable modes become uncontrollable or unobservable due to faults. Then no controller exists which stabilises the faulty plant and a new system mode of operation, for example, a safe shut-off operation, has to be invoked (Fig. 1.6).

Two principal ways of controller redesign have to be distinguished, which are described in more detail now: fault accommodation and control reconfiguration.

Fault accommodation. Fault accommodation means to adapt the controller parameters to the dynamical properties of the faulty plant. The input and output of the plant used in the control loop remain the same as for the faultless case (Fig. 1.13). Hence, the set $\mathcal{U} \times \mathcal{Y}$ of input and output signals is not changed and fault accommodation is the situation illustrated by Fig. 1.12.

A simple but well-established way of fault accommodation is based on predesigned controllers, each of which has been selected offline for a specific fault. The redesign step then simply sets the switch among the different control laws. This step is quick and can meet strong real-time constraints. However, the controller redesign has to be made for all possible faults before the system is put into operation and all resulting controllers have to be stored in the control software.

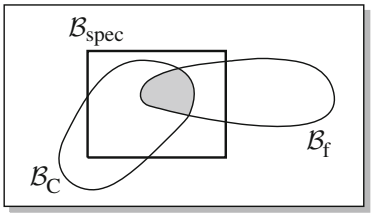


Fig. 1.12 Behavioural representation of fault accommodation

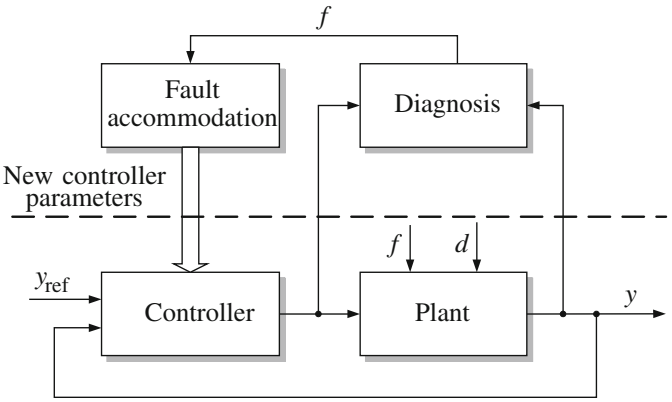


Fig. 1.13 Fault accommodation

More general ways of fault accommodation will be explained in Chap. 9. This treatment also includes the development of conditions under which fault accommodation is possible (recoverability analysis), which means that the plant is recoverable from the fault.

Control reconfiguration. If fault accommodation is impossible, the complete control loop has to be reconfigured. Reconfiguration includes the selection of a new control configuration where alternative input and output signals are used. The selection of these signals depends upon the existing faults. Then, a new control law has to be designed online (Fig. 1.14).

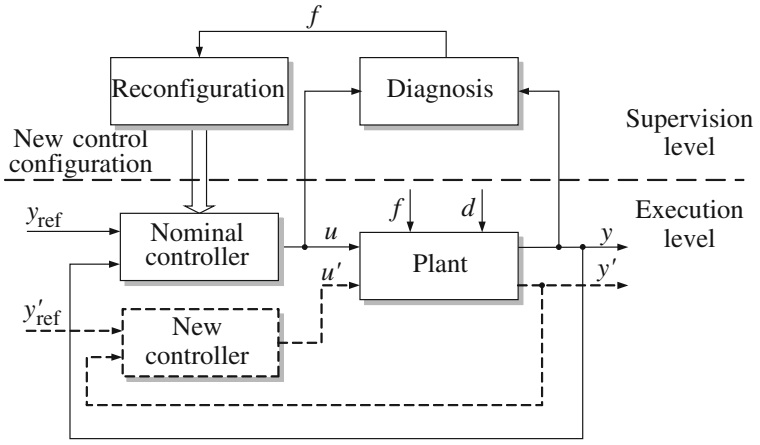


Fig. 1.14 Control reconfiguration

Control reconfiguration is necessary after severe faults have occurred that lead to serious structural changes of the plant dynamics:

- **Sensor failures** break the information link between the plant and the controller. They may make the plant partially unobservable. New measurements have to be selected and used in order to solve the control task.
- **Actuator failures** disturb the possibilities to influence the plant. They may make the plant partially uncontrollable. Other actuators have to be used.
- **Plant faults** change the dynamical behaviour of the process. If these changes cannot be tolerated by any control law, the overall control loop has to be reconfigured.

The necessity of control reconfiguration is particularly obvious if sensor or actuator failures are considered. If these components stop working completely, the fault leads to a breakdown of the control loop. There is no possibility to adapt the controller by simply changing its parameters to the faulty situation. Instead, other actuators or sensors have to be found, which are not affected by the fault and which have similar interactions with the plant so that a reasonably selected controller is able to satisfy the performance specifications on the closed-loop system. General method for the selection of a new control configuration and the redesign of the controller for the new configuration after sensor or actuator failures are given in Chaps. 8 and 9. Again, the possibility of finding a new controller that satisfies the control aims for the faulty system is a property of the plant, which is called here *reconfigurability*. Conditions for the reconfigurability are given in terms of the plant model.

Real-time aspects of fault accommodation and control reconfiguration. Both fault accommodation and control reconfiguration imply the online redesign of the controller, which is reminiscent of the "usual" controller design. However, although they may use well-known design methods, they also pose new problems that did not appear in the usual controller design problem, since they have to be carried out under additional restrictions and new circumstances:

- The design process has to be completely automatic, i.e. without interaction with a human designer.
- The methods used for fault accommodation and control reconfiguration have to guarantee a solution to the design problem (if the fault is recoverable) even if the performance is not optimal.
- Fault accommodation and control reconfiguration have to be done under real-time constraints.
- With the controller of the nominal system, a solution to the controller design problem is known, which may be used for control reconfiguration.

The real-time constraints can be seen from a detailed analysis of the time sequence that takes place between the occurrence of a fault and its recovery (i.e. the time when the accommodated or reconfigured control that satisfies the control objectives is applied). The following time windows can be distinguished:

- Before the fault occurrence at time t_f, the nominal system is controlled using the nominal control and the control objectives are satisfied.
- Between fault occurrence and fault recovery at time t_r, the faulty system is controlled using the nominal control law, and control objectives are in general not satisfied. The system may even become unstable.
- After the fault recovery time $t > t_r$, the faulty system is controlled using the accommodated or reconfigured control and the system objectives are satisfied again.

The second point above is critical, and the associated time window $t \in [t_f, t_r]$ should be made as short as possible. Note that this time window occurs due to three reasons:

- Fault detection and isolation delay
- Fault estimation delay
- Delay for the redesign of the accommodated or reconfigured control.

The fault detection and isolation delay is unavoidable in active fault-tolerant control. The fault estimation delay is unavoidable if online fault accommodation is used, since the model of the faulty system must be identified. The delay for the redesign is very short, if fault accommodation is implemented as switching between pre-designed controllers.

Fault accommodation and control reconfiguration is a recently started subject of research. There are several promising solutions, which are summarised in this book. In particular, Chap. 4 describes methods for fault propagation analysis, which can be used to find out where the fault propagation can be stopped. The structural analysis explained in Chap. 5 shows the redundancies that can be used for diagnosis and reconfiguration. Specific methods for continuous-variable plants are explained in Chaps. 8 and 9.

1.3.4 A Practical View on Fault-Tolerant Control

This section takes a view on fault-tolerant control from a practical perspective and emphasises the possible fields of application.

Physical redundancy versus analytical redundancy. The main advantage of fault-tolerant control over other measures for fault tolerance is the fact that fault-tolerant control makes "intelligent" use of the redundancies included in the system and in the information about the system in order to increase the system availability. The book describes systematic ways of fault-tolerant control, which give better solutions than ad hoc engineering based on experience and process knowledge. It utilises an analytic redundancy, which is cheaper than physically duplicating all vulnerable components. Note that the principle of reliability theory to build a reliable system by using less reliable components is applicable only if more components are used than necessary

for a given function. Fault-tolerant control does not always necessitate duplication of components but changes components (controllers) after faults have occurred.

Fault tolerance necessitates redundancies. One needs redundancies to detect faults by measuring all input and output signals. These measurements provide more information than the sole measurements of the input, which are sufficient for prediction tasks. On the other hand, redundant sensors or actuators are necessary for control reconfiguration. However, this does not mean that all sensors or actuators have to be implemented in duplicate. One additional sensor or actuator may provide analytical redundancy for every single sensor or actuator fault.

Performance degradation. In certain practical situations the performance specifications for the faulty system may be reduced in comparison to the faultless system. Clearly, the weaker the performance specifications the larger can the tolerable faults be.

Implementation. Another important issue results from the fact that fault-tolerant control methods cannot be sufficiently tested in operation (in contrast to control methods for the nominal system), because under practical circumstances it is usually impossible to provoke faults in the plant in order to test the reaction of the control system. It is, therefore, of high practical importance that the book presents systematic solutions to the analysis and design steps included in fault-tolerant control, the validity of which can be proved under the given assumptions. The implementation of the algorithms in the control equipment is not the subject of this book. To avoid faults in this step, methods for verification of control algorithms, for fault-tolerant computing and for fault-tolerant communication have to be used.

Severity of faults. A principal "threshold" for achieving fault tolerance is the fact that no method can guarantee a complete description of all possible faults of a system. Hence, 100 %-fault tolerance is impossible. However, for many applications, complete fault tolerance is not necessary. A reasonable application of fault-tolerant control starts with the selection of the most critical faults and continues with the investigation of fault tolerance against these faults.

Insignificant faults are difficult to detect but easy to compensate for, whereas severe faults are easy to identify but difficult to handle. This experience underlines the importance of fault diagnosis for fault-tolerant control.

1.4 Architecture of Fault-Tolerant Control

1.4.1 Architectural Options

The architecture of fault-tolerant control describes which components of the plant, the controller and the diagnostic system work together and which information is exchanged among these components. It is determined by different practical aspects

like the availability of computer resources, the character of the system to be controlled, which can have a large physical size or may be a small single entity, and the software structure used. These aspects will be considered in this book only with respect to the consequences for the diagnostic and control redesign methods.

The typical situation, which is mainly considered in the literature on fault-tolerant control, concerns the *embedded systems approach*, where the diagnostic and the controller redesign tasks are accomplished on a single computer board, which is directly connected to the system to be controlled. All measurement information are available on this board and, hence, all algorithms can utilise all information. This is the situation shown in Figs. 1.8, 1.13 and 1.14, where there is a single component for each task and all arrows represent perfect information links.

However, there are important practical circumstances under which the embedded systems structure cannot be applied and a distributed or a remote systems approach has to be used, where the fault-tolerant control algorithms and the available information are distributed among different components. These situations, which will be explained in the next paragraphs, have important consequences for the fault-tolerant control algorithms, because they distinguish from the embedded systems approach with respect to the information available. Either the algorithms have access only to a subset of the overall information used or the information links bring about severe time delays and even cause a dropout of data. These practical circumstances have to be taken into account when elaborating fault-tolerant control algorithms.

1.4.2 Distributed Systems

Distributed diagnosis. The term "distributed diagnosis" summarises three situations where the information is distributed among several components that are to ensure the fault tolerance of the system. Their main characteristics will be explained in the following for a diagnostic system, but the same considerations can be made for the controller redesign.

- **Distributed diagnosis** (in the narrow sense): The diagnostic system is designed as a unique entity and the resulting diagnostic algorithm is distributed over different components to cope with the computational effort needed. The result obtained is the same as in the embedded systems approach provided that the communication system does not restrict the performance. This method is elaborated for continuous systems in Chap. 10.
- **Decentralised diagnosis**: The diagnostic problem is decomposed into different subproblems which refer to the subsystems of the overall system under consideration. The subproblems are solved independent of each other. This approach is explained for discrete-event systems in Chap. 12.

- **Coordinated diagnosis**: Like in decentralised diagnosis the overall problem is decomposed, but the solutions obtained independently for the subproblems are combined by some coordinators to ensure their consistency.

Decentralised and coordinated diagnosis are illustrated by Figs. 1.15 and 1.16. Both methods are reasonable if the system to be diagnosed is composed of several interconnected subsystems. Usually such a structural decomposition is given by the physical system structure and the subsystems are often weakly coupled. This situation suggests a decomposition of the diagnostic task in such a way that the subtasks can be associated with the subsystems. It is then reasonable to implement N different diagnosers D_i for the N subsystems Σ_i of the overall system.

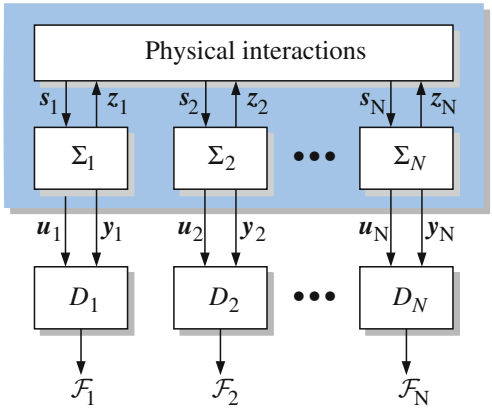


Fig. 1.15 Decentralised diagnosis

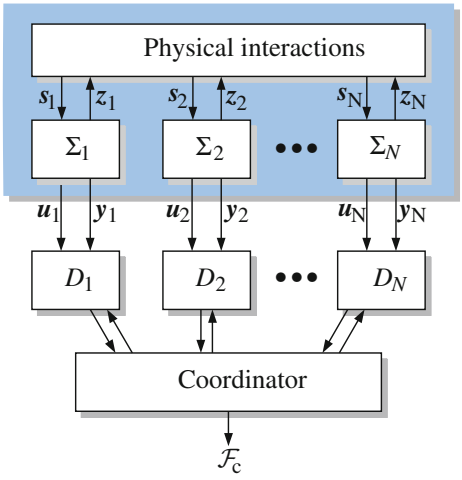


Fig. 1.16 Coordinated diagnosis

In decentralised diagnosis, the diagnoser D_i has available only the local input $\boldsymbol{u}_i$ and the local output $\boldsymbol{y}_i$. It solves its task by means of a model of the subsystem Σ_i. The result is given by the set $\mathcal{F}_i$ of fault candidates.

The main problem with this scheme is the consideration of the interactions among the subsystems, because the coupling signals $\boldsymbol{s}_i(t)$ and $\boldsymbol{z}_i(t)$, $(i = 1, 2, \ldots, N)$ are not assumed to be known to the diagnosers. Different approaches have been developed to solve this problem. The simplest way is to assume that there is no interaction, which means that the model used by the diagnoser D_i describes the isolated subsystem Σ_i. This is successful only if the interactions among the subsystems are weak. Other approaches use coarse models of the interactions. In any case, as there is no information exchange among the diagnosers, the overall diagnostic result

$$\mathcal{F}_\mathrm{d} = \cup_i \mathcal{F}_i$$

typically includes more fault candidates than the result $\mathcal{F}_g$ that a single diagnoser of the overall system would determine:

$$\mathcal{F}_\mathrm{d} \supseteq \mathcal{F}_\mathrm{g}.$$

This is the price for the complexity reduction of the diagnostic problem with respect to both the diagnostic algorithms applied and the information links to be implemented.

From an architectural point of view, the diagnosers are agents that solve their diagnostic problems independently of each other, which is in line with modern software structures and principles. However, as these explanations show, the overall diagnostic result is worse than a global solution.

The disadvantage of decentralised diagnosis compared to a centralised solution can be overcome by extending the diagnosers of the subsystems by a coordinator that combines the results $\mathcal{F}_i$ obtained for the subsystems to a result $\mathcal{F}_\mathrm{c}$ of the overall system. As the coordinator has a model of the interconnections of the subsystems, the overall result $\mathcal{F}_\mathrm{c}$ is better than the result of the decentralised diagnosis:

$$\mathcal{F}_\mathrm{c} \subseteq \mathcal{F}_\mathrm{d}.$$

If the link between the diagnosers and the coordinator is bidirectional, the coordinator can send information to the diagnosers that can be used to improve the local diagnostic results. The aim of the coordination is to retain the result of a global diagnosis

$$\mathcal{F}_\mathrm{c} = \mathcal{F}_\mathrm{g}.$$

Then the main advantage of coordinated diagnosis in comparison to a global diagnoser is the structure of the diagnostic system, which reduces the complexity of the overall algorithm.

Distributed control. From the viewpoints of the communication capacity and the local processing power, the distributed fault-tolerant control problem seems quite

comparable to the distributed diagnosis problem. Indeed, in decentralised control, each controller makes use of the locally available data $\boldsymbol{y}_i$ in order to produce the local control $\boldsymbol{u}_i$, while in distributed control it can use more data and therefore achieve better performance—or it can compensate for more faults. These possibilities are of course open under the condition that the communication system has the capacity to provide the data and the local computing device has the capacity to process them.

However, this is a rather simplistic view, because generally the overall system specifications are not *decomposable*, which means that they cannot be split into a set of independent local specifications for each subsystem. Because of the coupling variables it is well known, for example, that the interconnection of several stable systems does not necessarily result in a stable overall system. Conversely, if an unstable subsystem has lost all its actuators, the overall system might be stabilised by changing the controls of the other subsystem in such a way that it will be stabilised by the action of the coupling variables. Therefore, addressing fault-tolerant control for a distributed system is the problem of designing or redesigning several local controllers, whose actions interfere due to the coupling variables in such a way that the global specifications are satisfied, both in normal operation and in the presence of faults.

Redesigning the control law may result in recovery transients that are difficult to handle in embedded systems (Sect. 9.5 specifically addresses this problem). The recovery transient problem is even magnified in distributed systems, and the design of a fault-tolerant distributed control strategy has to consider the minimisation of the reconfiguration effort, which depends on the number of subsystems whose parameters or control laws are to be changed in order to recover a fault. These topics are addressed in Chap. 10, where a fault-tolerant control strategy is developed by which the set of data available to each local controller is increased. This strategy is called *information pattern reconfiguration*.

1.4.3 Remote Control and Diagnosis

Modern data communication networks provide the means to connect control and supervison components whenever data links can improve the performance of these components. Different architectures of networked control systems are currently investigated under the common headline of *cyberphysical systems* (Fig. 1.17), which consist of three layers: the physical layer that includes the system to be controlled as a composite system consisting of several subsystems Σ_i, $(i = 1, 2, \ldots, N)$, the layer of embedded systems, and the global network. From the control engineering viewpoint, the embedded systems are the computing facilities in which the control and supervision algorithms are implemented. These algorithms include means for fault diagnosis, feedback control and control reconfiguration. The data network can be used to connect any of these components. The flexibility of the communication structure on the one hand and the possible time delays and information packet dropouts on the other hand are characteristic elements that the global network introduces into the control system. To emphasise these network properties, the network is drawn as a cloud in the figures below.

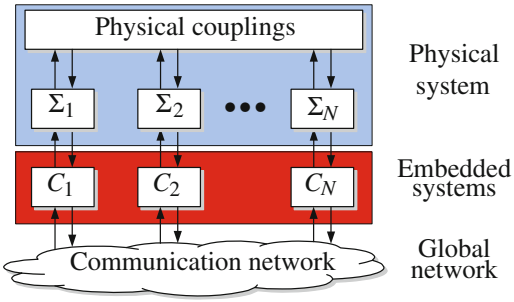


Fig. 1.17 Structure of a cyberphysical system

The digital network can be used in several ways to implement data connections between the sensors and actuators of the physical system and the controllers or among the control stations of a systems. Two architectures have attracted considerable interest in the literature on fault-tolerant control: networked diagnosis and remote diagnosis. The main ideas are explained in the following paragraphs:

Networked diagnosis. The notion of networked diagnosis is used if the sensor and actuator data are communicated over a digital network to the diagnostic units. If this communication is quick enough in comparison to the plant time constants, nothing is new in comparison to the structures considered before. However, the network has to be modelled as a separate element in the overall system if it may introduce substantial time delay due to heavy information traffic. Then diagnostic algorithms have to be developed and implemented that tolerate this time delay.

On the other hand, the communication network can be used to couple diagnostic units that run on the control equipment of separate subsystems. If these units do not interact, a decentralised diagnosis is implemented; otherwise a distributed or a coordinated structure is used. The specific properties of networked diagnosis appears if during the operation of the diagnostic units the communication structure is deliberately changed in order to improve the diagnostic result. This situation uses the flexibility offered by modern communication means. An important question asks how to adapt the communication structure to the intermediate diagnostic result. Methods have to be elaborated to answer this question in a systematic way and under real-time constraints.

Remote diagnosis. If the control unit, which is directly linked to the process under consideration, is not powerful enough to solve all its tasks it can be extended by remote components that are linked to the process via data networks like the Internet.

Figure 1.18 illustrates the situation of remote diagnosis for a car where the on-board component runs on the embedded control equipment of the car and the off-board component is implemented on a remote computer. The on-board diagnostic system has to cope with limited computation and memory resources whereas the remote system can take advantage of the larger computer capacity but has to solve its tasks by means of the restricted information obtained via the data network. This situation is typical also for other application areas like energy distribution networks

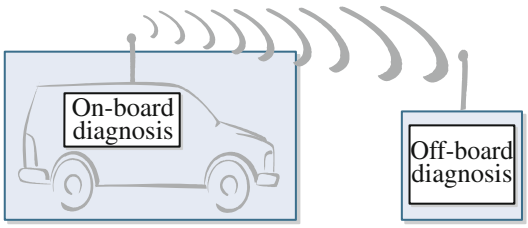


Fig. 1.18 Remote diagnosis

or building automation. The terms "on-board" or "off-board" components which are common in automotive applications are used here as general terms also for these other application fields.

In a general setting, remote diagnosis uses both an on-board and an off-board component. The practical circumstances under which these components have to work can be summarised as follows:

- The **on-board component** has to work with restricted computing power and memory size, which limits the algorithmic complexity of the task to be performed.
- The **off-board component** has (nearly) unlimited computing power but has to cope with limited and possibly biased measurement data.
- The **data link** causes time delays and data losses and restricts the amount of data that can be transmitted under real-time constraints.

The decomposition of the overall diagnostic task into subtasks for the on-board or the off-board component, respectively, has to take these restrictions into account.

The diagnostic process is usually structured in several diagnostic steps as described on p. 14, where the complexity of the model and the amount of measurement data to be used increase from fault detection over fault isolation towards fault identification. This fact together with the practical circumstances under which the on-board and the off-board component works lead to the following decomposition of the diagnostic task (Fig. 1.19):

- The **on-board component** solves the problem of fault detection. For this task, only the model of the faultless system is necessary. The result is a yes/no answer to the question whether a fault has occurred.
- The **off-board component** isolates and identifies the fault. These tasks can be solved only if detailed models of the faulty system together with fault models are available.

Both components work with appropriately selected input and output signals. All available input and output data are represented, respectively, by the sequences

$$\begin{aligned} V(0,\ldots,k_\mathrm{e}) &= (v(0), v(1), \ldots, v(k_\mathrm{e})) \\ W(0,\ldots,k_\mathrm{e}) &= (w(0), w(1), \ldots, w(k_\mathrm{e})). \end{aligned}$$

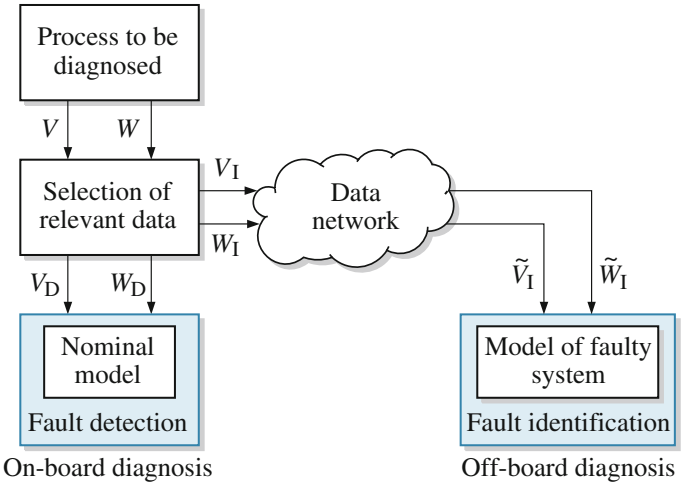


Fig. 1.19 Decomposition of the diagnostic task

Only part of these sequences have to be used for fault detection. That is, some data included in the sequence $V(0, \ldots, k_e)$ can be deleted to get the reduced sequence $V_D(0, \ldots, k_e)$. The same happens with the sequence $W(0, \ldots, k_e)$ to get the sequence $W_D(0, \ldots, k_e)$ used for fault detection.

A similar selection process concerns the data $V_I(0, \ldots, k_e)$ and $W_I(0, \ldots, k_e)$ used for fault identification. These data are transmitted over the data network, where data losses may change them into the new sequences $\tilde{V}_I(0, \ldots, k_e)$ and $\tilde{W}_I(0, \ldots, k_e)$ which are received by the off-board component.

Figure 1.19 shows that a design problem of remote diagnosis is to decide which data should be used by the on-board diagnostic component and which data should be transmitted over the data network to the off-board component. A further design problem concerns the adaptation of this selection to the intermediate diagnostic result. If the data link is used in a bidirectional way, the off-board component can send requests towards the data selection block in order to obtain those specific data that can bring about the best possible progress of the fault isolation or identification tasks.

An important issue of remote diagnosis is the asynchronous operation mode of the on-board and off-board components. Due to the information link, which is used only in certain time intervals and brings about time delays, both components cannot be synchronised but their activities have to be structured in such a way that they tolerate the asynchronous operation modes.

The scheme depicted in Fig. 1.19 is general enough to be applicable for the remote diagnosis of continuous-variable as well as discrete-event systems. It can be extended to fault-tolerant control, where the on-board component is either fault-tolerant itself or obtains accommodation or reconfiguration commands from the off-board component. This fault tolerance extends the autonomy of the on-board component.

1.5 Survey of the Book

Compared with the well-known controller design task, the main new problems to be solved in fault-tolerant control can be summarised as follows:

- **Modelling of dynamical systems subject to faults**.
  The dynamical model of the plant should not only describe the faultless, but also the faulty system for all faults $f \in \mathcal{F}$. Hence, it is not sufficient to have the model available, which has been used for the design of the nominal controller, but this model has to be extended for the fault cases. Furthermore, for the solution of the diagnostic problem and for the selection of reasonable control configurations, model classes other than differential equations or automata tables have to be used. Consequently, this book presents alternative means for describing dynamical systems, which are appropriate to answer the basic questions of fault-tolerant control. It is structured according to these models, where each of the Chaps. 4 through 12 deal with another kind of models and structures of fault-tolerant control.

- **Analysis of fault effects**.
  Fundamental problems concern the fault propagation through the system and the diagnosability of the faults. The analysis has to show whether the selected measurements provide sufficient information for detecting, isolating and identifying faults. For the controller redesign, the controllability and observability of the faulty system is important. Any fault-tolerant control has to rely on redundancies in the system, which can be activated to stop the evolution of the fault. These fundamental properties depend upon the structure of the system under investigation and, consequently, Chaps. 4 and 5 deal with structural models and structural analysis methods for investigating these properties.

- **Methods for fault detection and isolation**.
  The diagnostic methods explained in this book have been selected for the purpose of fault-tolerant control. Emphasis is laid on methods that do not only detect, but also isolate or identify faults. The structural analysis for finding analytical redundancy relations for fault detection and isolation in continuous-variable systems explained in Chap. 5 and the diagnostic methods for discrete-event systems described in Chap. 11 provide novel means of fault identification, which have not yet been published in a monograph or textbook.

- **Redesign of the controller**.
  Fault accommodation and control reconfiguration methods include severe extensions of well-known controller design methods, because they have to be carried out completely automatically without any interaction of a human designer. A further important issue is the reconfigurability analysis explained in Chaps. 4, 5 and 8. Chapter 8 presents the two possible fault-tolerance strategies, namely fault accommodation and control reconfiguration. The reconfiguration strategy is analysed from a global perspective that includes the specification and the development of control solutions, as well as their implementation and their evaluation. Chapter 9

develops specific approaches to the fault accommodation and system reconfiguration problems.

- **Implementation problems**.
  Because of the real-time constraints of fault-tolerant control there is a trade-off to be addressed between the online redesign of the control law (small memory requirements, but delayed fault recovery due to the redesign procedure) and the switching to a specific control law in a pre-computed bank of control laws (larger memory requirements but faster response to the fault occurrence). This trade-off is addressed in Chap. 8, where a mixed passive–active strategy is introduced to design a bank of controllers while minimising the number of elements in the bank. In distributed systems, a similar trade-off appears between the performance of local controllers and the amount of data that are to be communicated through the communication network (powerful local controllers imply more communicated data). Similarly, the performance of local diagnosers depends on the data they are provided with. Chapter 10 presents an approach based on the notions of minimal information patterns and minimal communication costs that achieve the desired diagnostic result.

- **Fault-tolerance evaluation**.
  An important issue in engineering design is the evaluation of the design result. Whereas for diagnostic systems evaluation criteria have been introduced, e.g. false alarm or missed detection probabilities, average time between false alarms, detection delays, mis-isolation measures, there are only a few approaches to the evaluation of the efficiency of fault-tolerance strategies. Roughly speaking, a fault-tolerance strategy is more efficient if it allows the recovery of more faults, in a faster way, with better performances, etc. Fault-tolerance evaluation is a research field still to be developed and this book presents some preliminary results in Chap. 8, which introduces several deterministic or probabilistic measures.

- **Architectures of fault-tolerant control**.
  The basic ideas of fault diagnosis and fault-tolerant control will be explained for a centralised structure, where one diagnostic or control unit is responsible for both tasks. However, in applications many systems consist of several subsystems that are physically coupled. For such systems, the control and supervision components are distributed over the subsystems and have to accomplish their tasks either independently of each other or with information exchange. Distributed architectures are introduced in Chap. 10 for continuous-variable systems and in Chap. 12 for discrete-event systems.

Outline of the book. The book is structured into three main parts and an introduction. The introduction (Chaps. 1 and 2) describes the main problems of fault-tolerant control and introduces two running examples that will be used throughout the book to illustrate the ideas and methods developed.

Part I of the book (Chaps. 3–5) summarises the models used for fault-tolerant control and explains methods for the structural analysis of dynamical systems subject to

faults. The main results to be obtained by the methods presented refer to the possibility to detect and identify faults (diagnosability analysis), the ways that fault influence a system (fault propagation analysis) and ways to circumvent faults (reconfigurability analysis). All results are obtained by means of global models that describe the system under consideration in terms of its components or that represent the couplings of the signals describing the system behaviour.

Part II deals with continuous-variable systems (Chaps. 6–10). It summarises results on fault diagnosis of deterministic or stochastic systems, explains how to analyse the recoverability of systems with respect to faults and introduces new methods for fault accommodation and reconfigurable control. Chapter 10 extends these methods to interconnected systems where the diagnostic units have to be distributed over the subsystems.

Part III is devoted to discrete-event systems (Chaps. 11 and 12). It concentrates on systems described by non-deterministic or stochastic automata and explains the main ideas of fault detection with centralised and decentralised structures.

In summary, this book covers a large range of problems and methods of fault-tolerant control starting with modelling of faulty systems, presenting diagnostic methods for different kinds of dynamical systems and finishing with new methods for fault accommodation and control reconfiguration. With this scope, it includes a lot of material that has not yet been published in a unified form. This particularly concerns structural methods of fault-tolerant control and diagnosis of discrete-event systems. The aim is not to provide an exhaustive survey of all methods but rather to give a detailed presentation of important methods and tools that proved to be effective in applications. Precise algorithmic descriptions, guidelines for parameter tuning and examples should help the reader to gain a thorough understanding of the material.

Comparison to other monographs. Several monographs have appeared during the last decade in the area of fault diagnosis and a few on fault-tolerant control. The following remarks should explain how this book differs from these recent publications.

Most of the monographs are restricted to a relatively narrow and advanced research topic, but describe their subjects in many details like the book [228] on robust estimation and its use for fault detection, [54] on active fault detection with the design of proper auxiliary signals, [190] on diagnosis of a specific class of discrete-event systems called active systems, [226, 227] on active fault-tolerant control systems with Markovian parameters, and [5] for sliding mode concepts of fault-tolerant control. Statistical tests represent broadly used methods to deal with uncertainties in fault diagnosis as explained in detail in [79] and demonstrated by several applications. An alternative method uses set-theoretic approaches to represent and process uncertainties [342].

The textbooks [64, 78, 124, 316] are essentially dedicated to fault diagnosis based on analytical models of the supervised process by observer-based approaches, parity space approaches and parameter identification techniques. The first one provides a thorough study of the observer-based approach including robustness issues and an introduction to nonlinear systems. The second one is essentially dedicated to the parity space approach, both in a deterministic and a stochastic context. It also

includes a discussion of robustness issues and an introduction to statistical testing and parameter identification methods.

The very recent books [156, 157] give a large survey of various methods for fault diagnosis and design of fault-tolerant structures and their applications. Fault diagnosis is tackled with model-based approaches (observer-based, parity space, and parameter identification methods), data-based techniques (simple single signal-based methods and classification approaches). Besides, basic redundant structures are presented for fault-tolerant control. Data-driven methods are thoroughly discussed in the monographs [182, 383] which also briefly introduce analytical and knowledge-based methods. These books also explain diagnostic methods based on fuzzy set theory and artificial neural networks.

Furthermore, several monographs on specific application areas appeared, for example, [395] on fault-tolerant control of aerospace vehicles, [91] on a flight control benchmark problem and [254] on process applications.

1.6 Bibliographical Notes

Consistency-based diagnosis. The logical background of consistency-based diagnosis is that experiments which are consistent with a given conjecture do not prove the conjecture to be true, but inconsistency indeed proves it false. This is the basis of the growth of scientific knowledge as analysed by the philosopher of science, Sir KARL POPPER [278]. The interested reader may also consult [279] for an intellectual biography.

Consistency-based diagnosis is a general diagnostic principle, which compares the measurements with the behaviour of some model. This idea has been elaborated first in the field of artificial intelligence [139, 212]. In Chap. 9 of the monograph [8] the behavioural notation has been used to show the common foundation of quantitative and qualitative methods for diagnosis. This interpretation of diagnosis uses the notion of the system behaviour defined in [384]. Several attempts have been made to combine diagnostic methods elaborated in control engineering and artificial intelligence [18].

Fault detection and fault isolation. Fault detection has been the subject of long research with [148, 273] as two of the earliest descriptions of the field. [79, 124, 266] provide a broad look at the current state of the art for continuous-variable systems, for which diagnostic methods are mainly based on state observation, on the parity space approach and on parameter estimation techniques. The monograph [316] gives a thorough introduction into fault diagnosis by means of identification techniques. [233] describes the main methods for evaluating the dependability of engineering systems in a broader perspective. The different structures of centralised, decentralised and coordinated diagnosis have been discussed for discrete-event systems in [235], the main issues of remote diagnosis in [114, 302].

Fault-tolerant control. Fault accommodation methods have been developed in the 1990s based on robust and adaptive control. The third approach is based on the switching among controllers, which have been designed offline for different fault situations. Surveys of these methods are given in [217, 264, 285, 410].

The systematic treatment of fault-tolerant control by reconfiguring the control loop concerned aerospace examples with [117, 151] being two of the earliest papers that used model matching or a pseudoinverse technique to give the new control loop a similar performance as the nominal closed-loop system. A major impetus for the development of new methods has been given by the COSY-benchmark problems published in [163]. Solutions to these problems which have been obtained by alternative methods are described in Chaps. 12 and 13 of the monograph [8].

Fault-tolerant control of networked systems. Data networking poses restrictions with respect to the amount of data received, time delays and packet dropouts, all of which are introduced by the network. These aspects have been investigated, for example, in [234, 272, 299, 379, 380, 407].

Chapter 2
Examples

Abstract This chapter illustrates the main problems of diagnosis and fault-tolerant control by means of three examples, which will be used later in the text.

2.1 Two-Tank System

Description of the system. As the first example, consider the two-tank system depicted in Fig. 2.1. The pump causes a liquid flow q_P into Tank 1 where the input $u(t)$ describes the pump velocity. u is determined by a security switch-off, which prevents an overflow of Tank 1. The inputs to the tank system prescribe the valve positions V_a and V_{12}. The only measured signal is the outflow q_M. Hence, the tank system results in the simple block diagram shown in Fig. 2.2.

In the faultless case, the valve V_a is closed and the valve V_{12} is used to control the level of Tank 2. Only in case of a valve fault, the upper pipe is used for this purpose.

The control aim results from the requirement of a batch process, in which the outflow of Tank 2 is used in succeeding parts of cascaded vessels and reactors. The valve V_{12} is used to fill and refill Tank 2 accordingly, where Tank 1 is a storage tank, which is to be filled to the height h_{max}, at which the security switch-off stops the pump.

Faults. Two faults are considered. First, a leakage in Tank 1 may occur, which causes the additional flow q_L out of Tank 1. The "size" of the leakage is given by the parameter c_L (Table 2.1). The different approaches to fault diagnosis presented in the following chapters use this two-tank example with different notions of this fault. Either a parametric fault is considered where c_L denotes the fault size to be identified or a symbolic fault f is used which represents the faulty situation with the outflow $q_L = c_L\sqrt{h_1}$ out of Tank 1 where c_L is the parameter given in Table 2.1.

The second fault is a blockage of the valve V_{12} in the closed position. This fault can be modelled by setting the valve constant c_{12} to zero.

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8_2

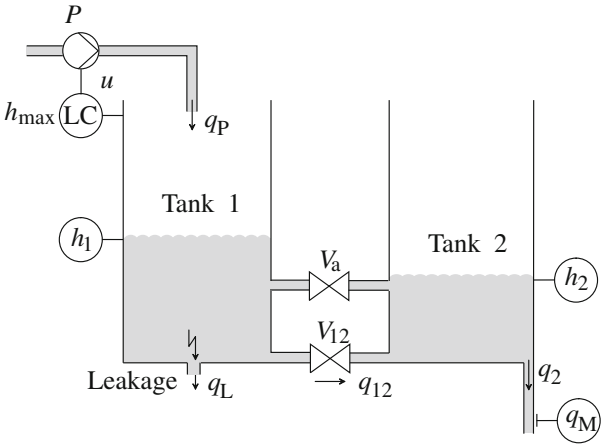
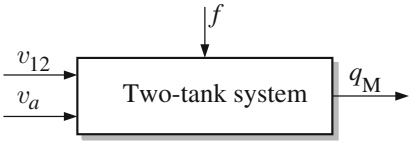


Fig. 2.1 Two-tank system

Fig. 2.2 Block diagram of the tank system



The example is used for illustrating the following diagnostic and fault-tolerant control problems:

- **Fault detection**: Determine whether a fault has occurred. The valve fault can be detected due to the decreasing outflow from Tank 2, which eventually vanishes. For the leakage the problem is more involved because neither the inflow q_P nor the level h_1 is assumed to be measured. Hence, the stationary outflow from Tank 2 is the same as before and the leakage and the fault can only be found by small dynamical effects that are "visible" in the outflow measurement just after the leakage occurs.

- **Fault isolation**: Determine which part of the system is faulty.

- **Fault identification**: Determine the size of the leakage.

- **Fault accommodation**: Design a fault-tolerant level controller that maintains the liquid level in Tank 1 at a given set-point independently of whether the leakage is present or not.

- **Control reconfiguration**: In case of the valve fault, the auxiliary valve V_a has to be used. The reconfiguration problem includes to *automatically* find that switching to the second pipe is the strategy to apply.

These problems are considered under different circumstances where the tank levels or the outflow from Tank 2 are measured numerically or in a quantised way. Therefore, different models are appropriate to describe the tank system and different methods have to be used to solve the diagnostic problem.

For this simple example, it is obvious under what conditions and how the given problems can be solved. If the pump is controlled according to level measurement h_1, a static tank level will be reached. However, the leakage can only be found if the dynamical changes of the tank levels or of the outflow from Tank 2 are taken into account, because the pump is assumed to be strong enough to maintain the level of Tank 1 at the prescribed value even in case of the leakage. Hence, the system has the same static behaviour with and without the fault.

The diagnostic result will certainly be different if the outflow is the only measurement compared with the case in which all tank levels are measured. Also, the diagnostic problem becomes more difficult if instead of the numerical values of the tank levels only quantised measurements are possible.

If q_P is an additional measurement, the fault can be detected by comparing the mean value of q_P with its nominal value. If this value is increased, more liquid flows out of Tank 1 which under the given circumstances can only occur if the tank has a leakage.

The fault-tolerant controller is simply found as a PI-feedback of the tank level h_1 towards the pump velocity u_P. For reasonable leakages (reasonable values of q_L) the controller is able to hold the level at the prescribed value even if the fault occurs.

Model of the tank system. The two tanks have the liquid levels $h_1(t)$ and $h_2(t)$, which are used as state variables in the model given below. The liquid flows are

Table 2.1 Signals and parameters of the tank system

Parameter	Value and unit	Meaning
h_1 , h_2	[m]	Tank levels in meters
q_M	[l/min]	Measured outflow in litres per minute
u	[1]	Control input to the pump
q_{12} , q_2 , q_P , q_L	[m^3/s]	Volume flows in cubic metres per second
A	$1.54 \cdot 10^{-2} m^2$	Cross-section area of both tanks
h_{max}	0.60 m	Height of both tanks
u_{nom}	1.0	Nominal pump velocity
u_{max}	5.0	Maximal pump velocity
c_{12}	$6.0 \cdot 10^{-4} m^{5/2}/s$	Flow constant of valve V_{12}
c_2	$2.0 \cdot 10^{-4} m^{5/2}/s$	Flow constant of the outlet of Tank 2
c_L	$8.0 \cdot 10^{-4} m^{5/2}/s$	Flow constant of a leakage in Tank 1
c_M	$12.0\ l/(min \cdot m^{1/2})$	Constant of outflow sensor
$\bar{q}_P$	$1.5 \cdot 10^{-4} m^3/s$	Flow constant of the pump

denoted by q, the ground area of the cylindric tanks by A. The parameters used in the example are summarised in Table 2.1.

The following Eqs. (2.1), (2.2) describe the mass balance, where the tank levels h_1 and h_2 are related to the liquid flows indicated in Fig. 2.1 as follows:

$$\dot{h}_1(t) = \frac{1}{A}(q_{\mathrm{P}}(t) - q_{\mathrm{L}}(t) - q_{12}(t)) \tag{2.1}$$

$$\dot{h}_2(t) = \frac{1}{A}(q_{12}(t) - q_2(t)). \tag{2.2}$$

The measured signal q_{M} is proportional to the outflow q_2:

$$q_{\mathrm{M}} = c_{\mathrm{M}} \cdot q_2. \tag{2.3}$$

The different flows used in the equations above can be obtained by TORICELLI's law:

$$q_{12}(t) = \begin{cases} c_{12}\ \mathrm{sign}\ (h_1(t) - h_2(t))\ \sqrt{|h_1(t) - h_2(t)|} & \text{if } V_{12} \text{ is open} \\ 0 & \text{else} \end{cases} \tag{2.4}$$

$$q_2(t) = \begin{cases} c_2\sqrt{h_2(t)} & \text{if } h_2(t) > 0 \\ 0 & \text{else,} \end{cases} \tag{2.5}$$

$$q_{\mathrm{P}}(t) = \begin{cases} u(t) \cdot \bar{q}_{\mathrm{P}} & \text{if } h_1(t) \le h_{\max} \\ 0 & \text{else,} \end{cases} \tag{2.6}$$

$$q_{\mathrm{L}}(t) = \begin{cases} c_{\mathrm{L}}\sqrt{h_1(t)} & \text{if } h_1(t) > 0 \text{ and Tank 1 has a leakage} \\ 0 & \text{else.} \end{cases} \tag{2.7}$$

The pump is controlled by the security switch-off included in the level controller LC shown in the figure such that the level in Tank 1 is maintained below the height $h_{\max}$. The pump velocity is given by the control input u_{P}. Its nominal value is given by $u = u_{\mathrm{nom}}$, and its maximal value by $u_{\max}$. If a control problem should be illustrated in the later chapters, then Eq. (2.6) is supplemented with an equation describing the control law $u = k(h_1)$.

The equations given above are hybrid because they include differential and algebraic equations as well as switching conditions, which result from the physical laws and from a security switch installed at Tank 1. Therefore, the differential equation includes several inequalities that describe the validity range of the given functions.

The tank will be used in many places to illustrate methods and results. For simplicity, often the parameter A is set to one so that the model gets the simpler form

$$\dot{h} = q_i(t) - q_o(t),$$

where q_i and q_o denote the input and the output flow.

2.2 Three-Tank System

Consider the three coupled tanks depicted in Fig. 2.3. These tanks are connected by pipes which can be controlled by different valves. Water can be filled into the left and right tanks using two identical pumps. Measurements available from the process are the continuous water levels h_i of each tank and, additionally, from tank T_2 discrete signals from two capacitive proximity switches signalling whether the water level in the tank is above or below the position of the sensor.

In the nominal case (Fig. 2.4), only the left tank T_1 and the middle tank T_2 are used. The right tank T_3 and pump P_2 act as redundant hardware. The purpose of the system is to provide a continuous water flow $q_2(t) = q_N$ to a consumer. Therefore, the water level in the middle supply-tank T_2 has to be maintained within the interval $h_{2\mathrm{L}} < h_2 < h_{2\mathrm{H}}$, i. e. between the two discrete level sensors of tank T_2.

Water flows between the tanks can be controlled by several valves ($V_{12\mathrm{L}}$, $V_{12\mathrm{H}}$, $V_{23\mathrm{L}}$, $V_{23\mathrm{H}}$). All valves can only be completely opened or completely closed (on/off valves). The connection pipes between the tanks are placed at the bottom of the tanks (pipes with valves $V_{12\mathrm{L}}$, $V_{23\mathrm{L}}$) and at a height of h_{H} (pipes with valves $V_{12\mathrm{H}}$, $V_{23\mathrm{H}}$). One of the considered faults is a leakage in tank T_1 (see below). If such a leakage occurs, there is an additional outflow q_{L} of tank T_1 (cf. Fig. 2.3).

Dynamical model. Depending on the water levels and the position of the valves, different nonlinear state-space models are valid. In general, the water flow q_{ij} from Tank i to Tank j can be calculated using TORICELLI's law

$$q_{ij} = c_{ij} \cdot \operatorname{sign}(h_i - h_j) \cdot \sqrt{|h_i - h_j|},$$

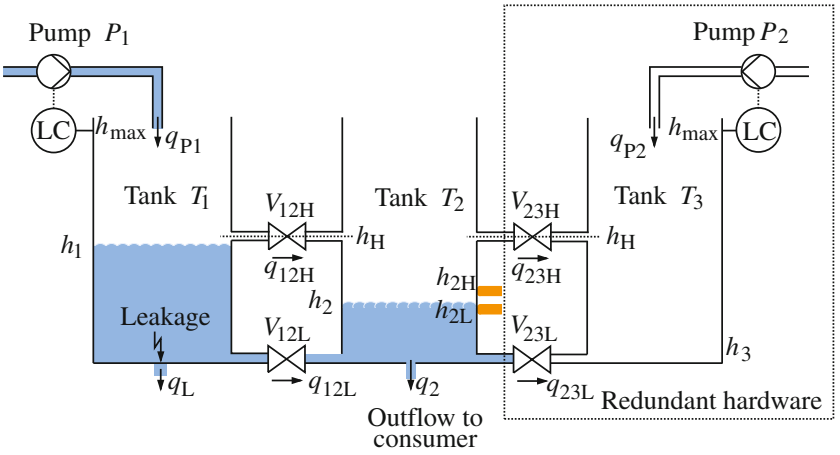


Fig. 2.3 Three-tank system

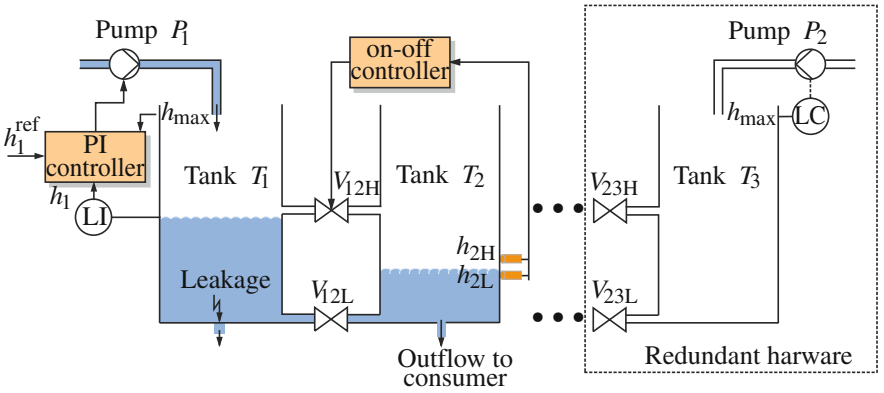


Fig. 2.4 Nominal configuration of the three-tank system

where c_{ij} is a constant depending on the geometry of the connecting pipe and the valve and h_i, h_j are the water levels. The change of water volume V in a tank is described by

$$\dot{V} = A \cdot \dot{h} = \sum q_{\text{in}} - \sum q_{\text{out}}, \tag{2.8}$$

where $\sum q_{\text{in}}$ is the sum over all water inflows and $\sum q_{\text{out}}$ the sum over all water outflows of the tank. In (2.8), A is the cross-section area and h the water level in the cylindric tank. For the three tanks Eq. (2.8) yields:

$$\dot{h}_1 = \frac{1}{A}(q_{\text{P1}} - q_{12\text{L}} - q_{12\text{H}} - q_{\text{L}}) \tag{2.9}$$

$$\dot{h}_2 = \frac{1}{A}(q_{12\text{L}} + q_{12\text{H}} - q_{23\text{L}} - q_{23\text{H}} - q_2) \tag{2.10}$$

$$\dot{h}_3 = \frac{1}{A}(q_{\text{P2}} + q_{23\text{L}} + q_{23\text{H}}). \tag{2.11}$$

The flows in Eqs. (2.9)–(2.11) depend on the levels h_1, h_2 and h_3 as well on the position of the valves and the commands u_{P1}, u_{P2} given to the pumps. For example, the existence of the flow $q_{12\text{H}}$ depends on the water levels h_1 and h_2 and the position of the valve $V_{12\text{H}}$. The flow is only non-zero if the valve is open and at least one liquid level exceeds the height h_{H} of the upper connecting pipe.

More precisely, the following expressions are obtained for the flows, with the parameters given in Table 2.2:

$$q_{\text{P1}} = \begin{cases} c_{\text{P1}} \cdot u_{\text{P1}} & \text{if } h_1 \le h_{\max} \text{ and } c_{\text{P1}} \cdot u_{\text{P1}} < q_{\text{P1}}^{\max} \\ q_{\text{P1}}^{\max} & \text{if } h_1 \le h_{\max} \text{ and } c_{\text{P1}} \cdot u_{\text{P1}} \ge q_{\text{P1}}^{\max} \\ 0 & \text{otherwise,} \end{cases}$$

$$q_{\mathrm{P2}} = \begin{cases} c_{\mathrm{P2}} \cdot u_{\mathrm{P2}} & \text{if } h_3 \le h_{\max} \text{ and } c_{\mathrm{P2}} \cdot u_{\mathrm{P2}} < q_{\mathrm{P2}}^{\max} \\ q_{\mathrm{P2}}^{\max} & \text{if } h_3 \le h_{\max} \text{ and } c_{\mathrm{P2}} \cdot u_{\mathrm{P2}} \ge q_{\mathrm{P2}}^{\max} \\ 0 & \text{otherwise,} \end{cases}$$

$$q_{12\mathrm{L}} = \begin{cases} c_{12\mathrm{L}} \ \operatorname{sign}(h_1 - h_2)\sqrt{|h_1 - h_2|} & \text{if } V_{12\mathrm{L}} \text{ open} \\ 0 & \text{otherwise,} \end{cases}$$

$$q_{12\mathrm{H}} = \begin{cases} c_{12\mathrm{H}}\sqrt{|h_1 - h_{\mathrm{H}}|} & \text{if } h_1 > h_{\mathrm{H}},\ h_2 \le h_{\mathrm{H}},\ V_{12\mathrm{H}} \text{ open} \\ -c_{12\mathrm{H}}\sqrt{|h_2 - h_{\mathrm{H}}|} & \text{if } h_1 \le h_{\mathrm{H}},\ h_2 > h_{\mathrm{H}},\ V_{12\mathrm{H}} \text{ open} \\ c_{12\mathrm{H}} \ \operatorname{sign}(h_1 - h_2)\sqrt{|h_1 - h_2|} & \text{if } h_1 > h_{\mathrm{H}},\ h_2 > h_{\mathrm{H}},\ V_{12\mathrm{H}} \text{ open} \\ 0 & \text{otherwise,} \end{cases}$$

$$q_{23\mathrm{L}} = \begin{cases} c_{23\mathrm{L}} \ \operatorname{sign}(h_2 - h_3)\sqrt{|h_2 - h_3|} & \text{if } V_{23\mathrm{L}} \text{ open} \\ 0 & \text{otherwise,} \end{cases}$$

$$q_{23\mathrm{H}} = \begin{cases} c_{23\mathrm{H}}\sqrt{|h_2 - h_{\mathrm{H}}|} & \text{if } h_2 > h_{\mathrm{H}},\ h_3 \le h_{\mathrm{H}},\ V_{23\mathrm{H}} \text{ open} \\ -c_{23\mathrm{H}}\sqrt{|h_3 - h_{\mathrm{H}}|} & \text{if } h_2 \le h_{\mathrm{H}},\ h_3 > h_{\mathrm{H}},\ V_{23\mathrm{H}} \text{ open} \\ c_{23\mathrm{H}} \ \operatorname{sign}(h_2 - h_3)\sqrt{|h_2 - h_3|} & \text{if } h_2 > h_{\mathrm{H}},\ h_3 > h_{\mathrm{H}},\ V_{23\mathrm{H}} \text{ open} \\ 0 & \text{otherwise,} \end{cases}$$

$$q_2 = \begin{cases} c_2\sqrt{h_2} & \text{if } h_2 > 0 \\ 0 & \text{otherwise,} \end{cases}$$

$$q_{\mathrm{L}} = \begin{cases} c_{\mathrm{L}}\sqrt{h_1} & \text{if } h_1 > 0 \text{ and leakage in tank 1} \\ 0 & \text{otherwise.} \end{cases}$$

Nominal configuration. In the nominal case, valves $V_{12\mathrm{L}}$, $V_{23\mathrm{H}}$, $V_{23\mathrm{L}}$ are closed and not in use. Valve $V_{12\mathrm{H}}$ is used to control the water level in tank T_2, pump P_1 to control the level in tank T_1. To control the water levels in the reservoir-tank T_1 and the supply-tank T_2, a conventional PI-controller and an discrete (on–off) controller are used (Fig. 2.4):

$$\begin{aligned} u_{\mathrm{P1}}(t) &= k(h_1(t), h_1^{\mathrm{ref}}) \\ &= K_{\mathrm{P}} \cdot (h_1^{\mathrm{ref}} - h_1(t)) + K_{\mathrm{I}} \cdot \int_0' (h_1^{\mathrm{ref}} - h_1(\tau))d\tau \end{aligned} \tag{2.12}$$

$$V_{12\mathrm{H}} = \begin{cases} \text{open} & : h_2 \le h_{2\mathrm{L}} \\ \text{close} & : h_2 \ge h_{2\mathrm{H}} \\ \text{no change} & : h_{2\mathrm{L}} < h_2 < h_{2\mathrm{H}}, \end{cases} \tag{2.13}$$

where K_{P} and K_{I} are controller parameters and h_1^{ref} is the set-point for tank T_1. Equation (2.13) describes under what conditions the on–off controller changes the position of the valve from opened to closed or vice-versa. All parameters of the controllers are given in Table 2.2.

Table 2.2 Parameters and variables of the three-tank system and the controllers

h_1, h_2, h_3	[m]	Tank levels in meters
q_{P1}, q_{P2}, q_2, q_L	[m^3/s]	Volume flows in cubic metres per second
q_{12L}, q_{12H}	[m^3/s]	Volume flows in cubic metres per second
q_{23L}, q_{23H}	[m^3/s]	Volume flows in cubic metres per second
A	$1.54 \cdot 10^{-2} m^2$	Cross-section area of the three tanks
h_{max}	0.60 m	Height of the three tanks
h_H	0.60 m	Height of the three tanks
c_{12L}	$1.6 \cdot 10^{-4} m^{5/2}/s$	Flow constant of valve V_{12L}
c_{12H}	$1.6 \cdot 10^{-4} m^{5/2}/s$	Flow constant of valve V_{12H}
c_{23L}	$1.6 \cdot 10^{-4} m^{5/2}/s$	Flow constant of valve V_{23L}
c_{23H}	$1.6 \cdot 10^{-4} m^{5/2}/s$	Flow constant of valve V_{23H}
c_2	$1.6 \cdot 10^{-4} m^{5/2}/s$	Flow constant of the outlet of tank 2
c_L	$1.6 \cdot 10^{-4} m^{5/2}/s$	Flow constant of a leakage in tank 1
c_{P1}	$1.0 \cdot 10^{-4} m^3/s$	Flow constant of pump 1
c_{P2}	$1.0 \cdot 10^{-4} m^3/s$	Flow constant of pump 2
q_{P1}^{max}	$1.0 \cdot 10^{-4} m^3/s$	Maximum flow of pump 1
q_{P2}^{max}	$1.0 \cdot 10^{-4} m^3/s$	Maximum flow of pump 2
h_1^{ref}	0.50 m	Set point of PI controller
K_P	10.0 1/m	Proportional gain of PI controller
K_I	$5.0 \cdot 10^{-2}$ 1/ms	Integral gain of PI controller
h_{2L}	0.09 m	Position of lower discrete level sensor
h_{2H}	0.11 m	Position of upper discrete level sensor

In summary, the nominal behaviour is characterised by the following:

- Only the left tank and middle tank are in use, water level h_2 must be medium, the set-point for h_1 is chosen to h_1^{ref}.
- Valves V_{12L}, V_{23L}, V_{23H} are closed.
- No leakage occurs ($q_L = 0$).
- The PI-controller (2.12) controls the level h_1 of tank T_1 with pump P_1 using a continuous level sensor.
- The on–off controller (2.13) controls the level h_2 of tank T_2 with valve V_{12H} using discrete level sensors.

Reconfiguration problem. Three different fault scenarios are given:

1. Fault f_1: Valve V_{12H} is closed and blocked.
2. Fault f_2: Valve V_{12H} is opened and blocked.
3. Fault f_3: A leakage in Tank T_1 occurs ($q_L \neq 0$).

The reconfiguration task is to find *automatically* a new control configuration of the three-tank system such that

- the water level h_2 remains between $h_{2\mathrm{L}}$ and $h_{2\mathrm{H}}$ for all scenarios, i. e. the relation

$$[h_2(k)] = \text{medium} \tag{2.14}$$

  should hold for $k \geq \bar{k}$ for a possibly small $\bar{k}$.
- for scenario 3, the loss of water is minimal, i.e.

$$[h_1(k)] = \text{empty} \tag{2.15}$$

  should hold for $k \geq \bar{k}$ for a possibly small $\bar{k}$.

The reconfiguration task consists in finding a new control structure by selection of actuators and sensors, new control laws and new set-points for the control loops, such that the control aims above are met. If needed, the use of redundant hardware components is possible. Obviously, the idea of reconfiguration cannot be satisfied by simply changing the parameters K_{P} or K_{I}, but a structural change of the system is necessary.

2.3 Ship Steering and Track Control

Ship navigation and steering is used as an example to illustrate different methods in both diagnosis and fault-tolerant control. A ship is illustrated in Fig. 2.5. The ship is steered by its rudder, the angle of which is δ. The ships heading angle is denoted ψ, the turn rate ω_3. The ship velocity ahead is v_1, velocity sideways is v_2.

To navigate a ship, information is needed on its position and heading angle as a minimum. In confined waters, distance is needed to a desired track that the ship is supposed to follow.

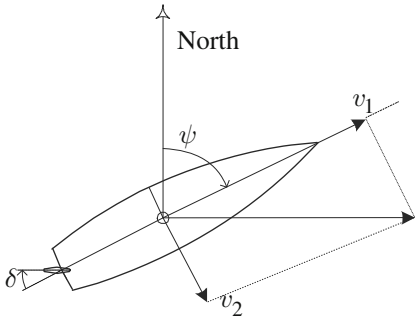


Fig. 2.5 Motion of a ship steered by its rudder. A rudder angle to port side (*left*) generates a turn to the port side of the ship. When turning to port, there is also a side velocity towards starboard (*right*)

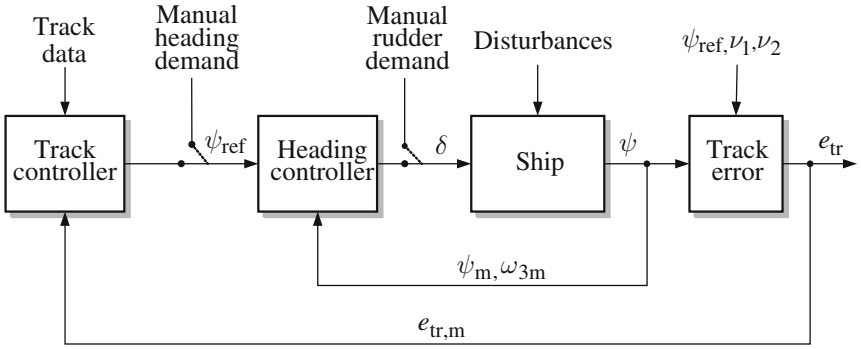


Fig. 2.6 Cascaded architecture of controllers for ship steering. The innermost loop is manual steering with rudder demand as input. The second loop provides automatic heading control, the third implements automatic track control

Should navigation data be wrong, ships may collide with banks or with other vessels. As unexpected manoeuvres can have fairly serious consequences, natural performance requirements exist to diagnosis and fault-tolerant control algorithms. Requirements are derived from the maximal motion the ship could make before a fault was diagnosed and a remedial action taken.

Control modes. In our ship steering example, three levels of steering control are considered:

- Hand steering. The rudder demand is manually set by a helmsman.
- Course control. An autopilot sets the rudder demand according to the deviation between instantaneous heading and a demanded course (heading reference). The ship's turn rate is used for derivative control action.
- Track control. A set of way-points specify a desired track for the ship to follow. The distance of the ship to the track is calculated and used by the track controller to command a heading reference to the heading controller. This reference is updated in each sampling cycle by the track controller.

A block diagram of the ship with the above-mentioned controllers is shown in Fig. 2.6.

Instrumentation. The ship motions and position are measured using dedicated sensors. The ship's heading is measured by some form of gyro compass, distance to a desired track is calculated from a position measurement, with the position measured by a GPS (Global Positioning System) receiver. Two identical gyro compasses are commonly available due to the critical nature of the heading measurement. In the sequel, we will consider the following types of instruments:

- Instrumentation with gyro compass and rate gyro as two separate units. The two measurements are independent.

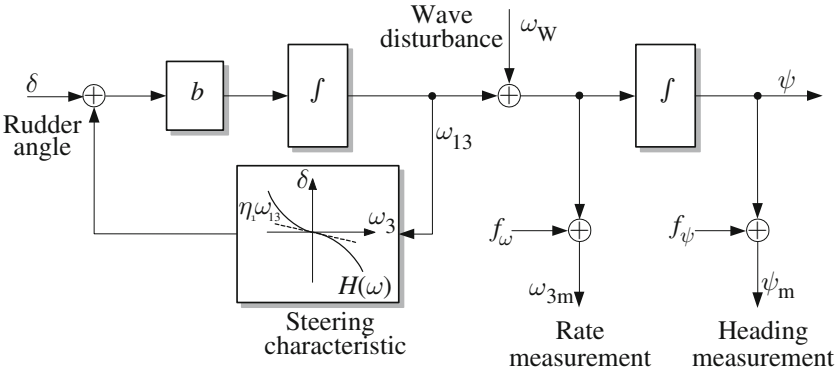


Fig. 2.7 A simple dynamical model of a ship steered by the rudder. Waves act as unknown input and measurement faults are considered on turn rate and heading angle measurements

- Measurement of track error by a navigation computer that measures ship's position using a GPS receiver.

Faults. For the example we consider four possible faults. These faults and the consequences they will have in the example are as follows:

- **Fault in the heading measurement**: In heading control mode, this fault will cause the ship to steer a wrong course. In track control mode, there will be a permanent track error present.
- **Fault in the turn rate measurement**: In heading control mode, this fault will cause a transient error in the heading, but will then be compensated by the controller. A similar behaviour will be seen in track control mode.
- **Fault in the measurement of distance to the desired track**: This has no effect in heading control. In track control mode, there will be an offset equal to the size of the fault.
- **Fault in the track controller**: It causes the heading demand output from this controller to remain at the value it had when the fault occurred. With heading demand being input to the heading controller, this will sooner or later cause the ship to steer away from the desired track.

The sensor faults are modelled as additive faults. The rate gyro measures ω_{3m} and the gyro measures the heading angle ψ_m. This is illustrated in the block diagram in Fig. 2.7.

Dynamics of the ship. On a ship, a desired turn rate is obtained by turning the rudder to a certain angle. The input variable is hence rudder angle and the output is turn rate. Waves act as a disturbance to the turn rate, and the combined signal is integrated to give the actual heading of the ship. This dynamics is illustrated in the block diagram in Fig. 2.7. Turn rate and heading angle are measured variables, the sensors are subject to faults. These are added as fault signals in Fig. 2.7.

The following equations describe the steering problem using the simple model. Waves contribute to turn rate by ω_w. The control input is the rudder angle δ. The measured signals are ψ_m and $\omega_{3\mathrm{m}}$.

In sections dealing with the stochastic case, measurement noise is present on sensor signals. If $\nu_\omega(t)$ and $\nu_\psi(t)$ are noise signals on the turn rate or heading measurements, respectively,

$$\begin{aligned}
\dot{\omega}_3(t) &= b(\delta(t) + H(\omega_3)) \\
\dot{\psi}(t) &= \omega_3(t) + \omega_w(t) \\
\psi_\mathrm{m}(t) &= \psi(t) + f_\psi(t) + \nu_\psi(t) \\
\omega_{3\mathrm{m}}(t) &= \omega_3(t) + \omega_w(t) + f_\omega(t) + \nu_\omega(t)
\end{aligned} \tag{2.16}$$

where $H(\omega)$ is the steady-state relation between turn rate and rudder angle. In the literature, this is the steering characteristic of the ship.

In the example, we treat the steering characteristic as linear such that

$$H(\omega_3) = \eta_1\, \omega_3$$

The sign convention is that angles are taken positive around the third axis, which points downwards as seen from a surface ship. A positive rudder angle (clockwise) will turn the ship counter-clockwise, which corresponds to a negative value of turn rate. Hence, η_1 is negative for a ship that is directionally stable.

In the real world, the relation between a rudder angle and the turn rate is not linear.

$$H(\omega_3) \approx \eta_0 + \eta_1\, \omega_3 + \eta_2\, |\omega_3|\, \omega_3$$

Large tankers or container ships may be directionally unstable in a region around zero turn rate angle. This is a consequence of a balance between hydrodynamical forces on the hull. As turn rate builds up, a directionally unstable ship eventually becomes stable. A directionally unstable ship will enter into a steady turn and move in a circle if the rudder is left in neutral position. A directionally unstable ship will be used to illustrate diagnosis techniques for unstable physical systems.

The variables and parameters in the ship example are listed in Table 2.3.

Heading control. The autopilot to control the ship heading in this example is a linear quadratic design, equivalent to a PD controller without any filtering, signal smoothing or integral action

$$\delta(t) = L_\omega\, \omega_{3\mathrm{m}} + L_\psi\, (\psi_\mathrm{ref} - \psi_\mathrm{m}). \tag{2.17}$$

A block diagram of the autopilot loop is shown in Fig. 2.8.

Track control. Track control means that the ship is commanded to follow a line (great circle) over the sea bottom. The desired track is specified to the controller, and position instruments provide the track error. The control architecture for track control was shown in Fig. 2.6.

Requirements. The requirement to fault-tolerant control for the ship steering example are the following:

Table 2.3 Signals and parameters of the ship steering example

Parameter	Typical value and unit	Meaning
ω_3, ω_w	[deg/s]	Turn rate (angular velocity in yaw)
ψ	[deg]	Ship's heading angle
δ	[deg]	Rudder angle
L_ψ	1 [deg]/[deg]	Gain in heading control
L_ω	2 [deg]/[deg/s]	Rate gain heading control
L_e	1 [deg]/[m]	Gain in track controller
b	2.2 [deg/s^2]/[deg]	Gain factor for ship
η_0	0.0 [deg]	Rudder bias
η_1	-10.0 [deg$\cdot$ s^{-1}]/[deg]	Slope of steering characteristic (Stable ship)
η_2	-20.0 [deg]/[deg^2/s^2]	2nd order parameter in steering characteristic
v_1	10 [m/s]	Ship's forward speed (surge)

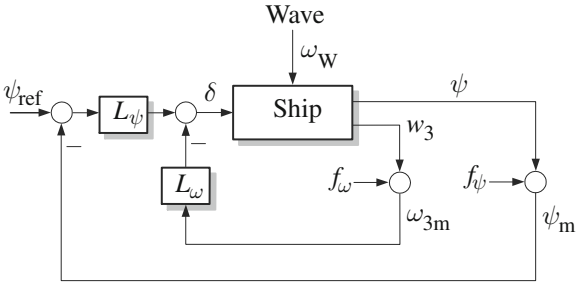


Fig. 2.8 Simple heading controller (autopilot) for the ship example

- An undesired alteration in the ship heading (ψ) must not exceed 5 deg.
- An undesired alteration in the ship turn rate (ω_3) must not exceed 0.2 deg/s
- An undesired alteration in the ship position relative to the track (e) must not exceed 5 m
- An undesired alteration in the ship velocity perpendicular to the track ($\dot{e}$) must not exceed 0.5 m/s

These requirements can be used as objective measures for requirements capture, including detection delay and time to reconfigure.

Part I
Analysis Based on Components and System Structure

Chapter 3
Models of Dynamical Systems

Abstract Dynamical systems can be modelled from different viewpoints. This chapter summarises the main notions. Each of the succeeding chapters uses one of these models for fault diagnosis and fault-tolerant control.

3.1 Fundamental Notions

Fault-tolerant control is based on models. These models have to describe the nominal as well as the faulty system. The following introduces the different models which can be used for fault-tolerant control, starting with the definition of a system as a set of interconnected components, and introducing faults as events which prevent the system components to perform the function they have been designed for.

Dynamical systems. A system is a set of interconnected components. Each of the components has been chosen (or designed) by the system engineer so as to achieve some function of interest. A function describes what the design engineer expects the component to perform, independently of how it is performed. A component performs some function because it has been designed so as to exploit some physical principles, which in general are expressed by some relationships between the time evolution of some system variables. Such relationships are called *constraints*, and the time evolution of a variable is called its *trajectory*.

The components are interconnected by energy or information flows. Energy flows characterise physical systems, which are called "process". Information flows characterise information and control systems.

To illustrate these notions, consider for example a tank. "Storage", which is the *function* classically associated with it, refers to a special operating mode in which the input and the output flows are both equal to zero. In that mode, the mass in the tank stays constant, which indeed justifies the "storage" denomination. However, many different functions could be assigned to a tank, for example, the decoupling (smoothing) of the output flow from some variations of the input flow. This example shows that the notion of function is not univoque, unless the function is understood through the mathematical expression of the constraint that it introduces. In the tank

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8_3

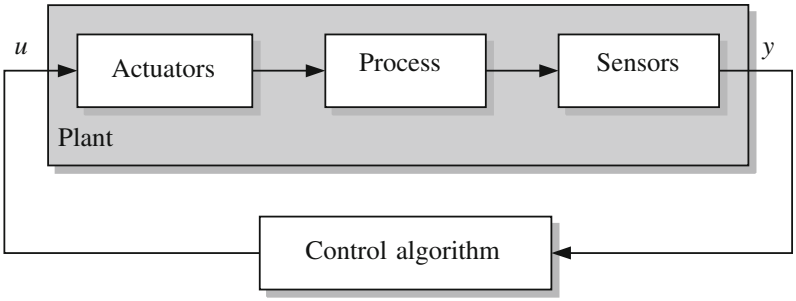


Fig. 3.1 Controlled system

example, the tank function would be the "integration" one, since the associated constraint is

$$\frac{\mathrm{d}A(h)h(t)}{\mathrm{d}t} = q_i(t) - q_o(t),$$

where $A(h)$ is the cross-sectional area, $h(t)$ is the level of the liquid contained in the tank at time t and $q_i(t)$ and $q_o(t)$ are the inflow and outflow at time t.

Controlled systems. Some of the components may have been introduced with the aim of controlling the process, i.e. being able to choose, between all the possible system trajectories, the one which will bring some expected result (Fig. 3.1). Those components which allow to impose the trajectory of a given variable (or to influence the trajectory of a given variable) are called actuators. They establish some constraint between the variables of the process and some control variable, which is called control signal.

For example, the function of an input valve is to control the input flow in some tank. An analog input valve is associated with the constraint

$$q_i(t) = ku(t)\sqrt{\Delta p}(t),$$

where k is some constant parameter, $u(t)$ is the control signal and $\Delta p(t)$ is the differential pressure on both sides of the valve. Note that, as it is seen from the expression of the constraint, the input valve actually controls the input flow only when the differential pressure $\Delta p(t)$ is controlled (or fixed) by another means. In practice, the signal $u(t)$ controls the ratio $\frac{q_i(t)}{\sqrt{\Delta p(t)}}$.

If instead of a continuous valve an on–off valve is used, a different constraint is associated, namely

$$u(t) = 0 \Rightarrow q_i(t) = 0$$
$$u(t) = 1 \Rightarrow q_i(t) = \alpha,$$

where α is a given constant and 0, 1 are two logic values which stand for the control signals of "closed valve" or "open valve", respectively.

Actuators may be driven (i.e. control signals may be generated) by human operators or by control algorithms. In both cases, closed-loop control demands some information about the actual values of some system variables to be known. Sensors are components which are designed so as to provide this information. An example of such a component is a level sensor, whose function is to provide an image of the actual level in the tank. An analog sensor is associated with the constraint

$$y(t) = h(t) + \varepsilon(t),$$

where $y(t)$ is the signal provided by the sensor at time t and $\varepsilon(t)$ is some stochastic process which models the measurement noise, e.g. with normal distribution $N(0, \sigma)$. A discrete sensor is associated with a constraint expressed by a set of rules, an example of which is given in the following table:

$$\begin{aligned} h(t) &\in [0, \alpha[\Rightarrow y(t) = 0 \\ h(t) &\in [\alpha, \beta[\Rightarrow y(t) = a \\ h(t) &\in [\beta, \gamma[\Rightarrow y(t) = b \\ h(t) &\in \geq \gamma \Rightarrow y(t) = c, \end{aligned}$$

where a, b, c and α, β, γ are given constants.

Thus a controlled system is a quadruple:

<process, actuators, sensors, control devices and algorithms>.

Example 3.1 Single-tank system
Consider the following controlled system:
<(tank, output pipe), input valve, level sensor, level controller>.

- Component 1: Tank
  Function: integration
  Constraint: $A\frac{\mathrm{d}h(t)}{\mathrm{d}t} = q_i(t) - q_o(t)$

- Component 2: On–off input valve
  Function: control the input flow
  Constraint: $q_i(t) = \alpha$ if $u(t) = 1$
  $q_i(t) = 0$ if $u(t) = 0$

- Component 3: Output pipe
  Function: deliver the output flow
  Constraint: $q_0(t) = k\sqrt{h(t)}$ where k is some parameter (the output pressure is supposed to be known).

- Component 4: Analog level sensor
  Function: provide an image of the actual level in the tank
  Constraint: $y(t) = h(t) + \varepsilon(t)$, $\varepsilon \sim N(0, \sigma)$
- Component 5: On–off control algorithm
  Function: regulate the level in the tank
  Constraint: if $y(t) \leq h_0 - r$ then $u(t) = 1$,
  if $y(t) \geq h_0 + r$ then $u(t) = 0$, where h_0 and r are given constants. □

Faults. Systems are designed in order to achieve some objectives. Normal operation is an operating mode in which the system objectives are achieved. Normal operation is defined as the simultaneous occurrence of two situations:

1. The components perform properly the functions they have been assigned. This means that they really behave as the designer expected when he designed them, i.e. the constraints they apply to the system variables are the nominal ones.
2. The variables occuring in the component constraints have values in some domains that are compatible with the system objectives.

From this, it follows that two kinds of faults can be distinguished. *Internal faults* change the constraints describing the components. *External faults* are associated with variables whose value does not allow to achieve the system objectives. It can be noticed that internal faults refer to the system state, while external faults refer to the system objectives. Indeed, healthy systems might be unable to achieve the objectives they have been assigned, as the result of inadequate input signals or strong disturbances. On the contrary, faulty systems might still be able to achieve their objective through fault accommodation procedures.

Example 3.2 Internal faults of the tank
Consider the single-tank system. Examples of internal faults are the following:

- Process fault: The tank is leaking.
  Then the description of Component 1 introduced in Example 3.1 is replaced by the following:
  Component 1: Leaking tank
  Constraint: $\frac{\mathrm{d}h(t)}{\mathrm{d}t} = q_i(t) - q_o(t) - q_l(t)$ where $q_l(t)$ is some (unknown) leakage flow.
- Actuator fault: The input valve is blocked open.
  Then Component 2 is described by the following:
  Component 2: Blocked-open input valve
  Constraint: $q_i(t) = \alpha$ whatever the value of $u(t)$.
- Sensor fault: The measurement noise has improper statistical characteristics.
  Then Component 3 is described by the following:
  Component 3: Level sensor
  Constraint: $y(t) = h(t) + \varepsilon(t)$ with $\varepsilon \sim N(0, \Sigma)$ (instead of $N(0, \sigma)$). □

Example 3.3 External fault of the tank
Component 4 defined in Example 3.1 is a control algorithm whose function is to regulate the level in the tank, the objective being to keep $h(t)$ within the interval $(h_0 - r, h_0 + r)$ for any initial value of the level which belongs to this interval. Note that this objective is expressed in terms of two inequality constraints:

$$h(t) \leq h_0 + r$$
$$h(t) \geq h_0 - r.$$

The control signal generated by this algorithm is the input of Component 2 (the actuator) which delivers an input flow α if $y(t) \leq h_0 - r$ holds. It is easily seen that even in the absence of any internal fault, the system objective cannot be achieved for output flows $q_0(t)$ which satisfy, in some time interval (t_1, t_2), the relation

$$\frac{1}{A}\int_{t_1}^{t_2} q_0(t)\mathrm{d}t > h(t_1) + \alpha(t_2 - t_1) - h_0 + r.$$

One can also notice that in the presence of a leakage in the tank (internal fault), the system objective may still be achieved provided that the above inequality does not hold when $q_0(t)$ is replaced by $q_0(t) + q_1(t)$. □

3.2 Modelling the System Architecture

Generic component models describe the system architecture, by describing the system components and their interconnection. For example, sensors, actuators and unitary process devices are elementary components, but higher level ones can be built from their interconnection. A set of interconnected components can be seen, at a higher hierarchical level, as one single complex aggregated component. For example, the aggregation of a tank, an input valve, an output pipe, a level sensor and a regulator (with consistent connection between them) is a high-level component, namely the single-tank system. Thus, components can be considered at any level in the system hierarchical decomposition, and any subsystem (including the whole system itself) can be considered as a component.

Therefore, the system architecture can be described by instantiating a generic component model, at any level of the system hierarchical description. The aim of the generic component model is to provide a common formal modelling frame for every system component, so as to perform systematic manipulations for the purpose of fault diagnosis and fault-tolerant control design. It is not intended to describe the behaviour of the variables which are associated with the component (this is the aim of the behaviour model), but the services that the component provides seen from the user viewpoint. In that context, the user is either another component or the human operator.

Services. A component S is first described by the list of the services that it provides to its users, $S = \{s_i, i \in I_s\}$. A service s_i is a transformation of some *consumed variables* ($cons_i$) into some *produced variables* ($prod_i$), which is performed by the component according to a given procedure ($proc_i$), either in a systematic way, or only upon some specific *request* ($rqst_i$).

For example, a tank consumes input and output mass flows, and produces a stored mass, using an integration procedure (note that the output flow is indeed an input variable for the integration procedure), thus providing an *integration* service (whose behaviour model is $\dot{h}(t) = q_i(t) - q_o(t)$). This transformation does obviously not need to be requested. On the contrary, the *measurement* service of a sensor consumes (an often neglected amount of) energy from the outside world and produces a signal which is the image of the measured variable, by means of the transducer, at each system clock pulse which requests the sensor analog to digital converter operation.

In general, the transformation procedure needs some resources (res_i) to be available, and it may be enabled or disabled at different times ($enable_i$). Therefore, the description of a service is a 6-tuple:

$$s_i = \{cons_i,\ prod_i,\ proc_i,\ rqst_i,\ enable_i,\ res_i\} \tag{3.1}$$

For example, the integration service of the tank is defined by the 6-tuple:

$cons = \{q_i(t), q_o(t)\}$
$prod = \{h(t)\}$
$proc : \dot{h}(t) = q_i(t) - q_o(t)$
$rqst = 1$ (which means that it is always true)
$enable = 1$
$res = \{tank,\ input\ pipe,\ output\ pipe\}$.

Versions. Some components exhibit built-in fault tolerance possibilities, which means that they are still able to provide some services even if the associated resources are faulty and no longer available. This is only possible if there are different means to perform the same transformation, among which at least one does not use the faulty resources. In that case, the service is said to exist under several *versions*, where each version is a 6-tuple like (3.1), which can be used indifferently for the same purpose. It is worth noting that all the versions of the same service share the same request and produce the same output value, but they cannot be simultaneously enabled, and at least one among the input signals, procedures and hardware resources are different from one version to another one. Moreover, since several versions might be able to provide the same result at a given time, there is the need for a mechanism which enables only one of them when the request for the service is issued.

For example, consider a sensor which includes two redundant transducers to measure the same variable. Let

$$\begin{aligned} y_1(t) &= x(t) + \varepsilon_1(t), \quad \varepsilon_1(t) \sim N(0, \sigma_1) \\ y_2(t) &= x(t) + \varepsilon_2(t), \quad \varepsilon_2(t) \sim N(0, \sigma_2) \end{aligned}$$

be the two measurement equations, where $x(t)$ is the unknown variable to be measured, $y_i(t), i = 1, 2$ are, respectively, the two transducers output, and $\varepsilon_i, i = 1, 2$ are the measurement noises, with the two Gaussian distributions $N(0, \sigma_i), i = 1, 2$. The *measurement* service of this sensor could obviously be provided under different versions, namely

Version	Procedure
1	$y(t) \Im \frac{\sigma_2}{\sigma_1 \Downarrow \sigma_2} y_1(t) \Downarrow \frac{\sigma_1}{\sigma_1 \Downarrow \sigma_2} y_2(t)$
2	$y(t) \Im y_1(t)$
3	$y(t) \Im y_2(t)$

where version 1 could be the nominal one, version 2 could be used when transducer 2 is faulty and version 3 would be used when transducer 1 is faulty.

Use-modes. Not all the services provided by a component are enabled at any time. For that reason, subsets of services are gathered into *use-modes*, whose evolution is described by an automaton, which shows the possible transitions from one use-mode to another one, and the conditions under which these transitions are fired.

For example, a typical controller could be described by three use-modes, namely *Off, Initialise, On*, whose content (services) and evolution (automaton) are given in the following table:

Mode	Possible transitions	Enabled services
Off	*To_Initialise, To_On*	
On	*To_Off, To_Initialise*	*Compute_control, Display_set_point*
Initialise	*To_Off, To_On*	*Enter_set_point, Display_set_point*

Building systems from components. As already mentioned, systems (or subsystems) are high-level components, which are built by the aggregation of lower level ones, following a bottom-up approach. Whatever the component level, its generic model includes its use-mode automaton, and the services which are available in each use-mode. Systematic aggregation procedures are defined in order to compose the generic models of low-level components and obtain the generic models of high-level ones.

Fault tolerance analysis. A use-mode is associated with one or several objectives that the component or system must achieve. At any time, the current use-mode defines the current objective, and the enabled services (requests for other services are rejected). The system fault tolerance results from the fact that, in spite of the failure of some resources, the services which are necessary to achieve the objectives of the current use-mode still exist (under at least one version).

3.3 System Behaviour - Basic Modelling Features

Variables. A first question which arises is to select those variables which are of interest to describe the system behaviour. Process components generally introduce power and energy variables, while control systems introduce control and information signals. Therefore, the system variables to be considered are all quantities which are constrained by system components (process, actuators, sensors, control and

estimation algorithms). Note that for systems that obey the Markov property, there is a minimal set of variables which summarise the whole past history of the system until time t (the state variables). The evolution of the state at time t only depends on its value at time t and on the values of the input at time t.

Once the system variables are defined, a second question is to decide about the set of values they can be assigned. Quantitative variables take their values in a subset of the real numbers (which is totally ordered, and provided with the four classical operations), while qualitative variables take their values in a given finite set of symbols, which may be ordered or not. It can be useful to define variables whose values are the segments of some partition of the real line. The coarser the partition, the coarser the granularity of the variable. A symbol is often associated with each segment of the partition, e.g. small, medium, large. Abrupt transitions from one value to another one can be avoided using fuzzy segments instead of crisp ones.

Time. The most classical time variable takes its values in the set of positive real numbers (continuous time). In discrete-time systems, the set of positive integers (or any set isomorphic to that one) is used when sampled-data systems are considered. This time representation is called synchronous since practical sampling systems are driven by a clock. On the contrary, in event-driven systems, time is considered only at each event occurence.

Constraints. The evolution of the system is described by a set of constraints which apply to the system variables. The constraints can be classified according to what they represent and to the form they take.

What constraints represent. In the basic modelling step, each system component is described by its own (local) constraints, and the overall system formed by the interconnection of the components is described by the concatenation of all constraints. In further steps, it may be interesting to solve some constraints and to summarise them within a more compact model.

Example 3.4 Single-tank system
For example, the tank associated with an input pump and an output pipe is a three-component system described by the three local constraints:

$$M_0 : \begin{cases} \text{Tank:} & A\dot{h}(t) = q_i(t) - q_0(t) \\ \text{Pump:} & q_i(t) = \alpha \cdot u(t) \\ \text{Pipe:} & q_0(t) = k\sqrt{h(t)} \end{cases}$$

where $u(t)$ is the control signal, and α and k are two parameters. More condensed models may be created as follows:

$$M_1 : \begin{cases} \text{Tank + pump:} & A\dot{h}(t) = \alpha \cdot .u(t) - q_0(t) \\ \text{Pipe:} & q_0(t) = k\sqrt{h(t)} \end{cases}$$

$$M_2 : \begin{cases} \text{Tank + pipe:} & A\dot{h}(t) = q_i(t) - k\sqrt{h(t)} \\ \text{Pump:} & q_i(t) = \alpha.u(t) \end{cases}$$

$$M_3 : \left\{ \text{Tank + pump + pipe:} \quad A\dot{h}(t) = \alpha.u(t) - k\sqrt{h(t)} \right.$$

Note that the last model uses the minimal number of variables (but it condenses the three components into one single constraint). In fact, $h(t)$ is the system state: the knowledge of $h(t_0)$ and of the input $u(\tau)$, $\tau \in [t_0, t]$ is the only knowledge that is necessary to produce $h(t)$ for any time t. □

The form constraints take. According to the different descriptions of the variables and of time, the constraints have different forms:

- The evolution of continuous variables (whose values are in the set of real numbers) can be described in continuous or in discrete time. Continuous-time descriptions basically use algebraic and differential equations and transfer functions (Laplace transform). Discrete-time descriptions are useful when computer controlled systems are considered, since the data are sampled at a constant rate by the system clock. They basically use algebraic and difference equations, and transfer functions (based on z-transform). Continuous-variable models will be described in Sect. 3.4.
- The evolution of qualitative (or symbolic) variables is best described using discrete-event models such as automata, Petri nets and sets of rules. Such models will be described in Sect. 3.6. Fuzzy variables (and models) can be used when it is wished to avoid abrupt transitions from one qualitative value to another one.
- In many real-life systems, continuous variables and qualitative variables co-exist.

Example 3.5 On/off temperature control system
For example, an on/off temperature control system would be described by the continuous model

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = -a\theta + b$$

when the heater is on, and by the model

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = -a\theta$$

when the heater is off (θ is the temperature to be controlled, and a, b are system parameters). The time evolution of such a system is described not only by the temperature (which is a continuous variable) but also by the heating mode (on/off) which is a qualitative one. Such systems are described by "hybrid models", which will be developed in Sect. 3.7. □

3.4 Continuous-Variable Systems

In continuous-variable systems, the input, state and output variables are defined for a continuum of values in $\mathbb{R}$. Two types of signals can enter such systems: continuous-time or discrete-time signals. For the first, the independent variable t is continuous, and thus such signals are defined over a continuum of time values. Discrete-time signals, on the other hand, are only defined over a time variable k, which belongs to a discrete set. A continuous-time (discrete-time) system processes continuous-time

(discrete-time) input signals and generates a continuous-time (discrete-time) output. Models for these two classes of systems are presented next.

Continuous-time model. A quite general state-space model for a continuous-time continuous-variable nonlinear system can be written as

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{g}(\boldsymbol{x}(t),\, \boldsymbol{u}(t),\, \boldsymbol{d}(t)), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{3.2}$$

$$\boldsymbol{y}(t) = \boldsymbol{h}(\boldsymbol{x}(t),\, \boldsymbol{u}(t),\, \boldsymbol{d}(t)), \tag{3.3}$$

where $\boldsymbol{x} \in |\mathcal{R}^n, \boldsymbol{u} \in |\mathcal{R}^m, \boldsymbol{y} \in |\mathcal{R}^p$ denote the state vector, the vector of known input signals, and the vector of measured output values and the dot the time-derivative:

$$\dot{\boldsymbol{x}}(t) = \frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t}.$$

$\boldsymbol{d} \in |\mathcal{R}^{n_\mathrm{d}}$ stands for the vector of unknown input signals or disturbances acting on the process. The functions $\boldsymbol{g}$ and $\boldsymbol{h}$ are, respectively, $|\mathcal{R}^n$-valued and $|\mathcal{R}^p$ -valued and they are assumed to be smooth. A model of the form (3.2), (3.3) can be obtained using physical laws to describe the considered process.

Example 3.6 Single-tank model
A continuous-valued signal $u(t)$ is considered instead of a binary one like in Example 3.4. Introducing the pipe constraint into the tank model and assuming a noise-free measurement of the level yield

$$A\dot{h}(t) = -k\sqrt{h(t)} + \alpha u(t) \tag{3.4}$$

$$y(t) = h(t), \tag{3.5}$$

which has the form indicated above with $\boldsymbol{d}(t) = \boldsymbol{0}$. □

Linear time-invariant systems can be used to describe the behaviour of a nonlinear system of the form (3.2), (3.3) around a specific set point. Linearisation around a point of operation $\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{d}}$ is obtained by introducing $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \bar{\boldsymbol{x}}, \tilde{\boldsymbol{u}} = \boldsymbol{u} - \bar{\boldsymbol{u}}, \tilde{\boldsymbol{d}} = \boldsymbol{d} - \bar{\boldsymbol{d}}$ and $\tilde{\boldsymbol{y}} = \boldsymbol{y} - \bar{\boldsymbol{y}}$ by performing the Taylor expansion

$$\frac{\mathrm{d}\tilde{\boldsymbol{x}}}{\mathrm{d}t} = \frac{\partial \boldsymbol{g}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{d}})}{\partial \boldsymbol{x}}\tilde{\boldsymbol{x}} + \frac{\partial \boldsymbol{g}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{d}})}{\partial \boldsymbol{u}}\tilde{\boldsymbol{u}} + \frac{\partial \boldsymbol{g}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{d}})}{\partial \boldsymbol{d}}\tilde{\boldsymbol{d}}$$

$$\tilde{\boldsymbol{y}} = \frac{\partial \boldsymbol{h}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{d}})}{\partial \boldsymbol{x}}\tilde{\boldsymbol{x}} + \frac{\partial \boldsymbol{h}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{d}})}{\partial \boldsymbol{u}}\tilde{\boldsymbol{u}} + \frac{\partial \boldsymbol{h}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{d}})}{\partial \boldsymbol{d}}\tilde{\boldsymbol{d}}$$

to obtain a set of linear equations (see Appendix A)

$$\frac{\mathrm{d}\tilde{\boldsymbol{x}}(t)}{\mathrm{d}t} = \boldsymbol{A}_\mathrm{c}\,\tilde{\boldsymbol{x}}(t) + \boldsymbol{B}_\mathrm{c}\,\tilde{\boldsymbol{u}}(t) + \boldsymbol{E}_{\mathrm{x,c}}\,\tilde{\boldsymbol{d}}(t), \quad \tilde{\boldsymbol{x}}(0) = \boldsymbol{x}_0 \tag{3.6}$$

$$\tilde{\boldsymbol{y}}(t) = \boldsymbol{C}_\mathrm{c}\,\tilde{\boldsymbol{x}}(t) + \boldsymbol{D}_\mathrm{c}\,\tilde{\boldsymbol{u}}(t) + \boldsymbol{E}_{\mathrm{y,c}}\,\tilde{\boldsymbol{d}}(t), \tag{3.7}$$

where the variables $\boldsymbol{x}(t)$, $\boldsymbol{u}(t)$, $\boldsymbol{y}(t)$ and $\boldsymbol{d}(t)$ are defined as above, and $\boldsymbol{A}_\mathrm{c}$, $\boldsymbol{B}_\mathrm{c}$, ... are matrices of appropriate dimensions with constant entries in $\mathbb{R}$. In the sequel, $\boldsymbol{x}(t)$ is used instead of $\tilde{\boldsymbol{x}}(t)$ following the tradition of linear systems theory.

Example 3.6 (**cont.**) **Single-tank system**
Let h_0 denote the nominal level around which the tank is normally operated. The nominal control signal u_0 is obtained by looking for the steady-state solution of (3.4) at $h = h_0$, which yields $u_0 = k\sqrt{h_0}/\alpha$. Define $\tilde{h}(t) = h(t) - h_0$ and $\tilde{u}(t) = u(t) - u_0$. A straigthforward computation yields

$$A\frac{\mathrm{d}\tilde{h}(t)}{\mathrm{d}t} = -\beta\tilde{h}(t) + \alpha\tilde{u}(t) \tag{3.8}$$

$$\tilde{y}(t) = \tilde{h}(t), \tag{3.9}$$

where $\beta = k/(2\sqrt{h_0})$. In the sequel, $\tilde{h}(t)$, $\tilde{u}(t)$ and $\tilde{y}(t)$ are replaced by $h(t)$, $u(t)$ and $y(t)$, respectively, keeping in mind that the latter represents discrepancies with respect to their nominal value. □

Discrete-time model. Discrete-time models can be used to model sampled-data systems. All signals are assumed to be sampled synchronously at a fixed sampling period T_s. The traditional theory of sampled-data systems relies on the assumption that the input signals are constant over T_s. This holds true for the control variables, as the output of digital to analog converters has this property. It is, however, an approximation for disturbances and faults.

By integrating the state Eq. (3.6) over one sampling period, the following discrete-time model can be deduced from (3.6), (3.7)

$$\begin{aligned}\boldsymbol{x}(k+1) &= \boldsymbol{A}_\mathrm{d}\boldsymbol{x}(k) + \boldsymbol{B}_\mathrm{d}\boldsymbol{u}(k) + \boldsymbol{E}_{\mathrm{x,d}}\,\boldsymbol{d}(k), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0\\ \boldsymbol{y}(k) &= \boldsymbol{C}_\mathrm{d}\boldsymbol{x}(k) + \boldsymbol{D}_\mathrm{d}\boldsymbol{u}(k) + \boldsymbol{E}_{\mathrm{y,d}}\,\boldsymbol{d}(k),\end{aligned}$$

where k (actually standing for kT_s) denotes the discrete-time instants,

$$\begin{aligned}\boldsymbol{A}_\mathrm{d} &= \exp\left(\boldsymbol{A}_\mathrm{c}T_s\right)\\ \boldsymbol{B}_\mathrm{d} &= \int_0^{T_s} \exp\left(\boldsymbol{A}_\mathrm{c}t\right)\boldsymbol{B}_\mathrm{c}\mathrm{d}t.\end{aligned}$$

$\boldsymbol{E}_{\mathrm{x,d}}$ is defined in a similar way as $\boldsymbol{B}_\mathrm{d}$. Furthermore, the relations $\boldsymbol{C}_\mathrm{d} = \boldsymbol{C}_\mathrm{c}$, $\boldsymbol{D}_\mathrm{d} = \boldsymbol{D}_\mathrm{c}$, $\boldsymbol{E}_{\mathrm{y,d}} = \boldsymbol{E}_{\mathrm{y,c}}$ hold.

Example 3.6 (**cont.**) **Discrete-time model of the single-tank system**
Letting $\phi = \exp(-\beta T_s)$ and

$$\gamma = \int_0^{T_s} \alpha\ \exp(-\beta t)\,\mathrm{d}t = \frac{\alpha}{\beta}(1 - \exp(-\beta T_s)),$$

the discrete-time model deduced from (3.8), (3.9) is written as

$$\begin{aligned} h(k+1) &= \phi h(k) + \gamma u(k) \\ y(k) &= h(k). \ \square \end{aligned}$$

Stochastic disturbances and measurement noise. For discrete-time stochastic models, measurement noise and stochastic disturbances possibly acting on the state variables are described by stochastic sequences (Appendix B). A discrete-time state-space model for the system then takes the form

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}_{\mathrm{d}}\boldsymbol{x}(k) + \boldsymbol{B}_{\mathrm{d}}\boldsymbol{u}(k) + \boldsymbol{E}_{\mathrm{x,d}}\,\boldsymbol{d}(k) + \boldsymbol{w}(k), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ \boldsymbol{y}(k) &= \boldsymbol{C}_{\mathrm{d}}\boldsymbol{x}(k) + \boldsymbol{D}_{\mathrm{d}}\boldsymbol{u}(k) + \boldsymbol{E}_{\mathrm{y,d}}\,\boldsymbol{d}(k) + \boldsymbol{v}(k). \end{aligned}$$

The only new notations are $\boldsymbol{v}(k)$ and $\boldsymbol{w}(k)$. They are samples of white noise sequences with zero mean and covariance matrix

$$E\left\{ \begin{pmatrix} \boldsymbol{w}(k) \\ \boldsymbol{v}(k) \end{pmatrix} \begin{pmatrix} \boldsymbol{w}(\ell)^{\mathrm{T}} & \boldsymbol{v}(\ell)^{\mathrm{T}} \end{pmatrix} \right\} = \begin{pmatrix} \boldsymbol{Q}_{\mathrm{w}} & \boldsymbol{Q}_{\mathrm{wv}} \\ \boldsymbol{Q}_{\mathrm{wv}}^{\mathrm{T}} & \boldsymbol{Q}_{\mathrm{v}} \end{pmatrix} \delta_{k\ell}.$$

Fault model. Fault signals are usually separated into two classes: additive and non-additive (or multiplicative) faults. *Additive faults* appear as additional terms in the state equations of a linear time-invariant system. For stochastic models, they result in changes of the mean value of the measured signals only. *Multiplicative faults* correspond to changes in the parameters of the state equations, namely changes in the entries of the matrices $\boldsymbol{A}_{\mathrm{c}}$, $\boldsymbol{B}_{\mathrm{c}}$, $\boldsymbol{C}_{\mathrm{c}}$, $\boldsymbol{D}_{\mathrm{c}}$ for a continuous-time model (or $\boldsymbol{A}_{\mathrm{d}}$, $\boldsymbol{B}_{\mathrm{d}}$, $\boldsymbol{C}_{\mathrm{d}}$, $\boldsymbol{D}_{\mathrm{d}}$ for a discrete-time model) or changes in the variance of the stochastic disturbance and noise.

A continuous-time linear system subject to additive faults can thus be modelled by

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}_{\mathrm{c}}\,\boldsymbol{x}(t) + \boldsymbol{B}_{\mathrm{c}}\,\boldsymbol{u}(t) + \boldsymbol{E}_{\mathrm{x,c}}\,\boldsymbol{d}(t) + \boldsymbol{F}_{\mathrm{x,c}}\,\boldsymbol{f}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ \boldsymbol{y}(t) &= \boldsymbol{C}_{\mathrm{c}}\,\boldsymbol{x}(t) + \boldsymbol{D}_{\mathrm{c}}\,\boldsymbol{u}(t) + \boldsymbol{E}_{\mathrm{y,c}}\,\boldsymbol{d}(t) + \boldsymbol{F}_{\mathrm{y,c}}\,\boldsymbol{f}(t). \end{aligned}$$

The types of faults that are accounted for in the above model include sensor faults, actuator faults and some component faults.

A continuous-time model subject to non-additive faults can be written as

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}_{\mathrm{c}}\,(\boldsymbol{\theta})\boldsymbol{x}(t) + \boldsymbol{B}_{\mathrm{c}}\,(\boldsymbol{\theta})\boldsymbol{u}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ \boldsymbol{y}(t) &= \boldsymbol{C}_{\mathrm{c}}\,(\boldsymbol{\theta})\boldsymbol{x}(t) + \boldsymbol{D}_{\mathrm{c}}\,(\boldsymbol{\theta})\boldsymbol{u}(t), \end{aligned}$$

where the entries in the different matrices are smooth functions of the parameter vector $\boldsymbol{\theta}$. Under healthy working conditions, the relation $\boldsymbol{\theta} = \boldsymbol{\theta}_0$ holds, while in faulty conditions, the relation $\boldsymbol{\theta} \neq \boldsymbol{\theta}_0$ is valid. An example of multiplicative fault is an abnormal change in the armature resistance of a DC motor.

Example 3.6 (**cont.**) **Single-tank system**
Consider again the continuous-time model (3.8), (3.9) and assume the process can be subject to sensor and actuator faults denoted, respectively, f_s and f_a. Equations (3.8) and (3.9) are modified as follows to account for such faults:

$$\begin{aligned} A\frac{\mathrm{d}h(t)}{\mathrm{d}t} &= -\beta h(t) + \alpha u(t) + \alpha f_a(t) \\ y(t) &= h(t) + f_s(t). \end{aligned}$$

Assume now that a leakage at the bottom of the tank occurs. To account for this phenomenon, (3.4) becomes

$$A\frac{\mathrm{d}h(t)}{\mathrm{d}t} = -k\sqrt{h(t)} - k_{\text{leak}}(t)\sqrt{h(t)} + \alpha u(t). \tag{3.10}$$

If Eq. (3.10) is linearised around the nominal level h_0 and the nominal parameter $k_{\text{leak},0} = 0$, the fault appears to be additive in the linear approximation. Indeed, the latter can be written as

$$A\frac{\mathrm{d}h(t)}{\mathrm{d}t} = -\beta h(t) + \alpha u(t) - \sqrt{h_0}k_{\text{leak}}(t). \square$$

3.5 System Structure

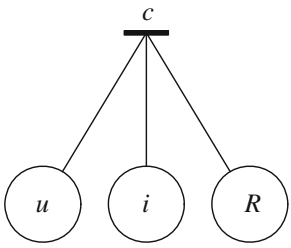
Detailed behavioural models are seldom available in the first phases of system design, and/or are very expensive to develop, especially when complex processes, with hundreds of variables, are considered, and simpler models have to be used. In such situations, structural models provide an interesting approach to the system analysis, since they only need a very primitive level of knowledge about the system behaviour.

Structural model. The structural model of a system is an abstraction of its behavioural model. For continuous-variable systems, the behaviour is described by a set of algebraic and differential equations. Analysing the structure of these equations amounts to analysing the links which exist between variables and parameters, independently on the form of the underlying equations.

For example, consider a system described by Ohm's law

$$u - Ri = 0. \tag{3.11}$$

The structural model associated with this system says: "There exists one constraint (call it c), which links two variables (u, i) and one parameter (R)". It is represented by a bi-partite graph $(\mathcal{C}, \mathcal{Z}, \mathcal{A})$, where $\mathcal{C}$ is the set of constraints, $\mathcal{Z}$ is the set of variables or parameters and $\mathcal{A}$ is the set of edges between $\mathcal{C}$ and $\mathcal{Z}$ or, equivalently, by this graph adjacency matrix, as shown in Fig. 3.2, where bars represent constraints and circles represent variables or parameters.



	u	i	R
c	1	1	1

Fig. 3.2 Structure of Ohm's law

Structural properties. Structural properties of a system are properties of its structure graph. Two systems which have the same structure are said to be structurally equivalent. This is possible, since the structure of a set of constraints is independent of the nature of these constraints, of the variables and of the value of the parameters. Indeed, the structural model would be the same if, instead of Eq. (3.11), Ohm's law was expressed by a look-up table, or if another system, which obeys the numerical model $u(i^2 + 3i + 1) = R$, was considered.

Since structural properties are properties of the structure graph, they are obviously shared by all the systems which have the same structure. Thus, structural properties are properties of a system which are independent of the values of its parameters.

Known and unknown variables. Two kinds of variables appear in the system constraints, namely the known and the unknown ones. Therefore, the set of variables $\mathcal{Z}$ is decomposed into two subsets $\mathcal{Z} = \mathcal{X} \cup \mathcal{K}$. Control and measurement signals are known variables, while the system states are unknown. Known variables obey measurement equations, which are introduced in the structural model. Assuming the voltage u is measured by a sensor whose output is y_1, the previous system obeys the two constraints

$$u - Ri = 0$$
$$y_1 - u = 0$$

and, dropping the parameter R, its structure becomes

$\nearrow$	y_1	u	i
c	1	1	**1**
m_1	1	**1**	

which shows that all the unknown variables in the system can be computed, since u can be computed from y_1 using the measurement equation m_1 (this is symbolised by the bold **1**) and, therefore, i can be computed from y_1 and u. Structural observability is indeed one of the properties that structural analysis allows to study. This is of course only a potential property, since constraints m_1 and c might be more complex

ones, and the numerical computations might be impossible in some particular cases (change, for example, constraint m_1 into $y_1(1-u)=1$ and suppose that the known value of y_1 is zero!).

Faults. When faults occur, the system components do not any longer obey the equations which define their nominal behaviour. Therefore, a given fault mode is described in structural analysis by the subset of constraints which do no longer hold when this fault occurs. These constraints are said to be violated. For example, the short circuit of the previous resistor is described by constraint c being violated when the nominal value of R is used, while a malfunction of the voltage sensor would be described by constraint m_1 being violated.

3.6 Discrete-Event Systems

From a global viewpoint, some dynamical systems can be seen as systems whose signals switch from one value to another one rather than changing their value continuously. In fault diagnosis, systems with discrete measurements occur naturally in the process industry where alarm messages represent discrete information, because the alarm can only be alerted or not and, hence, the corresponding signal is only known to exceed a given threshold or not. As the dynamical behaviour of such systems is described by events denoting the switches of the signal from one discrete value to the next, these systems are called discrete-event systems.

Discrete-valued signals. Due to the symbolic nature of the input, state and output, the symbols v, z and w are used for them. The discrete value sets are enumerated such that

$$\begin{aligned} v &\in \mathcal{V} = \{1, 2, \ldots, M\} \\ z &\in \mathcal{Z} = \{1, 2, \ldots, N\} \\ w &\in \mathcal{W} = \{1, 2, \ldots, R\} \end{aligned}$$

hold (Fig. 3.3). Every change of the symbolic value of v, z or w is called an *event*. For example, if the state jumps from the value j to the value i, a state event denoted by e_{ij} occurs. In Fig. 3.4, the events e_{13} and e_{32} are marked.

The model that will be introduced now describes in which order the events occur but it says nothing about the temporal distance of these events. The sequences of discrete values that the input, state or output assume for a time horizon k_e are denoted by

$$\begin{aligned} V(0 \ldots k_\mathrm{e}) &= (v(0),\ v(1),\ v(2), \ldots,\ v(k_\mathrm{e})) \\ Z(0 \ldots k_\mathrm{e}) &= (z(0),\ z(1),\ z(2), \ldots,\ z(k_\mathrm{e})) \\ W(0 \ldots k_\mathrm{e}) &= (w(0),\ w(1),\ w(2), \ldots,\ w(k_\mathrm{e})). \end{aligned}$$

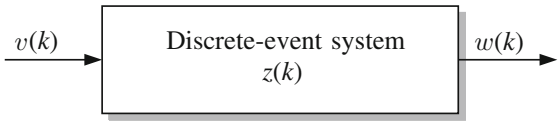


Fig. 3.3 Discrete-event dynamical system

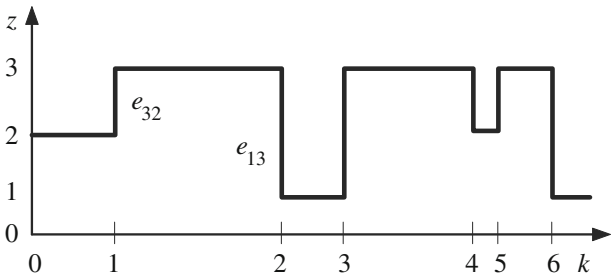


Fig. 3.4 Symbolic signal values and event sequence

In diagnosis, k_e denotes the current time instant and $V(0 \ldots k_e)$ and $W(0 \ldots k_e)$ the measured sequences to be processed.

Deterministic automata. A standard form for describing discrete-event systems is the deterministic automaton

$$\mathcal{A} = (\mathcal{Z},\ \mathcal{V},\ \mathcal{W},\ G,\ H,\ z_0),$$

which has the set of states $\mathcal{Z}$, the input set $\mathcal{V}$ and the output set $\mathcal{W}$. G and H are the state transition function and the output function, which determine the successor state or output in the following way:

$$z(k+1) = G(z(k),\ v(k)), \quad z(0) = z_0 \tag{3.12}$$

$$w(k) = H(z(k),\ v(k)). \tag{3.13}$$

z_0 is the initial state. k denotes the place that the input, state and output values have in the corresponding sequence.

Obviously, for a given initial state z_0 and input sequence $V(0 \ldots k_e)$, the state and output sequences $Z(0 \ldots k_e)$ and $W(0 \ldots k_e)$ can be generated by applying Eqs. (3.12) and (3.13) k_e times. The automaton is deterministic because the initial state and the input sequence unambiguously determine the state and output sequence.

Nondeterministic automaton. In the nondeterministic automaton

$$\mathcal{N}(\mathcal{Z},\ \mathcal{V},\ \mathcal{W},\ L_n,\ z_0),$$

the functions G and H of the deterministic automaton are replaced by the *behavioural relation* L_n

$$L_n : \mathcal{Z} \times \mathcal{W} \times \mathcal{Z} \times \mathcal{V} \in \{0, 1\}$$

which for every given state $z(k)$ and input $v(k)$ describes which successor state $z(k+1)$ can be assumed while generating the output $w(k)$. Hence, the dynamical behaviour of the automaton is described by all 4-tuples for which

$$L_n(z(k+1),\ w(k),\ z(k),\ v(k)) = 1 \tag{3.14}$$

holds. Equation (3.14) replaces Eqs. (3.12), (3.13) of the deterministic automaton. Obviously, for z_0 and $V(0 \ldots k_e)$, the sequences $Z(0 \ldots k_e)$ and $W(0 \ldots k_e)$ are not unique.

If the probabilities with which the automaton assumes the different 4-tuples on the left-hand side of Eq. (3.14) are known, instead of the nondeterministic automaton, a stochastic automaton

$$\mathcal{S} = (\mathcal{Z},\ \mathcal{V},\ \mathcal{W},\ L,\ \text{Prob}(z(0)))$$

can be used to describe the discrete-event system. The *behavioural relation*

$$L : \mathcal{Z} \times \mathcal{W} \times \mathcal{Z} \times \mathcal{V} \longrightarrow [0, 1]$$
$$L(z',\ w,\ z,\ v) = \text{Prob}\,(Z(1) = z',\ W(0) = w \mid Z(0) = z,\ V(0) = v)$$

describes the probability that the automaton steps from state z towards state z' while generating the output w if it gets the input v. Hence, a probability measure can be associated with each state sequence Z and output sequence W.

Model of a faulty discrete-event system. In order to describe the behaviour of a discrete-event system under the influence of faults, the fault $f(k)$ is introduced as an additional discrete-valued input. The fault may change over time and, thus, generate the sequence

$$F(0 \ldots k_e) = (f(0),\ f(1), \ldots,\ f(k_e)).$$

The additional input f extends the stochastic automaton, which becomes

$$\mathcal{S} = (\mathcal{Z},\ \mathcal{V},\ \mathcal{F},\ \mathcal{W},\ L,\ \text{Prob}\,(z_0))$$

with $\mathcal{F}$ denoting the set of possible fault values. The behavioural relation L is now a function of five arguments:

$$L(z',\ w,\ z,\ f,\ v) =$$
$$\text{Prob}\,\big(Z(1) = z',\ W(0) = w | Z(0) = z,\ F(0) = f,\ V(0) = v\big).$$

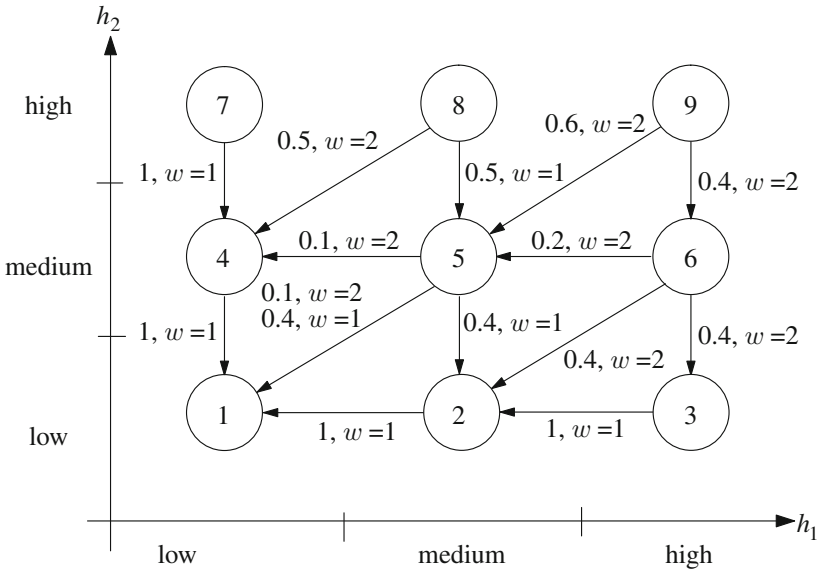


Fig. 3.5 Stochastic automaton describing the tank system for faulty pump ($q_P = 0$)

Example 3.7 Discrete-event model of the two-tank system
The question whether a given system should be dealt with as a continuous-variable or a discrete-event system depends not only on its properties but also on the task to be solved with the model. If for the two-tank system the tank levels should simply remain in a "high" region, it is sufficient to distinguish the levels "high", "medium" and "low" and to describe the behaviour of the system as a switching among these qualitative levels. The graph of the stochastic automaton describing this behaviour is depicted in Fig. 3.5. The automaton state $z = 1$ corresponds to the tank state $(h_1, h_2)^T$, where both tank levels are "low", i.e. do not exceed a given threshold. The other states are defined in a similar way as illustrated by Fig. 3.5. The automaton graph is drawn for faulty pump (no inflow to Tank 1) which makes the input useless. The output $w = 1$ denotes a small and $w = 2$ a large outflow from Tank 2. The labels of the arcs describe the outflow together with the probability with which the state transition described by the arc occurs. The automaton says, for example, that if the tank system is in state 6 (h_1 is high, h_2 medium) then it assumes next one of the states 5, 2 or 3 and that it goes from state 6 towards state 2 while generating the output $w = 2$ with the probability 0.4. All paths through the automaton symbolise a possible state sequence Z and define an associated output sequence W. □

3.7 Hybrid Systems

For many technological systems, both continuous and discrete phenomena play important roles. The mixture of discrete and continuous signals and discrete and continuous forms of the models used is typical for supervisory control tasks and plays a

particular role in diagnosis and fault-tolerant control. As the system possesses both real-valued and discrete-valued signals, combinations of differential equations and automata have to be used for its description (Fig. 3.6).

The main problem in dealing with hybrid systems results from the different ranges of the signals. The transition between these different ranges are represented by quantisers and injectors. The *quantiser* transforms a real-valued signal into a sequence of symbols, where the real-valued signal or signal vector is denoted by a lower-case letter like y or $\boldsymbol{u}$ and the corresponding quantised signals by $[y]$ or $[\boldsymbol{u}]$. If, in the simplest case, the quantiser decides to which real interval of a given set of intervals the current value $y(t)$ belongs, the value of the quantised signal $[y(t)]$ at the same time instant t is the number of the corresponding interval. Clearly, this interval can be associated with symbolic names like "normal", "high" or "low", which give a semantic signal value. As long as the signal does not leave a given interval, the quantised value remains the same. Hence, a continuous change of $y(t)$ is transformed into a sequence of discrete changes of $[y(t)]$, where the quantiser does not only determine the symbolic value of the signal but also the time instants at which these symbolic values change.

The *injector* carries out the inverse mapping. Its input is a symbolic signal like $[\boldsymbol{u}]$, which is associated with a real-valued signal $\boldsymbol{u}$. The relation between $[\boldsymbol{u}]$ and $\boldsymbol{u}$ can be either deterministic where every symbolic value is associated with a unique real value or nondeterministic where the associated real value is randomly selected from a given set of signal values or may vary within this set as long as the symbolic value does not change. In any case, the injector is the interface between symbolic and real-valued signals.

A standard structure of hybrid systems is shown in Fig. 3.6. The system has continuous input and output (u_c and y_c) as well as discrete input and output (u_d and y_d), where the attribute "discrete" refers to the signal value. In addition to that, the system may be considered as a discrete-time system where all signals are known only at given sampling time instants.

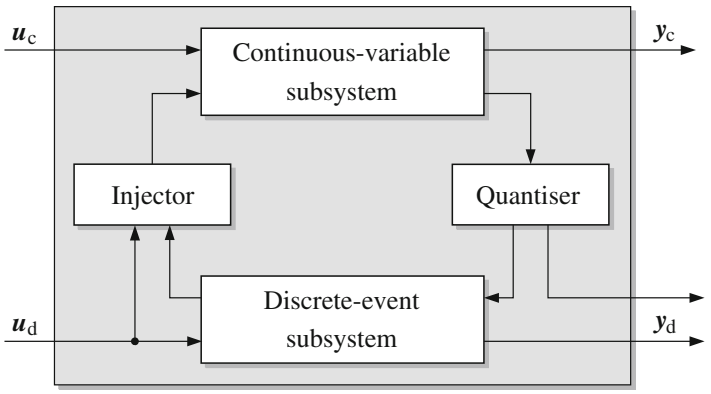


Fig. 3.6 Hybrid dynamical system

3.8 Links Between the Different Models

Since different models can be built in order to describe the same system, there must be some relations between them. The aim of this section is to present and discuss those relations.

Relation among the models. The most important difference between the models introduced so far concerns the value set of the signals. Continuous-variable descriptions refer to signals with real signal values, whereas discrete-event models use signals with discrete-signal values. The question which model is appropriate for a particular application depends upon the question whether the continuous movement of the system or a sequence of discrete events generated by the given system has to be investigated for solving the given task. Therefore, a given system may be considered simultaneously as a continuous system or a discrete system if different problems have to be solved.

For example, a tank system has to be considered as a continuous system if the level of the tank or a concentration of a certain substance in the liquid filling the tank has to be controlled. Level or concentration controllers measure the numerical value of the level or the concentration with a given sampling rate and fix the control input to be applied at the next time instant. The same tank system may be considered as a discrete-event system if it is a part of a batch process. Then a certain recipe is realised by imposing a discrete control sequence on the tank where the control command opens or closes valves to fill or empty the tank, heat or cool the liquid, etc. The controller, which is usually a programmable logic controller, reacts only on events, which are generated if the liquid or the temperature crosses given thresholds. The temporal distance of these events is of minor importance and, therefore, not described by the model.

Architecture and functions. Although functional models have not been developed in this chapter, it may be worth to discuss the link between architecture and function. The architecture model describes the system as a network of interconnected components. The reason why a given component belongs to the system is that it has been chosen to perform a specific function, in a given system operating mode. Thus, each service of a component is associated with a given function the component is expected to fulfil in some operating mode. For example, the "open" service of valve V_a in the tank example is associated with the function "increase the level in Tank T_2" when the level in Tank T_1 is higher than the level in Tank T_2 and higher than the level of the connecting pipe.

Architecture and behaviour. Components provide services which transform consumed variables into produced variables, according to some given procedure. Variables which are processed by services may be quantitative or qualitative. In any case, the procedures which describe the services of a component are nothing else than constraints which link the values of the variables associated with this component. The temporal behaviour of these variables is thus defined once the procedures

are given. Note that these procedures introduce algebraic and differential constraints for quantitative variables and discrete-event models for qualitative variables.

Example 3.8 Different models of the tank system
For example, the "open" service of valve V_{12} considered above introduces an algebraic constraint between the flow from tank T_1 to tank T_2 and the two levels h_1 and h_2

$$\begin{array}{ll} q_{12} = k(u)\sqrt{h_1 - h_2} & \text{if } h_1 \geq \max(h_{12}, h_2) \\ q_{12} = 0 & \text{if } \max(h_1, h_2) \leq h_{12} \\ q_{12} = -k(u)\sqrt{h_1 - h_2} & \text{if } h_2 \geq \max(h_{12}, h_1), \end{array}$$

where $k(u)$ is some coefficient which depends on u, the opening position of the valve, while the "close" service introduces the constraint

$$q_{12} = 0, \quad \forall h_1, h_2.$$

Also note that, since the set of services (i.e. the set of constraints, and therefore the behavioural model) depends on the system operating mode, the generic component model directly introduces a hybrid model for the system description. In the valve example, three operating modes should be considered to describe the behavioural model, namely

$$\begin{array}{l} \text{valve is open and } \max(h_1, h_2) \leq h_{12} \\ \quad \text{then } q_{12} = 0 \\ \text{valve is open and } h_1 \geq \max(h_{12}, h_2) \text{ or } h_2 \geq \max(h_{12}, h_1) \\ \quad \text{then } q_{12} = \text{sign}(h_1 - h_2)k(u)\sqrt{|h_1 - h_2|} \\ \text{valve is closed} \\ \quad \text{then } q_{12} = 0. \end{array}$$

If the functions are considered, two different operating modes have to be associated with the situation $q_{12} \neq 0$, namely

$$\begin{array}{l} \text{valve is open and } h_1 \geq \max(h_{12}, h_2) \\ \quad \text{then } q_{12} = k(u)\sqrt{h_1 - h_2} \text{ and level } h_2 \text{ increases} \\ \text{valve is open and } h_2 \geq \max(h_{12}, h_1) \\ \quad \text{then } q_{12} = -k(u)\sqrt{h_2 - h_1} \text{ and level } h_2 \text{ decreases.} \end{array}$$

It can be checked that a discrete-event model of this system can be built by considering the following events:

e_1: valve V_{12} opens
e_2: valve V_{12} closes
e_3: both levels h_1 and h_2 become lower than h_{12}
e_4: h_1 becomes higher than $\max(h_{12}, h_2)$
e_5: h_2 becomes higher than $\max(h_{12}, h_1)$. □

Behaviour and structure. The link between the behavioural model and the structural model is obvious, since the structural model is nothing but an abstraction of the

behavioural model. In each operating mode, there is a set of constraints $\mathcal{C}$ which link the values of the system variables $\mathcal{Z} = \mathcal{X} \cup \mathcal{K}$. The structure of these constraints is directly represented by the set of edges $\mathcal{A}$ in the bi-partite graph $(\mathcal{C}, \mathcal{Z}, \mathcal{A})$ whose nodes are, respectively, $\mathcal{C}$ and $\mathcal{Z}$.

Example 3.9 Structure of a valve in different operation modes
In the valve example, there are two different structures associated with the four different operating modes which appear on the hybrid description of the system behaviour:

- **Structure 1**: If the valve is open and $\max(h_1, h_2) \leq h_{12}$ or if the valve is closed, the following relation holds:

$\nearrow$	h_1	h_2	q_{12}	u
c_1				1
c_2			1	

  Constraint c_1 expresses that the control u is known, and constraint c_2 expresses that the flow q_{12} is also known (since $q_{12} = 0$).

- **Structure 2**: If the valve is open and $h_1 \geq \max(h_{12}, h_2)$ or $h_2 \geq \max(h_{12}, h_1)$,

$\nearrow$	h_1	h_2	q_{12}	u
c_1				1
c_2	1	1	1	1

  where constraint c_1 expresses that the control u is known, and constraint c_2 expresses the relation between the flow q_{12}, the control u and the two levels $h_1$and h_2. □

3.9 Exercises

Exercise 3.1 Model of ship dynamics
Using the notation from Sect. 2.3, the dynamical model of a ship is

$$\begin{aligned}
\dot{\omega}_3 &= b(\eta_1\omega_3 + \eta_3\omega_3^3) + b\delta \\
\dot{\psi} &= \omega_3 + \omega_w \\
y_1 &= \psi \\
y_2 &= \dot{\psi} \\
y_3 &= \delta.
\end{aligned}$$

1. Derive a linear model in state-space form, linearising about the point of operation

$$\bar{\omega}_3 = \omega_o, \quad \bar{\psi} = \psi_o, \quad \bar{\delta} = \delta_o.$$

2. Find also the model for the special case

$$\bar{\omega}_3 = 0, \quad \bar{\psi} = 0, \quad \bar{\delta} = 0. \square$$

Exercise 3.2 Model of industrial actuator
A block diagram of an industrial actuator is shown in Fig. 3.7. It consists of the following components:

- DC motor with input current i and motor speed n,
- power drive with known current command i_{com},
- gear with gear ratio N efficiency η and output angle θ, and
- unknown load torque Q_1.

Measurements are θ_m the angle after the gear and n_m the shaft speed at the motor.
The faults concern

- f_θ - position sensor fault
- f_n - tachometer fault
- f_i - actuator fault.

With

$$\begin{aligned} \boldsymbol{x}(t) &= (n(t), \theta(t))^{\mathrm{T}} \\ u(t) &= i_{\text{com}}(t) \\ d &= Q_l(t) \\ \boldsymbol{f}(t) &= (f_i(t), f_n(t), f_\theta(t))^{\mathrm{T}} \\ \boldsymbol{y}(t) &= (n_{\text{m}}(t), \theta_{\text{m}}(t))^{\mathrm{T}}, \end{aligned}$$

the actuator has the following state-space representation:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}u(t) + \boldsymbol{E}_{\text{x}}d(t) + \boldsymbol{F}_{\text{x}}\boldsymbol{f}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{F}_y\boldsymbol{f}(t). \end{aligned} \tag{3.15}$$

1. Show that

$$\boldsymbol{A} = \begin{pmatrix} -\frac{\alpha}{I_{\text{tot}}} & 0 \\ \frac{1}{N} & 0 \end{pmatrix}$$

and determine the remaining matrices in the state-space model.

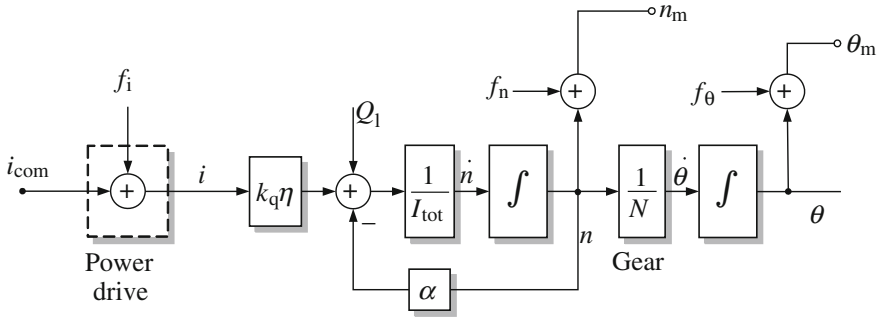


Fig. 3.7 Block diagram of actuator with additive faults-open loop

2. Show that the system transfer function (Laplace domain) is

$$\begin{aligned}\boldsymbol{x}(s) &= (s\boldsymbol{I} - \boldsymbol{A})^{-1}(\boldsymbol{B}u(s) + \boldsymbol{E}_{\mathrm{x}}d(s) + \boldsymbol{F}_{\mathrm{x}}\boldsymbol{f}(s))\\ \boldsymbol{y}(s) &= \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}(\boldsymbol{B}u(s) + \boldsymbol{E}_{\mathrm{x}}d(s) + \boldsymbol{F}_{\mathrm{x}}\boldsymbol{f}(s)) + \boldsymbol{F}_{y}\boldsymbol{f}(s).\end{aligned}$$

3. Using the shorthand notation

$$\boldsymbol{y}(s) = \boldsymbol{H}_{\mathrm{yu}}(s)u(s) + \boldsymbol{H}_{\mathrm{yd}}(s)d(s) + \boldsymbol{H}_{\mathrm{yf}}\boldsymbol{f}(s),$$

determine the three transfer function matrices $\boldsymbol{H}_{\mathrm{yu}}$, $\boldsymbol{H}_{\mathrm{yd}}$ and $\boldsymbol{H}_{\mathrm{yf}}$. Verify what is apparent from the block diagram,

$$n_m(s) = \frac{1}{sI_{\mathrm{tot}} + \alpha}(k_q\eta i_{\mathrm{com}}(s) + Q_{\mathrm{l}}(s) + k_q\eta f_i(s)) + f_n(s). \square$$

Exercise 3.3 Discrete-time model of industrial actuator
The following parameters apply to the industrial actuator described in Exercise 3.2: $k_q = 0.5$ Nm/A, $\eta = 0.8$, $N = 100$, $I_{\mathrm{tot}} = 2 \cdot 10^{-3}$ kgm^2, $\alpha = 10^{-4}$ Nms/rad.

1. Determine the numerical transfer function matrices $\boldsymbol{H}_{\mathrm{yu}}$, $\boldsymbol{H}_{\mathrm{yd}}$, $\boldsymbol{H}_{\mathrm{yf}}$. Find the values of gains and the location of poles and zeros;
2. Make a discrete-time model using a sampling time of 2 ms. Note values of gains, discrete-time (z-plane) poles and zeros in the discrete-time model; and
3. Determine the steady-state properties of the change in measurement values when step-wise faults and disturbance are applied. Faults or load steps appear one at a time and are not simultaneously present. □

Exercise 3.4 Industrial actuator with speed control
Figure 3.8 shows an actuator with speed control, a limit in the maximum current from the power drive and measurement of motor current i_{m}. The speed controller is $i_{\mathrm{com}} = k_t(n_{\mathrm{ref}} - n_m)$. The power drive has a gain of 1 in the linear range $|i| \leq i_{\max}$, otherwise $i = i_{\max}$ sign (i_{com}).

1. Write a dynamical model of the actuator in the form

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} n(t) \\ \theta(t)\end{pmatrix} &= \boldsymbol{A}_{\mathrm{cl}}\begin{pmatrix} n(t) \\ \theta(t)\end{pmatrix} + \boldsymbol{B}_{\mathrm{cl}}n_{\mathrm{ref}}(t) + \boldsymbol{E}_{x,cl}\,Q_{\mathrm{l}}(t) + \boldsymbol{F}_{x,cl}\,\boldsymbol{f}(t)\\ \begin{pmatrix} n_{\mathrm{m}}(t) \\ \theta_{\mathrm{m}}(t) \\ i_{\mathrm{m}}(t)\end{pmatrix} &= \boldsymbol{C}_{\mathrm{cl}}\begin{pmatrix} n(t) \\ \theta(t)\end{pmatrix} + \boldsymbol{D}_{\mathrm{cl}}n_{\mathrm{ref}}(t) + \boldsymbol{E}_{y,cl}\,Q_{\mathrm{l}}(t) + \boldsymbol{F}_{y,cl}\,\boldsymbol{f}(t)\end{aligned}$$

in the linear operating range and determine the elements of all parameter matrices $\boldsymbol{A}_{\mathrm{cl}}$, $\boldsymbol{B}_{\mathrm{cl}}, \ldots$ in the model;
2. Implement a simulation of the continuous-time model of the actuator of Fig. 3.8. Use $k_t = 1.0$ As/rad and $i_{\max} = \pm 20$ A in the current limiter block in the simulation; and
3. Validate the responses to step-wise changes in reference, load torque and fault signals and compare with those of the theoretical model. □

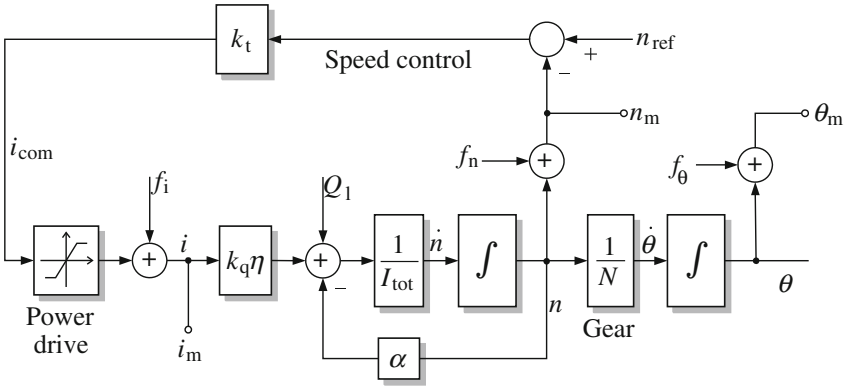


Fig. 3.8 Actuator with angular velocity feedback

Exercise 3.5 Model of a coffee machine
Describe the steps to produce a coffee with milk by means of a coffee machine by a deterministic automaton. How can this automaton be extended to hybrid model if the differential equations describing the continuous processes are associated to some of the automaton states? □

3.10 Bibliographical Notes

Modelling the system architecture. Modular- and object-oriented models have been developed to describe architectures of automated systems [4, 167, 224]. Such models are used in the description and the validation of real-time and distributed systems [363], and specific tools, like state charts, have been developed to describe their real-time operation [143]. Generic component models are architecture models, first developed for the description of intelligent sensors and actuators [323]. They have been used for the interoperability analysis of distributed architectures [3, 48]. There exists a bridge between SyncCharts and generic component models, which provides a means of analysing both the system architecture and its associated real-time behaviour [15].

For an introduction into the wide field of system identification, the reader is referred to the monographs [201, 318, 377].

Good introductions to discrete-event systems are [59, 211].

Chapter 4
Analysis Based on Components and Architecture

Abstract This chapter presents methods for modelling and analysing the component architecture of a system. It deals with the information that can be deduced from components and the way in which the components are connected. Simple and aggregated components are described in terms of their generic properties, which include the service offered by a component in different modes of operations and the conditions under which component faults occur. Properties of selected simple components are discussed and their aggregation into generic components at a higher level is illustrated. Formal methods for describing generic components are introduced. Algebraic and graph-theoretic methods are employed to analyse the propagation of faults through the faulty system.

4.1 Introduction

Architecture models describe a system as a set of interconnected components. This statement is true at any hierarchical level. Low-level components, such as sensors and actuators, are directly interfaced with the process. They provide low-level services: measurement of, or action on some specific process variable. Subsystems, composed of several components, can also be considered. They form higher-level devices which can be aggregated to even higher levels. Higher-level devices provide higher-level services. The primary track control loop in the ship example provides the ship to follow a desired path through shallow water; the cooling unit in a chemical reactor provides control of an exothermic reaction. The highest aggregation level is that of the system itself, when it is considered as one single component.

At any considered level, a component, whether simple or aggregated, can be described by its generic model. The services it provides can be organised in a number of use-modes. At the highest aggregation level, each of the system use-modes is associated with a given number of objectives to perform, and the system services are used to achieve those system objectives. The definition of the use-modes set, and for each use-mode the associated objectives result directly from the specification of the considered system.

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8_4

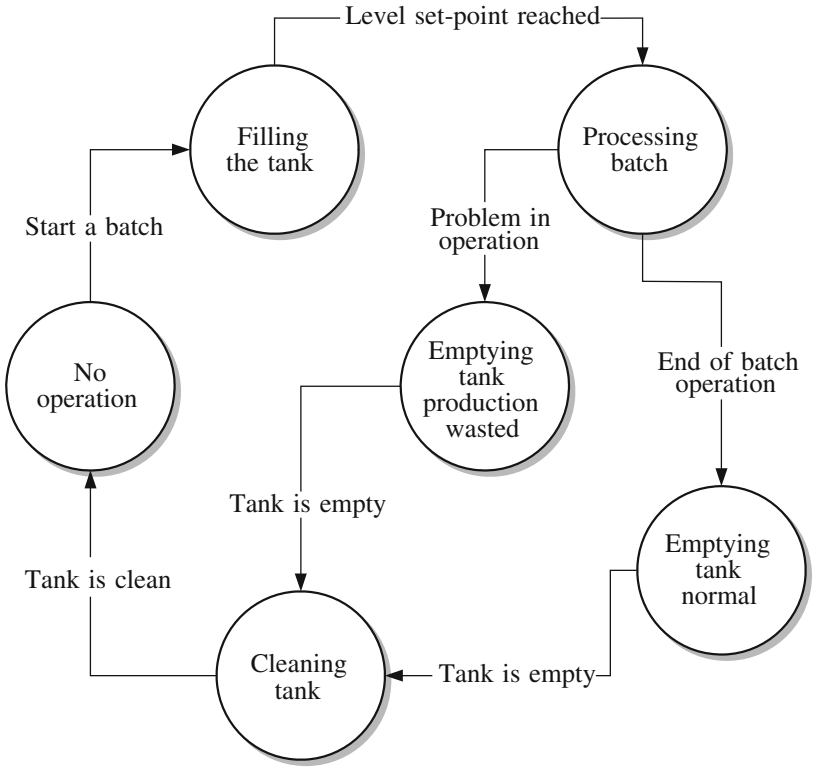


Fig. 4.1 Automaton of a batch process illustrated through use-modes

Example 4.1 Tank system
Suppose that the tank system is used in a food industry batch production process, where the processing of each batch needs the temperature to be controlled at a given value during a given period of time. Six different use-modes (UM) can be distinguished:

UM 0: No operation,
UM 1: Filling the tank,
UM 2: Processing the batch,
UM 3: Emptying the tank via the normal pipe,
UM 4: Emptying the tank via the "lost production" pipe, and
UM 5: Cleaning the tank.

The associated use-mode automaton is illustrated in Fig. 4.1.
There are six objectives associated with the different use-modes, namely

- Objective 0: No action (UM 0),
- Objective 1: Reach the full level set-point (UM 1),
- Objective 2: Regulate the temperature (UM 2),
- Objective 3: Reach the empty level set-point (UM 3 and UM 4),
- Objective 4: Clean the tank (UM 5), and
- Objective 5: Preserve the environment (UM 1, 2, 3, 4 and 5). □

The analysis of fault tolerance should answer the essential question whether a given system, in a given fault situation, is still able to achieve its objective. Overall objectives are associated with use-modes, and they are achieved using the services offered at the system level. Thus, the analysis of fault tolerance first needs the generic model to be derived at the system level. This can be done by defining procedures which aggregate low-level generic models into higher-level ones. The second step is to analyse the situation when faults appear, in order to conclude about the way services are affected.

This chapter presents two approaches for the analysis of fault tolerance using architecture models, the first studying fault propagation mechanisms and the second analysing the availability of services (which means the possibility of achieving the objectives) at the system level.

4.2 Faults in Components and Their Consequences

Having defined availability of services and the key concept in the generic description of components, tools are needed to analyse which conditions could cause a certain version of a service to become unavailable. Faults or partial failure in components would clearly be candidates for a service to become unavailable. This section introduces a Boolean formalism to analyse propagation of faults and the consequences faults can have on the services offered by a generic component.

Shut-down functions and interlocks are commonly used in industrial automation to prevent failures to dilate from one sub-system to another. The use of such functions has, however, the consequence that plant availability is sometimes reduced without good reason. With the ever higher degree of automation, this has been the key cause to increased vulnerability to simple faults, particularly in sensors and actuators. The approach in this text is to obtain dependability by giving a generic component or subsystem an ability to detect and isolate faults and react with actions that accommodate the control system to the fault. Fault accommodation is predetermined at the design stage. The scope of the methods presented in this section is to give a formal technique to obtain a list of which faults should be handled to regain an acceptable version services after faults have occurred locally in a generic component.

Open and closed-loop systems. Handling of faults in open-loop systems, e.g., monitoring and remote control, is technically straight-forward, but the reactions used to accommodate a fault need to be designed with careful consideration to safety and availability of the total plant. Optimisation at a local level may easily violate an overall safety goal.

Handling of faults in closed-loop components is a more difficult and challenging task. Properly designed systems can accommodate the effects of faults whereas less careful designs can let fault effects propagate to other subsystems.

Connection with reliability analysis. For the reasons given above, fault analysis need to incorporate analysis throughout a system. Traditional methods for fault

detection and isolation do not cover this problem. They are very able to detect the presence of a fault as a difference between actual and expected behaviour. Isolation of a particular fault requires a hypothesis about the observed effects from this fault. This is obtained by ad hoc engineering and requires deep process knowledge and engineering skills to make a successful design. It is expensive in terms of both key personnel and time.

Analysis of system reliability is not only mandatory for safety critical systems but is also more and more often used for common industrial systems, driven by the increasing environment and safety awareness in recent years. The state of the art is such that no method can guarantee a complete description of all possible fault modes of a system. Certain forms of risk analysis provide, nevertheless, a very systematic approach to fault modelling once possible component faults have been identified. Faults in common industrial components are subject to constant study, and a methodology based on component fault modelling could use accumulated knowledge for each type of component. The number of principally different components in a certain branch of industry is small enough to make this a manageable exercise.

A systematic approach. A systematic approach can be made if the basic methodology from risk analysis is adopted to the detailed mathematical models needed for real-time fault diagnosis. A link has to be established from quantitative, static risk models at the component level to qualitative, dynamical fault diagnosis descriptions of input–output relations to achieve this goal.

The link is obviously to merge the component-based generic dynamical models (energy, momentum and flow relations) with component fault models from the risk analysis. The generic dynamical models can be extended to subsystem input–output descriptions, for example using a system behaviour description approach. This methodology guarantees that all relevant component faults are included in a mathematical system model, and all relevant dynamical relations are preserved due to modelling being done at the component level.

The systematic approach shall provide the following information:

1. List of faults to detect,
2. Mathematical model for use in fault diagnosis,
3. Basic character/criticality of each fault, and
4. Required reaction to each fault.

This is elaborated in the following.

4.3 Fault Propagation Analysis

Several approaches exist to analyse systems based on the components they comprise. The fields of risk analysis and reliability engineering have developed several approaches to assess the risks associated with component breakdown. On commonly accepted standard in everyday industrial use is the failure modes and effects analysis (FMEA) technique. It is based on description of the failure modes of the individual

components, and would thus serve our purpose. The failure mode and effects analysis technique is well established and supported by both databases with breakdown information and mean-time between failure history for many components. It was hence natural to develop a method for analysis of fault propagation based on such available information on component failure modes.

Failure modes and effects analysis. Failure modes and effects analysis is a tool originally developed by reliability engineers. It analyses potential effects caused by simple or aggregated components ceasing to behave as intended, i.e. they stop providing the service designated to the component. A failure modes and effects analysis procedure starts with listing, for each component, in which ways can this component fail. This is referred to as failure modes. Databases are available with information about failure modes for a large number of industrial components. The output of a failure modes and effects analysis procedure is the effect on the system and its environment that would be the consequence if the particular component should fail in each of the ways available to it (failure effects).

An example for a typical failure modes and effects analysis worksheet is illustrated for a pressure gauge in the following table.

item ident.	*failure mode*	*failure cause*	*failure effect*	*risk assess. sev prob*
press. gauge PG 24	false high reading	defective stuck	toxins not destroyed	4 0.0002
	false low reading	defective stuck	potential burns	3 0.0002

The failure modes and effects analysis worksheet has columns for *item identification*, *failure modes, failure cause, failure effect* and *risk assessment* for the end effects at system/environment level. There are also columns for *risk code* and *actions required*, not shown here.

The information on end effects in a failure modes and effect analysis scheme is firmly linked to the system architecture, which is not explicit in the worksheet. Analysis and design of fault-tolerant systems require a fully flexible representation not offered by the failure modes and effect analysis scheme itself, but the information on component failure modes is very useful and is exploited in the following.

Fault propagation matrix. A traditional failure mode and effects analysis starts with selection of the lowest level of analysis. In the present context, this means sensors, valves, motors and similar components. All potential faults and their effects are determined. A fault propagation scheme for each component shows how fault effects out of the component relate to faults at input, output, or parts within the components. This is illustrated in Fig. 4.2.

Analysis of the propagation of faults is conveniently based on matrix methods. The *fault propagation analysis* (FPA) uses a Boolean mapping of faults onto effects for each component or each set of aggregated components.

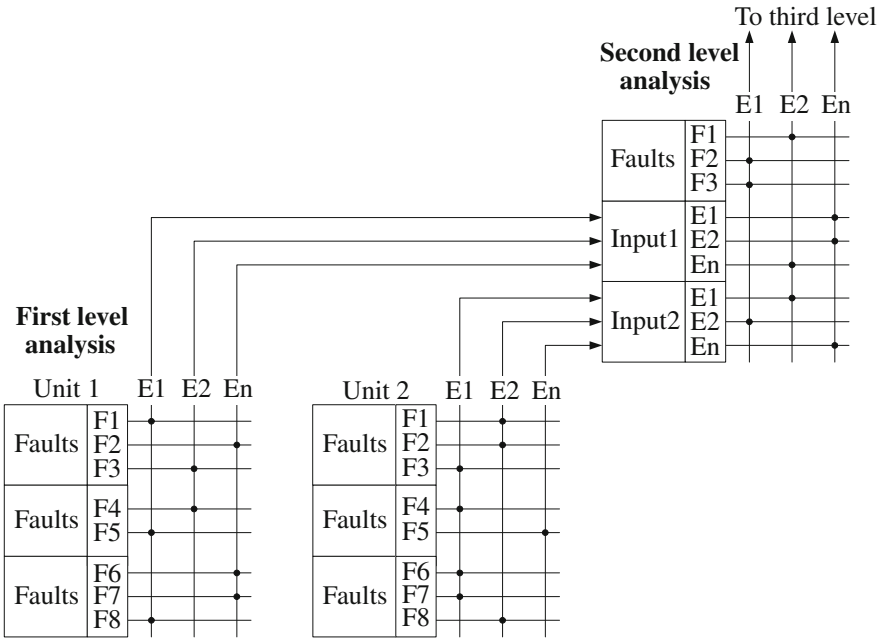


Fig. 4.2 Traditional failure modes and effects analysis scheme illustrated graphically for two component levels

Definition 4.1 (*Fault propagation matrix*) For a given Boolean mapping $\boldsymbol{M}$

$$\boldsymbol{M}: \ \mathcal{F} \times \mathcal{E} \ \rightarrow \ \{0, 1\}$$

of the set of component faults $f_c \in \mathcal{F}$ onto the set of effects $e_c \in \mathcal{E}$, the fault propagation matrix is defined as follows:

$$m_{ij} = \begin{cases} 1 & \text{if} \quad f_{cj} = 1 \Rightarrow e_{ci} = 1 \\ 0 & \text{otherwise.} \end{cases}$$

A fault propagation matrix scheme can be expressed as

$$\boldsymbol{e}_{ci} \leftarrow \boldsymbol{M}_i^f \otimes \boldsymbol{f}_{ci},$$

where $\boldsymbol{M}_i^f$ is a Boolean matrix representing the propagation. The operator $\otimes$ is the inner product disjunction operator that performs the Boolean operation

$$e_{cik} \leftarrow (m_{ik1} \wedge f_{ci1}) \vee (m_{ik2} \wedge f_{ci2}) \ldots \vee (m_{ikn} \wedge f_{cin}).$$

When effects propagate from other components, we get, at level i:

$$\boldsymbol{e}_{ci} \leftarrow \boldsymbol{M}_i^f \otimes \begin{pmatrix} \boldsymbol{f}_{ci} \\ \boldsymbol{e}_{c\,(i-1)} \end{pmatrix}.$$

This is a surjective mapping from faults to effects: there is a unique path from fault to end effect, but several different faults may cause the same end effect.

System descriptions are obtained from interconnection of component descriptions. Merging three levels gives the end effects at the second level,

$$\boldsymbol{e}_{c2} \leftarrow \left(\boldsymbol{M}_2^f \otimes \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ 0 & \boldsymbol{M}_1^f \end{pmatrix} \right) \otimes \begin{pmatrix} \boldsymbol{f}_{c2} \\ \boldsymbol{f}_{c1} \end{pmatrix}.$$

Eventually, end effects at the system level are reached.

Reverse analysis. The effect vector corresponding to a particular fault f_k is hence the kth column of $\boldsymbol{M}^f$. Reversely, given a particular $\boldsymbol{e}$, the set of faults that could cause this effect is obtained by checking which columns of $\boldsymbol{M}^f$ match the observed $\boldsymbol{e}$. This can be written using the operator $\odot$ defined by the operation

$$\begin{aligned} f_i &\leftarrow (m_{i1} = e_1) \wedge (m_{i2} = e_2) \ldots \wedge (m_{in} = e_n) \\ i &= 1, \ldots, \dim f \end{aligned}$$

and apply this on $\left(\boldsymbol{M}^f\right)^{\mathrm{T}}$,

$$\boldsymbol{f}_{\mathrm{c}} = \left(\boldsymbol{M}^f\right)^{\mathrm{T}} \odot \boldsymbol{e}_{\mathrm{c}}.$$

Analysis of the system matrix can easily show where in the system the propagation should be detected and stopped, the operation we would achieve by fault handling. Proper handling of a fault would imply the particular entry(ies) in the $\boldsymbol{M}^f$ matrix change from "1" to "0".

Experience from applying fault propagation analysis to larger systems shows that we might need to include occurrence of one fault and the non-occurrence of another in the description. This would imply to extend $\boldsymbol{f}_i$ to $\left[\boldsymbol{f}_i, \bar{\boldsymbol{f}}_j\right]^{\mathrm{T}}$ in the above expressions.

Analysis of a system with three simple components and a description of their architecture is shown in the following example.

Example 4.2 Propagation with three components
A system with three components and open-loop structure is

$$\begin{aligned} e_{c3} &\leftarrow \boldsymbol{M}_3^f \otimes \begin{pmatrix} f_{c3} \\ e_{c2} \end{pmatrix} \\ e_{c2} &\leftarrow \boldsymbol{M}_2^f \otimes \begin{pmatrix} f_{c2} \\ e_{c1} \end{pmatrix} \\ e_{c1} &\leftarrow \boldsymbol{M}_1^f \otimes (f_{c1}) \end{aligned}$$

The fault effect scheme for this example is

$$
\begin{aligned}
e_{c3} &\leftarrow \boldsymbol{M}_3^f \otimes \begin{pmatrix} f_{c3} \\ e_{c2} \end{pmatrix} \Rightarrow \\
e_{c3} &\leftarrow \left(\boldsymbol{M}_3^f \otimes \begin{pmatrix} \boldsymbol{I} & 0 \\ 0 & \boldsymbol{M}_2^f \end{pmatrix} \right) \otimes \begin{pmatrix} f_{c3} \\ f_{c2} \\ e_{c1} \end{pmatrix} \Rightarrow \\
e_{c3} &\leftarrow \left(\boldsymbol{M}_3^f \otimes \begin{pmatrix} \boldsymbol{I} & 0 \\ 0 & \boldsymbol{M}_2^f \otimes \begin{pmatrix} I & 0 \\ 0 & \boldsymbol{M}_1^f \end{pmatrix} \end{pmatrix} \right) \otimes \begin{pmatrix} f_{c3} \\ f_{c2} \\ f_{c1} \end{pmatrix} \Rightarrow \\
e_{c3} &\leftarrow \boldsymbol{M}_3^f \otimes \begin{pmatrix} \boldsymbol{I} & 0 \\ 0 & \boldsymbol{M}_2^f \end{pmatrix} \otimes \begin{pmatrix} I & 0 \\ 0 & \boldsymbol{M}_1^f \end{pmatrix} \otimes \begin{pmatrix} f_{c3} \\ f_{c2} \\ f_{c1} \end{pmatrix} \equiv \boldsymbol{M}_{sys}^f \otimes f_{sys}
\end{aligned}
$$

Effects are seen to be propagated to the next level of analysis and act as part's faults at that level. This is continued until the system level is reached. The schemes give an surjective mapping from faults to effects: There is a unique path from fault to end effect, but different faults may cause the same end effect. □

It is noted that the Boolean propagation matrix can be split into columns propagating faults and columns propagating input effects,

$$\boldsymbol{M}_i^f = \left(\boldsymbol{M}_{i,f}^f \,|\boldsymbol{M}_{i,e}^f \right)$$

Merging two levels can then be re-written

$$
\begin{aligned}
e_{c2} &\leftarrow \left(\boldsymbol{M}_2^f \otimes \begin{pmatrix} \boldsymbol{I} & 0 \\ 0 & \boldsymbol{M}_1^f \end{pmatrix} \right) \otimes \begin{pmatrix} f_{c2} \\ e_{c1} \end{pmatrix} \Rightarrow \\
e_{c2} &\leftarrow \left(\boldsymbol{M}_{2,f}^f \quad \boldsymbol{M}_{2,e}^f \otimes \boldsymbol{M}_1^f \right) \otimes \begin{pmatrix} f_{c2} \\ f_{c1} \end{pmatrix}
\end{aligned}
$$

This illustrates how faults from the current level are propagated through the component being considered, while faults from a lower level propagate through this lower level and through the present.

Remark 4.1 (*Single-fault assumption*) This discussion of fault propagation is based on the assumption that only a single fault is present. If the analysis should cover the occurrence of multiple faults, or propagation of one particular fault being dependent on a particular other fault not being present, we would need a more complex logic description than introduced above. Results do exist, but are considered outside the scope of the present text. □

Example 4.3 Autopilot - gyro system diagnosis
Failure of ship's motion control systems have been the cause of many severe accidents. Some of these were caused by faults in the gyro-system providing the heading and turn rate motion feedback to the autopilot. Early detection of such faults could prevent the control system from unwanted alteration of the ship's heading. This example illustrates fault detection on a ship's gyro compass and an associated turn rate sensor.

Faults are possible in either of the two measurements and the purpose of the fault detection is to isolate the faulty sensor. Subsequent fault accommodation should then switch the faulty sensor out and estimate the missing signal from that of the good sensor. Fault-tolerance against these faults is obtained by implementing the scheme as an autonomous part of the heading control loop.

FPA scheme for rate gyro. FPA schemes describe the properties of signals or first physical quantities related to the function or output of the component. The effects listed in the first row are quantised descriptions of the properties of the signals. The relationship from input to output is indicated in matrix form in the propagation analysis. The causes due to different effects of the component are listed in the table. These very specific details about component failure are not used in our analysis, but are used as a good starting point for a systematic analysis. FPA schemes are available from several databases of component reliability, in particular from components used in the nuclear, space and avionics industries, where post-failure analysis has been made systematically. The scheme for the rate gyro is illustrated in the following table.

Signal	low	high	fluctuating	undefined
Fault	Electric short Electrical or mechanical defect	Electric short Electrical defect	Wire defect Unit damaged	Wire defect Unit damaged
Input:	low rate	high rate	supply power	dismounted

The FPA matrix for the rate gyro is defined by considering a generic failure of the rate gyro, which can cause any of the output signal conditions: *low, high, fluctuating* or *undefined*.

$$\boldsymbol{e}_{rg} \leftarrow \boldsymbol{M}_{rg}^{f} \otimes \begin{pmatrix} \boldsymbol{f}_{\omega} \\ \boldsymbol{e}_{ship} \end{pmatrix}$$

$$\begin{pmatrix} e_{rg,l} \\ e_{rg,h} \\ e_{rg,f} \\ e_{rg,u} \end{pmatrix} \leftarrow \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} f_{\omega} \\ e_{ship,l} \\ e_{ship,h} \end{pmatrix}$$

Observation of the set of end effects would show which fault(s) could be the cause(s) to a particular end effect combination,

$$\begin{pmatrix} \boldsymbol{f}_{rg} \\ \boldsymbol{e}_{ship} \end{pmatrix} \leftarrow \boldsymbol{M}^{b}_{rg} \odot \boldsymbol{e}_{rg}$$

$$\begin{pmatrix} f_{rg} \\ e_{ship,l} \\ e_{ship,h} \end{pmatrix} \leftarrow \begin{pmatrix} 1\ 1\ 1\ 1 \\ 1\ 0\ 0\ 0 \\ 0\ 1\ 0\ 0 \end{pmatrix} \odot \begin{pmatrix} e_{rg,l} \\ e_{rg,h} \\ e_{rg,f} \\ e_{rg,u} \end{pmatrix}$$

The interpretation is clearly that

$$e_{rg,l} \ \wedge e_{rg,h} \ \wedge e_{rg,f} \ \wedge e_{rg,u} \Rightarrow \ f_{rg}$$
$$e_{rg,l} \ \wedge \neg e_{rg,h} \ \wedge \neg e_{rg,f} \ \wedge \neg e_{rg,u} \Rightarrow \ e_{ship,l} \quad \text{etc.}$$

A complete systematic analysis will show which faults have severe end effects, i.e. effects that could cause the ship to make an unexpected alteration of heading. These include faults on either of the rate or heading gyro units, a fault in the steering gear and a fault in the heading reference to the autopilot. This list of faults is used when modelling the system and the faults to be diagnosed are identified from this list of high severity fault events. The fault propagation analysis uses knowledge of the overall characteristics of the effects of faults. To proceed in further detail with detection, we will later need a model where the specific faults are described as change of the parameters in a generic mathematical model. A generic fault model for the rate gyro is

$$\omega_{3m}(t) = (1 + \alpha_{\omega}(t))\,\omega_3(t) + f_{\omega}(t) + \nu_{\omega}(t),$$

where ω_{3m} is the measured signal, ω_3 the true turn rate. Faults will occur as changes in either of the signals α_{ω} or f_{ω}, both of which are functions of time. The signal $f_{\omega}(t)$ is additive in this model, $\alpha_{\omega}(t)$ is non-additive. Both are zero when no faults are present. The signal $\nu_{\omega}(t)$ represents measurement noise. Note that a non-additive fault can be omitted in the fault model, since a gain fault can be modelled through an additive term as

$$f_{\omega}(t) = \alpha(t)\,\omega_3(t). \quad \square$$

Completeness. Completeness of the fault effect vector is a necessary prerequisite for later fault detection and isolation, since the only faults that can be isolated are those specified in the design. Completeness is obtained if all possible component faults are considered. This is not achievable in a rigorous sense, but engineering experience from risk analysis makes it possible for practical purposes.

It is noted that completeness does not ensure that component fault isolation is possible because the mapping from fault to effects is not an isomorphism (one-to-one mapping): An observed effect could be caused by any out of several component faults.

Definition of generic components. The above introduction of fault propagation matrix to characterise propagation of fault through components leads to include the fault propagation matrix in the formal definition of a generic component

Definition 4.2 (*Generic component model* (*extended*)) A system component is defined by the model given in Definition 4.2 together with the additional model part:

$$
\begin{aligned}
&< \text{FPA input} \mid \text{use-mode} > ::= \{< \text{list of internal faults;} \\
&\qquad\qquad\qquad\qquad\qquad \text{list of input effects} >\mid \text{use-mode}\} \\
&< \text{FPA output} \mid \text{use-mode} > ::= \{< \text{list of output effects} >\mid \text{use-mode}\} \\
&< \text{FPA description} \mid \text{use-mode} > ::= \{< \text{FPA input; FPA output;} \\
&\qquad\qquad\qquad\qquad\qquad \text{FPA matrix;} > \mid \text{use-mode}\}.
\end{aligned}
$$

Example 4.4 Temperature control
This example illustrates the aggregation of simple components into a complex component that provides a temperature control service. The problem considered is to accommodate some of the faults that would stop the primary service of the temperature control loop in Fig. 4.3. A three-way valve controls the mixing of hot water returning from a tank with tempered water from a heat exchanger. The control objective is to keep the cooling water temperature of an exogenous process in the tank at a constant value. The valve is controlled by the temperature control loop, which consists of

- the actuator with AC motor,
- the temperature sensor (T),
- the controller with process interface (AI, AO, A/D, D/A), and
- the filter system.

The control loop is shown in Fig. 4.3.

The temperature control loop is a cascade control with position control of the valve as the inner loop. Stability of the total loop is not guaranteed if the inner loop becomes open due to a component fault.

Three-way valve actuator. The valve is driven by an AC motor that will rotate left or right, when activated to either side by a double acting relay. End-stop switches are supposed to avoid motor overload by preventing the motor from turning further when an end stop switch is reached. The potentiometer gives position feedback. The position loop fails if either potentiometer or end-stop switches fail.

Figure 4.4 shows the graphical representation of the FPA scheme for the closed-loop valve position controller. Bold lines in the scheme show how faults propagate. The important observation is that propagation could be stopped at the points marked with stars. This means that fault handling should be applied exactly at these points.

It is intuitive that accommodation of a position or limit switch fault could be done fairly simple:

1. Use an estimate of the valve position in the motor controller instead of a faulty position signal.
2. Override a limit switch information if both position sensor feedback and an estimated position show that a limit switch fault has occurred.

An observer for this purpose is elementary. The estimated valve position is increased or decreased in proportion to the time either of the two motor relays. This requires no additional hardware but a few lines of observer code. Accommodation of any of these sensor faults will make it possible to continue operation while giving an alert about required maintenance. Without accommodation, the temperature control loop would probably fail due to the loop becoming unstable without the internal position feedback.

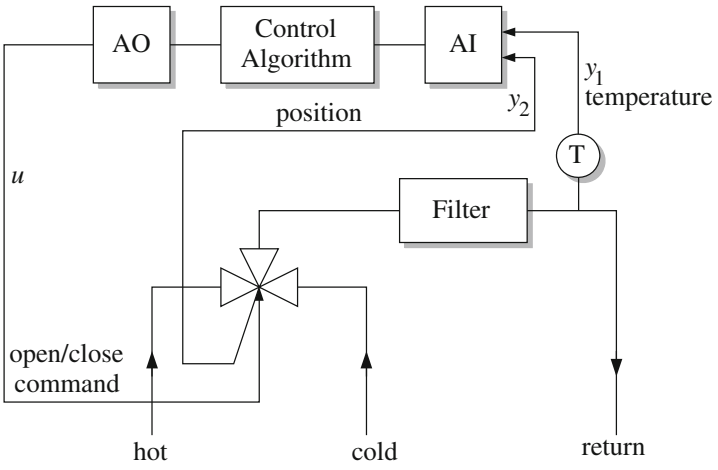


Fig. 4.3 Piping and instrumentation diagram representation of a temperature control loop with three-way valve

Figure 4.4 shows the FPA scheme in block diagram form for the valve control part of the loop. The components are potentiometer, limit switches, motor, three-way valve and digital controller.

Faults in a limit switch will prevent motion in clockwise or counter-clockwise direction—opening or closing of the valve. The consequence is a severe offset of the temperature control if fault handling is not initiated. A breakdown of the position feedback element will cause a breakdown of the temperature control loop because the motor will be driven rapidly to fully open or fully closed position.

Because several faults can cause the same effect, it is necessary to isolate the failure source. When the source is isolated it is possible to decide the reaction:

- **Actuator fault**. Fault in the valve up–down relay switch or in the ac-motor: The position controller must stop immediately. This will cause a loss of the temperature control service.
- **Actuator fault**. Fault in the valve end-stop switch: The position controller switches to use the position sensor and up–down commands for estimation of position. The service continues until maintenance.
- **Position sensor fault**. The controller should be re-configured. In the analytical relation between duration of relay pulses and motor shaft position, a position estimate is readily available. The estimate is used until the fault is repaired.
- **Temperature sensor fault**. The reference to the position controller fails. The controller is re-configured and a time-history roll-back is made of the reference signal and the mean is used as the new reference until the fault has been repaired.

This examples illustrate situations where temperature would deviate significantly or the control would simply fail with a commonly applied controller design. The temperature control service would no longer be available, and the overall use-mode of the tank would need to be changed to emptying, for safety reasons. By contrast, fault accommodation could assure availability of a reduced temperature control service, for several likely faults, thus enhancing the overall plant availability with simple means. □

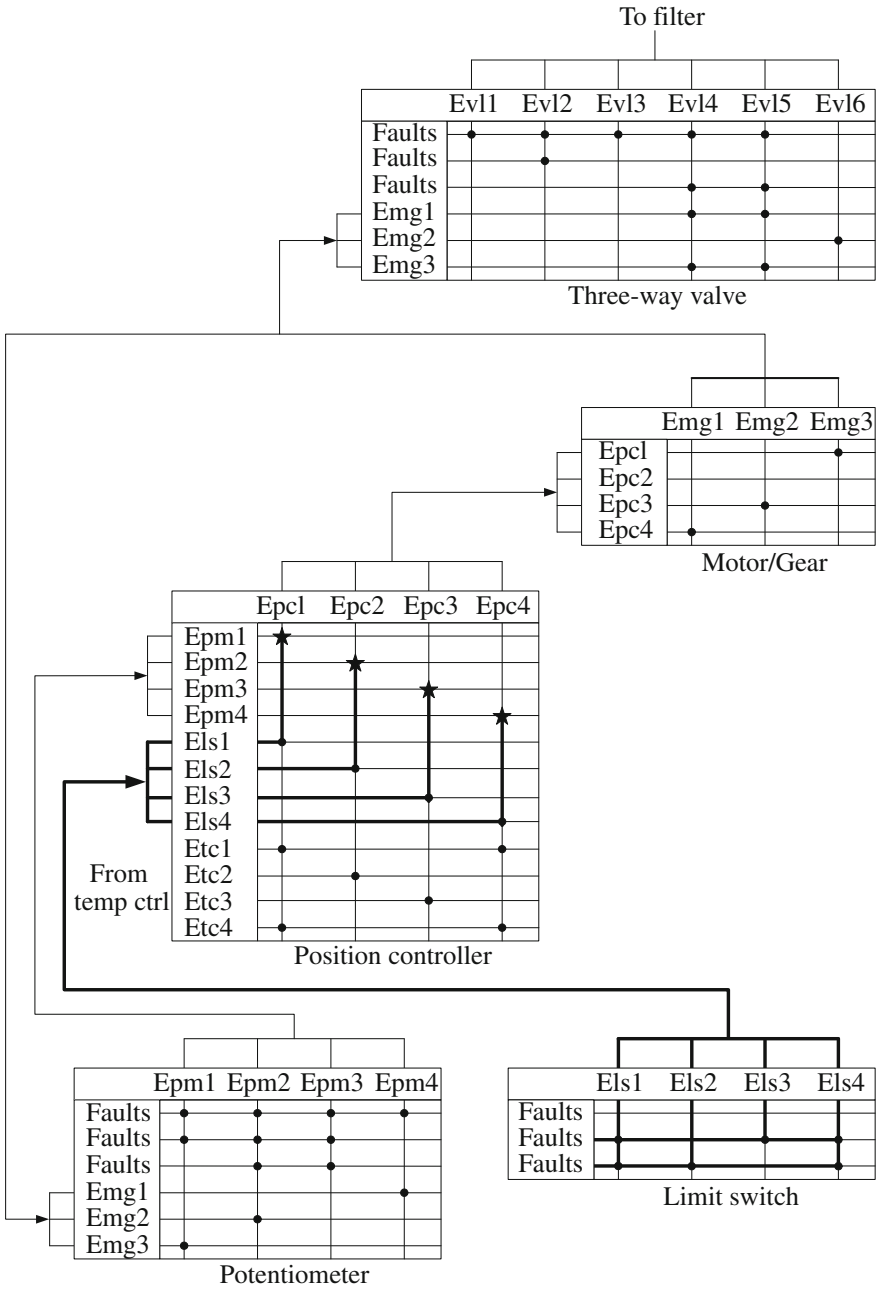


Fig. 4.4 Propagation of fault effects in closed-loop control of three-way valve. *Solid lines* show fault propagation, points marked with *star* show where propagation can be stopped

4.4 Graph Representation of Component Architecture

The above discussion has shown that a block diagram for the FPA analysis of an aggregated component consists of

- the external input faults or effects propagated to the component,
- the FPA representation of lower level components within the aggregated component, and
- the end effects for the aggregated component.

The latter can be considered output of the FPA analysis. The task of dealing with closed-loops in the FPA diagram can be eased by employing a graph formulation. An appropriate FPA graph is first defined. It is then shown that how closed loops are identified and finally how cut sets can be obtained.

Definition 4.3 (*Fault propagation analysis graph*) Let a system be comprised the following items: input effects ι, components with FPA blocks γ and output effects ζ. Define a set of vertices as $V = \{\iota, \gamma, \zeta\}$ of an FPA graph. Connections between the system items are edges of the graph. The edges constitute the set E. The FPA graph Γ is an ordered pair of disjoint sets (V, E). $V = V(\Gamma)$ is the set of vertices and $E = E(\Gamma)$ the edge set.

We further define an orientation of the edges in the FPA graph.

Definition 4.4 (*Orientation*) An edge (i, j) is said to connect a vertice j to i. If an edge is oriented and connects vertice j with i, then $e_{ij} = 1$.

This leads to a matrix representation of the FPA graph with oriented edges.

Definition 4.5 (*Directed adjacency matrix*) The directed adjacency matrix $\boldsymbol{D}$ of Γ, with respect to a given orientation of Γ, is the $n \times n$ matrix (d_{ij}) whose entries are

$$d_{ij} = \begin{cases} 1 & \text{if } v_i \text{ is the positive end of an edge from } v_j \\ 0 & \text{otherwise,} \end{cases}$$

where the number of vertices in the graph is n.

The directed adjacency matrix is thus square. The entries of the ith row show which connections point to the ith item (input, component, or output) in the fault propagation diagram. The cardinality of "1" entries in the directed adjacency matrix is equal to the number of edges in the graph.

Remark 4.2 (*Input vertex*) The ith vertex is an input vertex if and only if the ith row in the adjacency matrix comprises zeroes only. □

Remark 4.3 (*Output vertex*) The jth vertex is an output vertex if and only if the jth column in the adjacency matrix comprises zeroes only. □

Definition 4.6 (*Walk of length k*) A walk of length k in Γ is a finite sequence of vertices in the graph Γ $\{v_0, v_1, \ldots, v_k\}$ such that v_{t-1} and v_t are adjacent for $1 \leq t \leq k$.

Graph theory is very useful in respect to showing some general properties of the graph Γ.

Lemma 4.1 (Biggs) *The number of walks of length k in* Γ *from* v_i *to* v_j *is the entry in position* (i, j) *of the matrix* $\boldsymbol{D}^k$.

A closed loop is a walk that leads from an item and back to itself. Hence, a closed loop of length k will appear as a 1 in the diagonal of $\boldsymbol{D}^k$ for each vertex that is part of the loop. This gives an algorithm to find closed loops in a fault propagation graph.

Theorem 4.1 (Loops of length k in a fault propagation graph) *A graph with a vertex* v_i *has exactly one walk back to itself of length k if and only if the ith diagonal entry of* $\boldsymbol{D}^k$ *is 1. Each vertex in the loop has its diagonal entry equal to 1.*

A vertex v_i *that participates in n closed loops will have a diagonal entry in* $\boldsymbol{D}^k$ *equal to n. The number n will include possible multiple rounds in a loop if its length is an integer fraction of k.*

Diagonal elements of a vertice hence show how many closed loops of length k or k/M the vertice is part of, where M is an integer number. The power k of D used in the calculation shall not exceed the number of vertices in the graph.

Example 4.5 Closed loops in a fault propagation graph
A fault propagation graph is illustrated in Fig. 4.5.

The directed adjacency matrix is D in Eq. (4.1). Powers of the adjacency matrix are

$$\boldsymbol{D} = \begin{pmatrix} 0&0&0&0&0&0 \\ 1&0&0&1&0&1 \\ 0&1&0&0&0&0 \\ 0&1&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&1&0&0 \end{pmatrix}, \quad \boldsymbol{D}^2 = \begin{pmatrix} 0&0&0&0&0&0 \\ 0&1&1&1&0&0 \\ 1&0&0&1&0&1 \\ 1&1&0&1&0&1 \\ 0&1&1&0&0&0 \\ 0&1&1&0&0&0 \end{pmatrix} \tag{4.1}$$

$$\boldsymbol{D}^3 = \begin{pmatrix} 0&0&0&0&0&0 \\ 1&2&1&1&0&1 \\ 0&1&1&1&0&0 \\ 1&1&1&2&0&1 \\ 1&1&0&1&0&1 \\ 1&1&0&1&0&1 \end{pmatrix}, \quad \boldsymbol{D}^4 = \begin{pmatrix} 0&0&0&0&0&0 \\ 2&2&1&3&0&2 \\ 1&2&1&1&0&1 \\ 1&3&2&2&0&1 \\ 1&1&1&2&0&1 \\ 1&1&1&2&0&1 \end{pmatrix}$$

$$\boldsymbol{D}^5 = \begin{pmatrix} 0&0&0&0&0&0 \\ 2&4&3&4&0&2 \\ 2&2&1&3&0&2 \\ 3&4&2&4&0&3 \\ 1&3&2&2&0&1 \\ 1&3&2&2&0&1 \end{pmatrix}, \quad \boldsymbol{D}^6 = \begin{pmatrix} 0&0&0&0&0&0 \\ 4&7&4&6&0&4 \\ 2&4&3&4&0&2 \\ 4&6&4&7&0&4 \\ 3&4&2&4&0&3 \\ 3&4&2&4&0&3 \end{pmatrix}.$$

The diagonal of the $\boldsymbol{D}^k$ matrix shows the special characteristics:

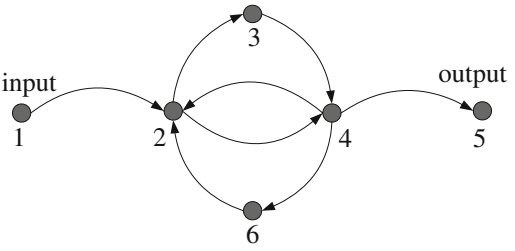


Fig. 4.5 A fault propagation graph example. One vertice is input (*1*), another is output (*5*)

- $\boldsymbol{D}^2$: 1 loop of length 2. It is {(2,4)}.
- $\boldsymbol{D}^3$: 2 loops through 2 and 4, one through 3 and 6. They are {(2,3,4), (2,6,4) }.
- $\boldsymbol{D}^4$: 2 loops through 2 and 4, one through 3 and 6. They are {(4,2,4,2), (3,4,6,2)}.
- $\boldsymbol{D}^5$: 4 loops through 2 and 4, one through 3 and 6. They are {(2,3,4,2,4), (6,2,4,2,4), (2,4,6,2,4), (2,4,2,3,4)}.

Element (5,1) in $\boldsymbol{D}^k$ shows for which k there is a connection from input to output. The shortest walk from input (1) to output (5) has length 3, as seen from element (5,1) in $\boldsymbol{D}^3$.

It should be noted that multiple rounds in loops are indeed part of the loop count for a vertex as seen in the diagonal entry of the $\boldsymbol{D}^k$ matrix. □

4.5 Fault Propagation with a Closed Loop

The failure mode and effects analysis scheme for a set of components connected in a closed logical loop is principally described as

$$\boldsymbol{e}_{ci} \leftarrow (\boldsymbol{M}_i^f \quad \boldsymbol{I}) \otimes \begin{pmatrix} \boldsymbol{f}_{ci} \\ \boldsymbol{e}_{ci} \end{pmatrix}.$$

Looking at the logic operation of this equation, the solution is

$$\boldsymbol{e}_{ci}^{+} \leftarrow \begin{cases} \boldsymbol{M}_i^f \otimes (f_{ci}) & \text{if and only if } \ \boldsymbol{e}_{ci}^{-} = \text{“0”} \\ \text{“}\mathbf{1}\text{”} & \text{if and only if } \ \boldsymbol{e}_{ci}^{-} = \text{“}\mathbf{1}\text{”}, \end{cases}$$

where $\boldsymbol{e}_{c1}^{-}$ is the state prior to the calculation, $\boldsymbol{e}_{ci}^{+}$ is the state resulting from the calculation. It is seen that once triggered, the effect $\boldsymbol{e}_{ci}$ remains permanently true, also after the fault disappears. This mechanism is a penalty of the Boolean representation of faults and their propagation, and the price for this is to get a fast tool for a first analysis of fault propagation.

When a closed logical loop is present, we hence need to cut an appropriate connection within the loop and investigate whether a “true” signal into the broken connection will produce a “true” at the other end of the cut. If this is the case, the loop is a tautology, a formula that is true in every interpretation, which can be eliminated. If the

"true" in produces a "false" at the other end of the cut, the logical loop is a contradiction and no solution exists. We then need to define the input of the cut as a new input in our failure mode and effects analysis description of the system, and analyse the propagation of an imagined "fault" condition = "true" from this point.

The non-existence of a logical model for a closed loop is not equivalent to instability in the continuous model representation. The stability of a closed-loop system cannot be determined from the properties of an over-simplified logical model of fault propagation.

In FMEA analysis, the closed-loop problem is artificially circumvented by treating the closed loop as a unit without feedback. The FMEA approach is hence to ignore the feedback loop as such and consider the closed-loop operation of the component as the functionality of the component and its' closed-loop considered as an extended component. Failure within the feedback loop itself is then treated by modelling this event as a separate fault. In essence, this is exactly what is done in this Boolean fault propagation analysis approach that was presented here. When we meet a closed loop, analysis can be achieved by extending the system with auxiliary faults. This technique is illustrated below.

4.5.1 Cutting the Closed Fault Propagation Loop

The existence and location of closed logical loops can be determined quite easily with the graph-theory-based tool we presented in Sect. 4.5. The directed adjacency matrix $\boldsymbol{D}$ and powers of $\boldsymbol{D}$ up to degree k showed which vertices of the FPA graph are parts of closed loops, if any, and it informed on the paths from input to output.

When a logical loop cut has to be made, it should be made such that the path from input to output is not interrupted, while cutting the relevant loop(s). The cut is conducted by cutting an edge, defining a new input and output as needed. The variables associated with the extra (new) input and output vertices are given by the variables associated with the edge that was cut. Logic analysis of the system is carried out using the new input as additional faults. The new output is observed. If the variable (effects) at the output is identical with the input, the relation is a tautology and can be removed from the analysis. If the result is a logic contradiction, the new input needs to remain in the analysis, and the logical loop remains open.

In conclusion, the representation of a fault effect as a Boolean signal {0, 1} and the fact that Boolean algebra prohibits dealing with closed-loop logic, unless the logic signals are clocked and therefore delayed as in flip-flop circuits, is a serious obstacle to a pure Boolean analysis of fault propagation.

Example 4.6 Ship track control - fault propagation
The propagation of faults from the track error sensor to the track controller are investigated in this example.

Track error sensor. An analysis of the track error sensor leads to the following internal failure modes

- $f_{\text{tes,hardware}}$: hardware fault causing abrupt fault (no output signal)

The input effects are effects from other components or external faults that are propagated to the component. Here effects from faults in the GPS receiver are considered:

- $e_{\text{gps,o}}$: GPS signal offset
- $e_{\text{gps,u}}$: GPS signal unavailable.

The output from the track error sensor is the track error signal. The effects are track_error low $e_{\text{te,l}}$, track_error high $e_{\text{te,h}}$ and track_error unavailable $e_{\text{te,u}}$.

$$\begin{pmatrix} e_{\text{te,l}} \\ e_{\text{te,h}} \\ e_{\text{te,u}} \end{pmatrix} \leftarrow \begin{pmatrix} 1\ 1\ 0 \\ 1\ 1\ 0 \\ 0\ 0\ 1 \end{pmatrix} \otimes \begin{pmatrix} f_{\text{tes,hardware}} \\ e_{\text{gps,o}} \\ e_{\text{gps,u}} \end{pmatrix}$$

Track controller. One internal fault in the track controller is considered in this example:

- $f_{\text{tc,software}}$: Software fault causing constant heading demand signal as output

$$\begin{pmatrix} e_{\psi_{\text{dem,l}}} \\ e_{\psi_{\text{dem,h}}} \\ e_{\psi_{\text{dem,f}}} \\ e_{\psi_{\text{dem,u}}} \end{pmatrix} \leftarrow \begin{pmatrix} 0\ 1\ 0\ 0 \\ 0\ 0\ 1\ 0 \\ 0\ 0\ 0\ 0 \\ 1\ 0\ 0\ 1 \end{pmatrix} \otimes \begin{pmatrix} f_{\text{tc,software}} \\ e_{\text{te,l}} \\ e_{\text{te,h}} \\ e_{\text{te,u}} \end{pmatrix}$$

Combining the Boolean propagation matrices for the track error sensor and the steering controller leads to a description of the propagation of the combined fault vectors for the two components to the output of the steering controller (Fig. 4.6).

$$\begin{aligned} \boldsymbol{M}^{f}_{\text{tes}\rightarrow\text{tc}} &= \left(\boldsymbol{M}^{f}_{\text{tc,f}} \ \ \boldsymbol{M}_{\text{tc,e}} \otimes \boldsymbol{M}^{f}_{\text{tes}} \right) \\ &= \left(\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \ \ \begin{pmatrix} 1\ 0\ 0 \\ 0\ 1\ 0 \\ 0\ 0\ 0 \\ 0\ 0\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1\ 1\ 0 \\ 1\ 1\ 0 \\ 0\ 0\ 1 \end{pmatrix} \right) \\ &= \begin{pmatrix} 0\ 1\ 1\ 0 \\ 0\ 1\ 1\ 0 \\ 0\ 0\ 0\ 0 \\ 1\ 0\ 0\ 1 \end{pmatrix} \end{aligned}$$

The resulting fault propagation is then

$$\begin{pmatrix} e_{\psi_{\text{dem,l}}} \\ e_{\psi_{\text{dem,h}}} \\ e_{\psi_{\text{dem,f}}} \\ e_{\psi_{\text{dem,u}}} \end{pmatrix} \leftarrow \begin{pmatrix} 0\ 1\ 1\ 0 \\ 0\ 1\ 1\ 0 \\ 0\ 0\ 0\ 0 \\ 1\ 0\ 0\ 1 \end{pmatrix} \otimes \left(\begin{pmatrix} f_{\text{tc,software}} \\ f_{\text{tes,hardware}} \\ e_{\text{gps,o}} \\ e_{\text{gps,u}} \end{pmatrix} \right). \qquad \square$$

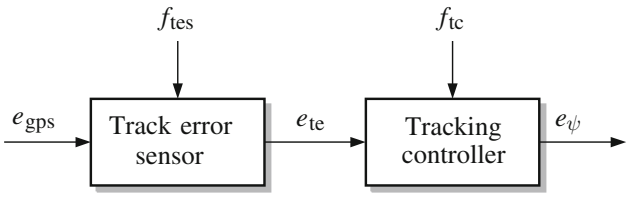


Fig. 4.6 Propagation of faults through the track error sensor and track controller

4.5.2 Assessment of the Severity of the Fault Effects

The consequences of a fault are judged from the implications the end effects could have on safety, on availability of the plant, on environment etc. A judgement of severity should be made of the possible combination of end effects, considering end effects that can arise from any single fault. This has as prerequisite that a single-fault assumption is sufficient for the analysis.

If a double fault is considered, the underlying logic description of fault propagation matrices must support such analysis.

4.5.3 Decision About Fault Handling

The implication is that an automated analysis will need to consider closed loops as special cases. The interpretation of a closed loop in an FPA scheme is merely the observation that closed-loop operation may amplify or attenuate the effects of a fault. Which of the two happens depends on the dynamical properties of the control loop and this question is outside the scope of the FPA analysis.

The component-based analysis can thus provide both a list of fault effects and a suggestion of where in a system fault propagation can be stopped. In the design method, it is then up to the designer to evaluate the severity of each fault effect and determine which fault accommodation actions shall be implemented.

The question how to handle faults will be discussed in Chap. 7.

4.6 Generic Component Models

Generic component models describe the system architecture from a formal point of view, so as to perform systematic manipulations for the purpose of fault diagnosis and fault-tolerant-control design. Contrary to box models, which carry no information about the component behaviour in different operating situations (normal, faulty, different modes), generic component models describe components, which offer services according to the current use-mode. The user may be a human operator (who

directly accesses the component through some man–machine interface) or another component, which accesses the services either through direct or remote connection (as in distributed systems in which services are requested via a local area network).

A generic model of a component describes its operational mode through the services it provides.

4.6.1 Services

From the user viewpoint, a system component provides one or several services. For example, a level sensor provides a signal which is a one-to-one correspondence to the level in the tank. However, the signal may be validated or not, it may be filtered or not, the sensor might memorise the minimal (the maximal) level value encountered on a given time window, it might provide an alarm if the level exceeds a given threshold, etc. All these are examples of services the sensor might provide in the normal operating mode. Other services could be provided in the installation, initialisation, degraded operation, maintenance modes.

Input, output and procedures. A service can be described by input variables, output variables, and some procedure which transforms the former into the latter. For example, a tank consumes input and output mass flows, and produces a stored mass, using an integration procedure (note that the output flow is indeed an input variable for the integration procedure), thus providing an *integration* service whose behavioural model is

$$\dot{h}(t) = q_i(t) - q_o(t),$$

where q_i is mass flow in q_o is mass flow out and $\dot{h}$ is mass increase rate. The *measurement* service of a sensor consumes energy from the outside world and produces a signal which is the image of the measured variable, by means of the transducer. The *controller* service of a controller consumes signals from sensors and produces signals to actuators. It also consumes data (the set-point) that have been previously written in the data base (using the *write* service).

Requests and enabling conditions. Services may be provided unconditionally or on specific request. For example, the *integration* service is systematically provided by a tank (no special request is necessary), at any time and whatever the values of the input and output as long as the tank level is within its rated capacity (no activation condition is needed). A sensor connected on some input port of a microprocessor system would provide the *measurement* service on a *read* request, which would be associated with some specific clock signal (the activation condition) issued to the analog to digital converter. A new set-point would be entered in the regulator memory on a specific *write* request from the human operator (again the request would be associated with an activation condition). The distinction between a request and an activation condition is that the request for a service is issued by the user, while the activation condition is processed by the component.

Resources. The normal running of a service needs some hardware resources. The tank is obviously necessary for the *integration* service to be performed. The transducer, filter, analog to digital converter, power supply and amplifier are necessary for the sensor to perform the *measurement* service. The microprocessor system is needed by the controller to provide the *regulation* service.

Summarising the preceding description under a formal model, a service is a 6-tuple:

<consumed variables, produced variables, procedure, request, activation condition, resources>.

As a consequence, a component is viewed as the set of services it can provide to the users, thus leading to the component model

$$S(k) = \{s_i(k), i \in I_s(k)\} \tag{4.2}$$

$$s_i(k) = \left\{cons_i(k),\ prod_i(k),\ proc_i(k),\ rqst_i(k),\ active_i(k),\ res_i(k)\right\}, \tag{4.3}$$

where $S(k)$ is the set of services of component k, I_s is the set of the indices of the possible services, and the other notations are straightforward.

Modes of operation. Obviously, not all the services provided by a component can be requested at any time during the system's life. A system generally goes through different operating modes, each with its set of prerequisites to function. For example, a request for the level control service from the controller of the single-tank system is denied: when the set-point has not been written (this calls for some initialisation mode); when the tank is empty (during a no-operation mode); when it is emptying during a cleaning mode.

For that reason, definition (4.2) is further extended, by adding some organising structure on the set of services. Normal operating modes are called *use-modes*. They provide the formal model of the structure of the set of services.

Definition 4.7 (*Use-mode* (*UM*)) A use-mode is a subset of services of a component. The set of use-modes covers the set of services, i.e. each service belongs at least to one use-mode, and each use-mode contains at least one service.

Let $M(k) = \{m_i(k), i \in I_m(k)\}$ be the set of use-modes of component k and $S_i(k) \subseteq S(k)$ be the services available in mode $m_i(k)$. Note that the formal definition of a use-mode does not tell which subsets of services have to be selected to form consistent use-modes. This matter is left to the design engineer, who indeed must group into use-modes subsets of services which are consistent in some given operation frame. In the single-tank system, the six possible use-modes are *Filling the tank, Processing batch, No operation* etc.

4.6.2 Introduction of the Generic Component Model

The consequence of the use-mode definition is that the component model must now include a (higher) level description, which models the component possible transitions from one use-mode to another one, and the conditions under which these transitions take place. Indeed, at any time t, the component is in one and only one use-mode, a discrete-event system behaviour which is easily modelled using a deterministic automaton (see Sect. 3.4. for an extensive presentation of discrete-event models). Note also that in order to obtain transitions between use-modes, it is necessary to add new services to $S(k)$. In the controller example, three possible use-modes and the corresponding list of services could be the following:

$$\begin{aligned} \textit{No-operation} &: m_1 = \{\textit{set\_mode\_}m_2,\ \textit{set\_mode\_}m_3\} \\ \textit{Initialise} &: m_2 = \{\textit{enter\_set-point},\ \textit{display\_set-point}, \\ &\qquad \textit{set\_mode\_}m_1,\ \textit{set\_mode\_}m_3\} \\ \textit{In-control} &: m_3 = \{\textit{read\_set-point},\ \textit{calculate\_control\_signal}, \\ &\qquad \textit{set\_mode\_}m_1\}. \end{aligned}$$

Taking into account the services and their organisation into use-modes, the generic model of a component is now defined.

Definition 4.8 (*Generic component model*) A system component is defined by the following formal model[1]:

$$\begin{aligned} &< \text{component k} > ::= < \text{state transition graph } G(M(k), \tau(k), m^0(k)) > \\ &\quad < M(k) > ::= < \text{set of use-modes } \{m_i(k), i \in I_m(k)\} > \\ &\quad < \tau(k) > ::= < \text{set of transitions } \left\{\tau_{ij}(k),\ i, j \in I_m(k)\right\} > \\ &\quad < m^0(k) > ::= < \text{initial use-mode} > \\ &< \text{use-mode } m_i(k) > ::= < \text{set of services } S_i(k) \subseteq S(k) > \\ &\quad < \text{service } s_1(k) > ::= < \text{pre-ordered versions} \\ &\qquad\qquad \left\{s_1^j(k), \quad j \in J(s_1(k))\right\} > \\ &\quad < \text{version } s_1^j(k) > ::= < \text{consumed vars } cons_1^j(k), \\ &\qquad\qquad \text{produced vars } prod_1(k), \\ &\qquad\qquad \text{procedures } proc_1^j(k),\ \text{request } rqst_1(k), \\ &\qquad\qquad \text{activation cond. } activ_1^j(k), \\ &\qquad\qquad \text{hardware and software resources } res_1^j(k) > \\ &< \text{transition } \tau_{ij}(k) > ::= < \text{condition } c_{ij}(k),\ \text{origin } m_i(k), \\ &\qquad\qquad \text{destination } m_j(k) > . \end{aligned}$$

This definition will be extended later.

[1]The notion of versions has been introduced in Sect. 3.2 and will be elaborated in more detail in Sect. 4.6.4.

4.6.3 Simple Components

A simple component is described by the services offered and its use-mode automaton. A component is considered simple when it has no internal means to change the services it provides. Simple components are typically without build-in computational means. Two examples of an actuator and a sensor are treated below.

Actuator for flow control. Flow control is the most widespread actuator function in machinery systems. It is used where a shut-off of a pipe connection is needed, where the flow of a medium is to be controlled, and where control loops manipulate a flow of a liquid in order to change a temperature. Flow control can be open/closed, variable throughput, or redirection of flow from one pipe into two (three-way valves).

The actuator system consists of a three-way valve. It has a common port and two other ports, referred to as A and B. The rotor position determines the opening area between the common port and ports A and B. The valve distributes the flow between ports A and B. The distribution is controlled by the rotor angle. An electro-mechanical device is attached to the valve to control the rotor position.

The electro-mechanical valve actuator consists of a motor and a gear. The rotor position is changed by running the motor in clockwise or counter-clockwise directions. The motor can be in one of the following states: stopped, rotation clockwise, or rotation counter-clockwise.

The state is controlled by activation of two relay contacts. They are denoted "open" and "close", respectively. A potentiometer is used to measure the actual rotor position. Figure 4.7 shows the principle in the actuator operation and electrical connections.

Limit switches on the rotor provide indication of rotor end positions and provide overload protection by preventing the motor to turn further in the direction of which the limit switch has been activated.

The service provided by the flow control valve is described by the six-tuple:

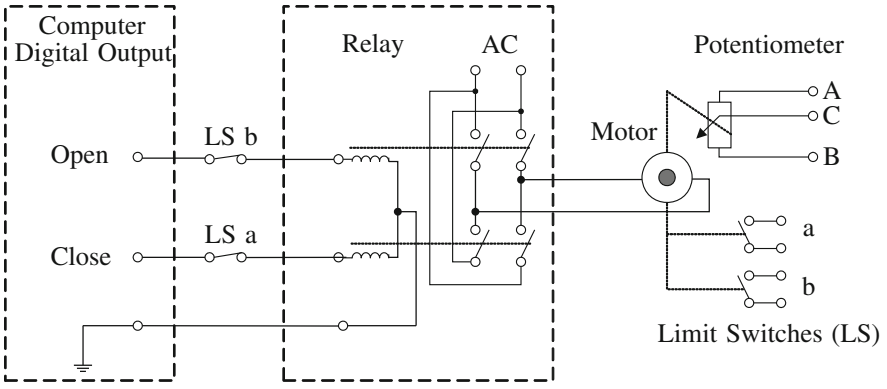


Fig. 4.7 Operation of three-way valve actuator with relay-operated induction motor. (Abbreviations: *o* open, *c* close, *LS* Limit switch, *AC* Alternating Current)

$$
\begin{aligned}
&< \textit{consumed variables} > ::=< \textit{open},\ \textit{close},\ \textit{in\_limit} >,\\
&< \textit{produced variables} > ::=< \textit{angle of output shaft},\ \textit{measured angle},\\
&\qquad\qquad \textit{in\_limit} >,\\
&< \textit{procedure} > ::=< \textit{angle} = \int \textit{up} \cdot \mathrm{d}t - \int \textit{down} \cdot \mathrm{d}t >,\\
&< \textit{request} > ::=< \textit{none} >,\\
&< \textit{activation condition} > ::=< 220\,\mathrm{V}\ \textit{present} >,\\
&< \textit{resources} > ::=< \textit{pot. meter},\ \textit{limit switches},\ \textit{controller},\\
&\qquad\qquad \textit{geared motor} > .
\end{aligned}
$$

The produced variable is the angle of output shaft, which influences the flow. Various faults can cause the flow to differ from the expected or desired value. The following table summarises various faults that can cause such a deviation.

Component/ Effect	Flow too low	Flow not related to control angle	Fluctuating flow	Flow too high output
Fault	Pipe broken, setpoint low, power low	Pipe clogged, pipe leak	Setpoint fluctuating	Too high input flow, setpoint high
	Pipe clogged	Pipe A or B broken, clogged or leak		
	Damage, wear	Hysteresis	Damage, wear	

Potentiometer. A potentiometer changes the position of contact between a resistance element and a wiper when the turning angle is changed. The potentiometer can be considered a voltage divider with a division ratio that is a function of the turning angle. Figure 4.8 shows the typical connection diagram.

Component/ Effect	Signal too low	Not related to angle	Fluctuating signal	Signal too high
Fault	Broken wire at A, short at A-B	Loss of supply	Vibration	Broken wire at A, short-circuit A-C
	Short B-C	Broken wire at C	loose conection	
	Stuck, shaft or element broken	Wiper fault		

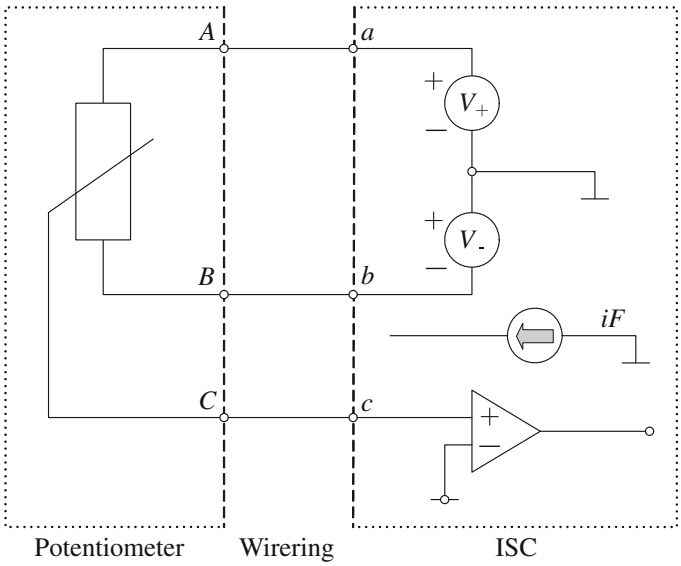


Fig. 4.8 Electrical diagram of potentiometer and computer interface to enable fault detection at the single sensor level

The service provided by the potentiometer as a sensor is an electrical signal proportional to the physical angle of rotation. Several faults will cause loss of this service. Short-circuit of any terminal to supply or to ground, or arbitrary wire disconnection are common events to cause component faults.

4.6.4 Complex Components

From a formal point of view, a component is a set of services. The consideration of aggregated, complex components leads to extend this description to the consideration of fault-tolerance capabilities.

Versions of services. Let $s_i(k)$ be a service provided by component k, and suppose that embedded fault tolerance is available. This means that the service $s_i(k)$ would not be interrupted even if the resources it needs were no longer available. This is only possible if it exists within component k under several versions, namely $s_i(k) = \left\{ s_i^j(k),\ j \in J(s_i(k)) \right\}$ where $s_i^j(k)$ is the jth version of service $s_i(k)$.

From this extension, definition (4.2) can now be stated as:

$$S(k) = \{ s_i(k),\ i \in I_s(k) \}$$
$$s_i(k) = \left\{ s_i^j(k),\ j \in J(s_i(k)) \right\},$$

where each version $s_i^j(k)$ of service $s_i(k)$ is a 6-tuple like (4.3), which can be used indifferently for the same purpose.

Versions pre-ordering. Designing the activation conditions of service versions rests on the definition of a pre-ordering. The versions of a service are separated into classes such that a service in class l is preferred to a service in class m if and only if $l < m$. For example, some versions might be more precise, faster, less consuming, etc. than others.

The design of the activation conditions is then straightforward: at time t when the request for service $s_i(k)$ is issued (or at any time if no request is necessary), the version which is to be run is the lowest rank one whose resources are all non-faulty. Two services in the same class are not ranked, which means that the choice is indifferent. Thus the activation conditions of each version can be chosen arbitrarily, provided they are mutually exclusive. Two examples of commonly used strategies are as follows:

1. Give an arbitrary preference order. Use version 1 as long as its resources are not faulty. Move to version 2 should version 1 fail.
2. Use the preceding scheme but regularly change the service ranking (e.g. in a circular way) so as to distribute the operation equally over time of the different versions.

A simple example of versions ranking is given by the *measurement* service of a sensor which includes two redundant transducers to measure the same variable. Let

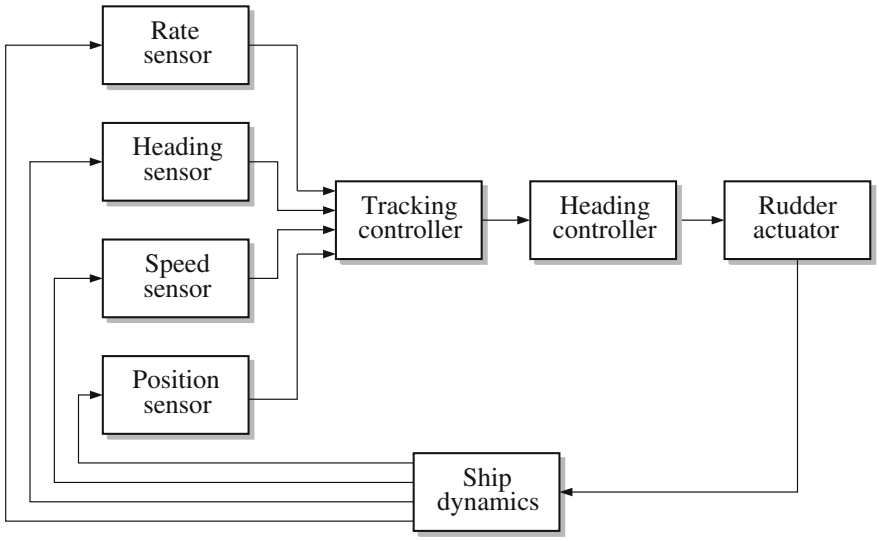


Fig. 4.9 Fault propagation in the ship steering problem

$$y_1(t) = x(t) + \varepsilon_1(t), \quad \varepsilon(t) \sim N(0,\ \sigma_1)$$
$$y_2(t) = x(t) + \varepsilon_2(t), \quad \varepsilon(t) \sim N(0,\ \sigma_2)$$

be the two measurement equations, with $\sigma_2 > \sigma_1$. The following table gives the different versions of the *measurement* service which are provided by this (intelligent) sensor, along with their ranking.

Class	Procedure	Fault situation
0	$y(t) = \frac{1}{\sigma_1+\sigma_2}[\sigma_2 y_1(t) + \sigma_1 y_2(t)]$	No fault
1	$y(t) = y_1(t)$	Transducer 2 faulty
2	$y(t) = y_2(t)$	Transducer 1 faulty

Example 4.7 Component analysis of ship track control
To illustrate the component analysis, it will be applied to the ship steering example introduced in Sect. 2.3 (Fig. 4.9). The subcomponents of the ship steering controller are as follows:

- Rate sensor (rate gyro),
- Heading sensor assembly (gyro compass),
- Track error sensor (Navigation computer with GPS input),
- Speed sensor (ship's log),
- Track control algorithm (software), and
- Heading control algorithm (software).

Ship steering controller. The ship steering controller has the following use-modes:

UM 0: No operation,
UM 1: Hand steering,
UM 2: Heading control mode, and
UM 3: Tracking control mode.

For each use-mode, a set of services are offered to the user (person or other subsystem)

$s(0) = <$ *set_track*; *set_heading* $>$
$s(1) = <$ *set_track*; *set_heading* $>$
$s(2) = <$ *set_track*; *set_heading*; *keep_heading* $>$
$s(3) = <$ *set_track*; *keep_track* $>$.

The *keep_heading* service calculates the necessary rudder action in order to maintain the heading of the ship based on the measured rate and heading. The service can be defined as

$$\begin{aligned} \textit{keep\_heading} = \quad & <\psi, \psi_{\text{ref}}, \omega_m>; \\ & <\delta>; \\ & <\textit{heading controller}>; \\ & <\psi_{\text{ref}}\ \textit{defined}>; \\ & <\textit{none}>; \\ & <\textit{Gyro},\ \textit{rate sensor}>. \end{aligned}$$

The *keep_track* service calculates the necessary heading in order to maintain the track of the ship based on the measured rate, heading and position. The service can be defined as

$$
\begin{aligned}
\mathit{keep\_track} = \quad & < \mathit{track\_error}\ e,\ \mathit{track},\ \mathit{ship\ speed}\ v_1 >; \\
& < \psi_{\text{ref}} >; \\
& < \mathit{track\ controller} >; \\
& < \mathit{track},\ \mathit{reference\ defined} >; \\
& < \mathit{none} >; \\
& < \mathit{Gyro},\ \mathit{rate\ sensor},\ \mathit{speed\ sensor} > .
\end{aligned}
$$

The *set_track* service prompts the user to input waypoints for the desired track:

$$
\begin{aligned}
\mathit{set\_track} = \quad & < \mathit{waypoints} >; \\
& < \mathit{track} >; \\
& < \mathit{track\ planner} >; \\
& < \mathit{user\ input} >; \\
& < \mathit{none} >; \\
& < \mathit{electronic\ seamap} > .
\end{aligned}
$$

The *set_heading* service prompts the user to set a desired heading:

$$
\begin{aligned}
\mathit{set\_heading} = \quad & < \psi_{\text{in}} >; \\
& < \psi_{\text{ref}} >; \\
& < \psi_{\text{ref}} = \psi_{\text{in}} >; \\
& < \mathit{user\ input} >; \\
& < \mathit{none} >; \\
& < \mathit{none} > .
\end{aligned}
$$

In the above service definitions, some resources are not considered, for example electrical power. □

4.6.5 Building Systems from Components

Systems are built from the interconnection of different components. Components are interconnected because the services delivered by some of them consume variables which are produced by services of others. The *measurement* service of a sensor, for example, consumes variables produced by the environment of the system, and produces variables which are consumed by the *regulation* service of the regulator, which in turn produces variables which are consumed by the *power modulation* service of the actuator.

In the generic model approach, interconnections are taken into account by considering higher-level components which are composed of lower level ones. Therefore, systems are built following a bottom-up approach.

Indeed, system architectures can be described at different hierarchical levels. Sensors, actuators, process components are at the field-level. Higher-level components can be built from the aggregation of lower level ones at any hierarchical level. For example, the aggregation of a tank, an input valve, an output pipe, a level sensor and a regulator (with consistent connections between them) is a high-level component,

the "single-tank system". Whatever the component level, its generic model can be built defining its use-mode automaton, and the services which are available in each use-mode. Aggregation procedures have to be defined in order to build the generic model of high-level components from the generic models of the low-level components they are composed of. High-level services allow to fulfil the system mission, and the analysis of the overall system fault tolerance can be based on the search of the existence of different versions of high-level services.

Aggregation of operation modes. The generic model of a component is first given by its use-mode automaton. In each use-mode, the component is able to perform a set of services (each of them under a variety of versions) in order to achieve some prespecified objective. Recall that the use-mode automaton which describes a component is a graph $A(M,\ \tau,\ m^0)$. Let $A(M(k),\ \tau(k),\ m^0(k))$ and $A(M(l),\ \tau(l),\ m^0(l))$ be the deterministic automata associated with two components k and l, and let kl be a higher-level component, which aggregates these two ones. The automaton $A(M(kl),\ \tau(kl),\ m^0(kl))$ associated with the component kl is obviously contained in the asynchronous product of the two automata $A(M(k),\ \tau(k),\ m^0(k))$ and $A(M(l),\ \tau(l),\ m^0(l))$:

- $M(kl) \subseteq M(k) \times M(l)$
- $\tau(kl) \subseteq \tau(k) \cup \tau(l)$
- $m^0(kl) = \left(m^0(k), m^0(l)\right)$

Let $(\alpha,\ \beta) = \mu \in M(k) \times M(l)$. This means that the mode μ of the high-level device kl is defined as component k being in mode α and component l being in mode β. Since not every such association is meaningful, the set of modes $M(kl)$ has to be selected by the designer, by eliminating from $M(k) \times M(l)$ the non-significant or non-allowed associations.

Example 4.8 Temperature controller
Consider a temperature controller as a high-level component, obtained by the aggregation of three low-level ones, namely a temperature sensor, a PI regulator, and a heating valve. The use-modes of the low-level components are as follows:

Sensor: {*off, calibration, automatic*}
Regulator: {*off, on* }
Valve:{*off, manual, automatic*},

where the calibration mode of the sensor and the manual mode of the valve are used for maintenance operations. Then, the asynchronous product of the three use-mode automata gives 18 compound modes. Many combinations are inconsistent, e.g. (*off, on, manual*) or (*calibration, on, automatic*), leaving only three consistent modes to describe the temperature controller device. The three use-modes of the high-level component are, therefore, {*off, maintenance, automatic*}, and they are defined as follows from the use-modes of the low-level components:

$$\begin{aligned} \mathit{off} &= (\mathit{off}, \mathit{off}, \mathit{off}) \\ \mathit{maintenance} &= (\mathit{calibration}, \mathit{off}, \mathit{off}) \vee (\mathit{calibration}, \mathit{off}, \mathit{manual}) \\ &\quad \vee (\mathit{off}, \mathit{off}, \mathit{manual}) \\ \mathit{automatic} &= (\mathit{automatic}, \mathit{on}, \mathit{automatic}). \end{aligned}$$

□

Aggregation of services. Let $S(k)$ and $S(l)$ be the services offered by two low-level components k and l, and let us consider the high-level component kl which is their aggregation. Let $(\alpha, \beta) = \mu \in M(kl)$ be a consistent use-mode, then any combination of the services $S_\alpha(k)$ (available when component k is in mode α) and $S_\beta(l)$ (available when component l is in mode β) can be used. In other words, any *program* using the services of $S_\alpha(k)$ and $S_\beta(k)$ as *instructions* can be a service available in the mode μ. Again, every combination is not consistent, and only programs with functional interpretations in the application framework are to be considered.

Example 4.8 (cont.) Temperature controller
The following program defines a high-level service, available in the automatic mode of the temperature controller component:

Regulation service:
Repeat,
  Request the measurement service of the sensor,
  Request the calculation service of the regulator,
  Request the actuation service of the valve,
Until end of the regulation service.

Note that if the measurement service of the sensor is available under three versions, the calculation service under two versions, and the actuation service under two versions, then the regulation service is available under twelve versions. □

Hierarchical levels. System architectures can be described at different hierarchical levels. Sensors, actuators, process components are at the field-level (they exchange data at fieldbus level). Higher-level components can be built from the aggregation of lower level ones at any hierarchical level. For example, the aggregation of a tank, an input valve, an output pipe, a level sensor and a controller (with consistent connections between them) is a high-level component which can be named the single-tank subsystem. Whatever the component level, its generic model can be built defining its use-mode automaton, and the services which are available in each use-mode. Aggregation procedures have to be defined in order to build the generic model of higher-level components from the generic models of lower level components. High-level services enable fulfilment of the system mission, and the analysis of the overall system fault tolerance can be based on the search of the existence of different versions of high-level services.

4.7 Fault-Tolerance Analysis

Fault tolerance is defined as the possibility of achieving a given (set of) objective(s) in the presence of a given (set of) fault(s). In the generic model, objectives and services are associated with each use-mode. Thus, the system is fault tolerant in the current use-mode as long as services which allow to achieve the current objectives are available in this use-mode.

Therefore, the analysis of fault tolerance rests on three points.

1. Are there services which allow to achieve the current objectives?
2. How are these services to be managed when faults occur?
3. How are the use-modes to be managed when faults occur?

4.7.1 Relation Between Services and Objectives

The generic component model describes the normal behaviour of the system components, at any hierarchical level, since high-level components are built, following a bottom-up approach, from the aggregation of low-level ones.

At the highest hierarchical level, the system itself is modelled as a single component, which aggregates all the elementary components, and whose services correspond to the missions or objectives that it has to achieve. Different aggregation paths can be followed between the field-level (associated with elementary components) and the system level. A natural way of building the bottom-up aggregation procedure is to make use of the intuitive decomposition of the system into subsystems whose functions (and services) can be clearly defined.

System pyramidal decomposition. Hierarchical decomposition splits a system into a set of subsystems, which can themselves be further decomposed, each subsystem being associated with a clear functional viewpoint. For example, a chemical process can be decomposed into

- a subsystem which aims at controlling the pH,
- a subsystem which aims at controlling the level, and
- a subsystem which aims at controlling the temperature.

The pH control subsystem may be further decomposed into the valve controlling the acid inflow, the valve controlling the base inflow and the stirr motor.

Since some components may belong to several subsystems, it is convenient to use a pyramidal decomposition (Fig. 4.10) whose number of levels is decided by the

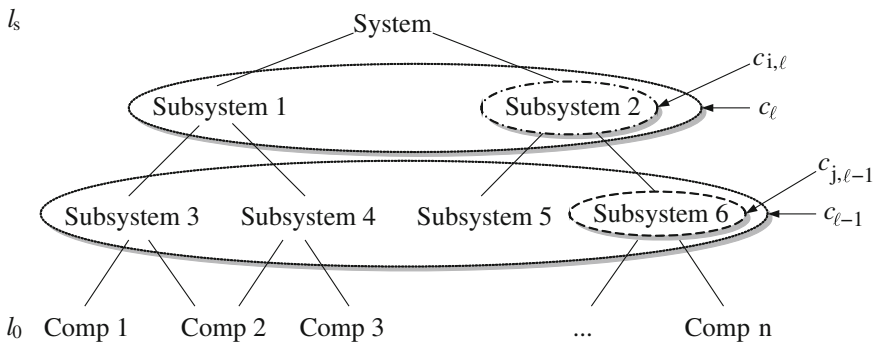


Fig. 4.10 Aggregation of low-level components into high-level ones

designer so as to obtain the view of the system which suits him best. Let $l = 1$ be the lowest decomposition level (the level of the field components) and $l = n$ be the highest decomposition level (the level of the system itself).

In a pyramidal decomposition, the following properties hold ($l \geq 2$):

- Each component of level $l - 1$ belongs to at least one component of level l.
- Any component of level l includes at least one component of level $l - 1$.

High-level services. Let C_l be the set of components modelled at level l, ($l = 1, \ldots, n$), and let $c \in C_l$ and $\gamma \in C_{l-1}$ ($l \geq 2$). Remember that the services of high-level components are behaviours which use the services of the low-level components they aggregate. Of course, not any combination of low-level services makes sense, and it is necessary, for the subsystems of the pyramidal decomposition to be consistent, that component $\gamma \in C_{l-1}$ is included in component $c \in C_l$ only if at least one service of component γ is used in at least one program which defines a service of component c associated with a known functional interpretation. Indeed, associating elementary services to define a high-level service is always realised to answer a functional requirement of the application. The system pyramidal decomposition is in this case, a real help define high-level services from low-level ones.

Let $S(\gamma)$ be the services offered by a component $\gamma \in C_{l-1}$ and let $c \in C_l$ be the aggregation of a set Γ of such components. Let

$$cons\,(c) = \cup_{\gamma \in \Gamma} \cup_{s \in S(\gamma)} cons\,(s)$$
$$prod\,(c) = \cup_{\gamma \in \Gamma} \cup_{s \in S(\gamma)} prod\,(s).$$

Note that $cons(c) \cap prod(c)$ may be non-empty, since some components in Γ may consume variables produced by some other ones. Note also that any relation between a subset of variables of $cons(c)$ and a subset of variables of $prod(c)$ can be obtained as the result of a program using the services of $\cup_{\gamma \in \Gamma} S(\gamma)$. If there exists such a relation which makes sense from a functional point of view in the system, then the subsystem $c \in C_l$ can be created at level l and the procedure which establishes such a relation can be defined as a service of the aggregated component c. Note finally that there might exist several subsets of components in Γ and several procedures which establish the same relation between the above mentioned variables. Then, the service exists under several versions. The set of all the versions of a service which can be obtained by aggregation of lower level ones can be found in a rather automated way, for simple kinds of programs composed of sequences and parallel executions.

Once the services of the subsystems have been determined at any level (including the overall system level), the ordering of the versions is left to the designer.

Example 4.9 High-level regulation service
Consider the three following low-level components:

- γ_1 is the temperature sensor, whose measurement service is defined by

$$< \theta,\ \hat{\theta},\ f_1,\ rqst_1,\ enable_1,\ res_1 >$$

where θ is the actual temperature, $\hat{\theta}$ is its estimate provided by the sensor, f_1 is the procedure which is used to produce the estimate: $\hat{\theta} = f_1(\theta)$. The request, enabling conditions and resources are not of interest here.

- γ_2 is the controller, whose computation service is defined by

$$< (\hat{\theta},\ \theta^*),\ u,\ f_2,\ rqst_2,\ enable_2,\ res_2 >$$

where θ^* is the temperature set-point, u is the control signal, and f_2 is the procedure which is used to produce the control signal: $u = f_2(\hat{\theta},\ \theta^*)$.

- γ_3 is the actuator, whose heating service is defined by

$$< u,\ \pi,\ f_3,\ rqst_3,\ enable_3,\ res_3 >$$

where u is the control signal, π is the delivered heating power, and f_3 is the procedure which is used to produce the heating power: $\pi = f_3(u)$.

Considering the set $\Gamma = \{\gamma_1,\ \gamma_2,\ \gamma_3\}$ as a candidate for aggregation into one higher-level component c, the above sets are $cons\,(c) = \left\{\theta,\ \hat{\theta},\ \theta^*,\ u\right\}$ and $prod\,(c) = \left\{\hat{\theta},\ u,\ \pi\right\}$, from which it is seen that the simple program sequence "measure, compute, actuate" can provide a relation between $\theta,\ \theta^*$ and π, whose functional interpretation is of course that of a "regulation" service.

Moreover, note that if the measurement service is provided e.g. under four versions (as in the following example), and the heating service is provided under two versions, then the regulation service is provided under eight different versions. □

4.7.2 Management of Service Versions

Consider a service s at the system level. It is a set of pre-ordered versions $s = \left\{s^j,\ j \in J(s)\right\}$. Each version can be used for the same purpose, namely to achieve the system objective(s) in a given use-mode, but the pre-ordering expresses a preference between them.

Two conditions have to be fulfilled for service versions to be enabled at some given time. First, the service must belong to the list of services of the current use-mode. This is a straightforward condition, which insures that only services consistent with the current objectives can be run.

The second condition is related with faults. Suppose that some resource from the set res^j is detected faulty by the fault diagnosis procedure at time t. Then, obviously, running the version s^j of the service s would produce incorrect values of the variables *prod*, and this should be forbidden, putting $enable^j = 0$. Note that this might be impossible, since faults might have ever-lasting delivery of some service as a consequence.

Example 4.10 Unavailable and ever-lasting services
Let *V_open* and *V_close* be two services delivered by an on/off valve, and suppose that the valve is blocked closed. Then, the service *V_open* becomes unavailable while the service *V_close* is permanent in time. □

Thus, the consequence of faults is that some services become permanent in time, while some others exist under a number of available versions which depend on the remaining, non-faulty resources. The status of these service obviously depends on the number of available versions, according to the following classification:

- at least one version is available: the service is available,
- no version is available: the service is unavailable.

Note that when more than one version is available, the lowest rank version of the service (which is the most preferred among the available ones) is to be run when the service is requested. Note also that the severity of the failure of a given resource with respect to the service can be evaluated by counting the number of versions which still are available after the failure has occurred. A resource for which this number is zero, or whose failure causes the service to run permanently in time is called a *critical resource*.

Example 4.11 Management of service versions in an intelligent sensor
Consider again the *measurement* service of the temperature sensor and suppose that it includes two redundant transducers and an observer. Let

$$y_1(t) = x(t) + \varepsilon_1(t), \quad \varepsilon_1(t) \sim N(0,\ \sigma_1)$$
$$y_2(t) = x(t) + \varepsilon_2(t), \quad \varepsilon_2(t) \sim N(0,\ \sigma_2)$$

be the two local measurement equations, and

$$\hat{y}(t) = f(z_1(t), z_2(t))$$

be the observer algorithm, where $z_1(t),\ z_2(t)$ are remote measurements obtained from a local area network communication (LAN) system. The following table gives the different versions of the *measurement* service which are provided by this sensor, along with their ranking.

Class	Procedure	Faultsituation
0	$y(t) = \frac{1}{\sigma_1+\sigma_2}(\sigma_2 y_1(t) + \sigma_1 y_2(t))$	no fault
1	$y(t) = y_1(t)$	T_1 ok., T_2 faulty, A/D ok.
1	$y(t) = y_2(t)$	T_2 ok., T_1 faulty, A/D ok.
2	$y(t) = \hat{y}(t)$	T_1 and T_2 or A/D faulty, LAN ok.

Suppose that the measurement request is issued by the system clock according to some sampling period, and that the measurement service is consistent with the current use-mode. Then, it will be provided under version 0 if both transducers are operating well, and under version 1 in the presence of a single transducer fault (note that the two versions 1 are mutually exclusive,

so that no conflict is possible). If the local Analog to Digital Converter (ADC) fails, version 2 can still be used, and the service will become unavailable only when local measurements and fieldbus communication will be all faulty. Remind that the local ADC was a critical resource when no observer was included in the sensor. Note also that version 2 of the measurement service might be much more unprecise than versions 0 and 1, thus the service would be degraded when this version is used. However, it would still be acceptable, otherwise version 2 should never have been included in the list of versions by the design engineer. □

4.7.3 Management of Operation Modes

Remember that the set of services is organised into use-modes, whose behaviour is described by a deterministic automaton. Let $A(M,\ \tau,\ m^0)$ be the use-mode automaton at the system level

- $M = \{m_i,\ i \in I_m\}$ is the set of the use-modes. Remember that each of these modes m_i is associated with the set of the services $S_i \subseteq S$ by which it is defined,
- $\tau = \left\{\tau_{ij},\ i,\ j \in I_m\right\}$ is the set of transitions,
- m^0 is the initial use-mode.

Critical services. In this chapter, use-modes have been further associated with objectives which have to be fulfilled thanks to those services. Let O_i be a set of objectives associated with use-mode m_i. As long as the set of services $S_i \subseteq S$ associated with m_i are available, the objectives O_i can obviously be achieved (otherwise, the component would be inconsistently designed). Note that this is true, by definition, whatever the version of the service. Versions of high rank provide degraded service, thus achieving the objective in a degraded but still acceptable manner. If not, the versions of that rank should not have been included in the list of possible versions of the service.

Suppose now that some fault has occurred such that some services of S_i become unavailable or run permanently. Then some objectives of O_i might become impossible to achieve. Let *critical services* be services whose unavailability or permanent running implies that at least one objective of the mode to which they belong cannot be achieved. Then, the $A(M,\ \tau,\ m^0)$ automaton model is extended as follows: $M = \{m_i,\ i \in I_m\}$ is the set of the use-modes. Each mode m_i is associated with the set of objectives O_i and the set of the services $S_i \subseteq S$ which is decomposed into $S_i = S_i^c \cup S_i^{nc}$, where S_i^c are the critical and S_i^{nc} are the uncritical ones.

Example 4.12 Critical service in the single-tank system
Consider the single-tank system given in the introduction, used in a food industry batch production process, and suppose the current use-mode is UM_2: processing the batch. In that use-mode, the system objective is to regulate the temperature, and the regulation service is thus a critical resource, since UM_2 objective cannot be achieved if this service is lost. □

Staying in a mode. Consider the system operation in a given current use-mode, and suppose that faults occur which cause the loss or permanent running of services (remember that as long as at least one service version is available, the service is

not lost). When non-critical services are lost or run permanently, the system can obviously remain in the current use-mode, since this use-mode objectives can still be achieved—eventually in a degraded manner—and thus the system is fault tolerant with respect to the current objectives and the current fault situation.

On the contrary, when critical services of the current use-mode are lost or run permanently, the objectives associated with that use-mode can no longer be achieved, and the system is to be given other objectives. This strategy is called an objective reconfiguration strategy. The only way it can be implemented is by firing a transition towards another use-mode, whose objectives will become the current ones (for example change the production recipe, or stop the production and transfer the system to a safe state, in which maintenance can be undertaken).

In general, several other use-modes can be reached from the current one, and the choice of the destination use-mode (i.e. of the new system objectives) is a difficult decision problem, which has to be considered in the system design stage. Unless the system objectives can be ranked according to a total ordering relation, the solution to that problem can in general only be partially automated, thus leaving a very important role to human operators in fault situations.

Transitions between modes. When objective reconfiguration is necessary, the system is commanded to another use-mode whose objectives will become the new ones. The system should, obviously, be able to achieve these new objectives, which means that in the destination use-mode, no critical service is unavailable nor is permanently running as a result of the current fault situation.

This remark is also valid when the transition does not follow from an objective reconfiguration strategy but from the normal operation of the system.

4.8 Exercises

Exercise 4.1 Fault propagation analysis for industrial actuator
Consider the component block diagram of the position servo shown in Fig. 4.11. The blocks in the figure are motor, amplifier, controller, potentiometer, gear and reference.

1. Construct a fault propagation matrix M_{p} for the potentiometer (use the FMEA matrix from p. 102). Use angle as input and voltage v_C at terminal C as output. Consider only the fault f_{p1}, which indicates the broken wire at C.
2. Using the above figure, define the component architecture in form of the directed adjacency matrix $\boldsymbol{D}$.
3. Determine which node is an input node from the adjacency matrix.
4. Determine the number of closed loops and the length of these.

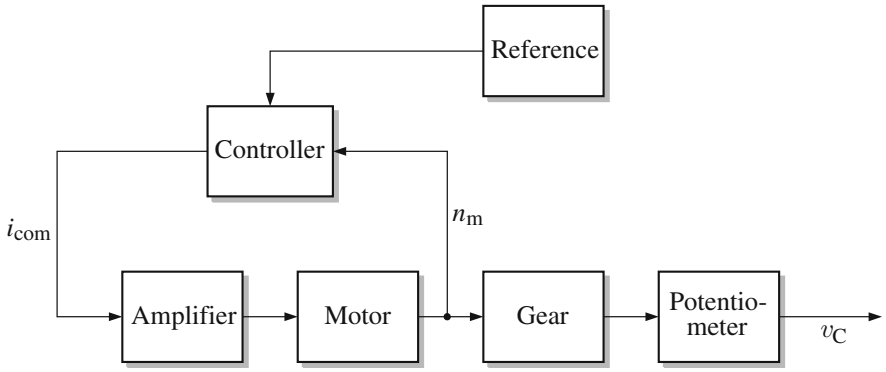


Fig. 4.11 Component diagram for speed loop part of the industrial actuator

The combined fault propagation matrix for motor and amplifier is defined by input and internal faults:

f_{a1} low gain in amplifier
f_{m1} brush partial disconnect
e_{a1} low input command
e_{a2} high input command
e_{a3} fluctuating input command

output:

e_{n1} low output speed
e_{n2} high output command
e_{n3} fluctuating output speed
e_{n4} output speed not related to input command

$$\boldsymbol{M}_{am} = \begin{pmatrix} 1\,1\,1\,0\,0 \\ 0\,0\,0\,1\,0 \\ 0\,1\,0\,0\,1 \\ 1\,1\,0\,0\,0 \end{pmatrix}.$$

Without further explanation, assume that the gear has the propagation matrix

$$\boldsymbol{M}_{g} = \begin{pmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{pmatrix}.$$

5. Determine the fault propagation from i_{com} to θ_m, using $\boldsymbol{M}_{am}$, $\boldsymbol{M}_{g}$ and $\boldsymbol{M}_{\text{p}}$.
6. Compute the inverse fault propagation matrix, i.e. observe the end effects on θ_m and see which faults or input effects could cause the end effects. □

Exercise 4.2 Component-based analysis of battery charger
A battery charger component diagram is given as shown in Fig. 4.12. The behaviours of the individual components are not known in detail but are given as

$$
\begin{aligned}
&\textit{Power stage mean current:} && I = d \cdot \max(V_{\text{sup}} - V_{\text{bat}}, 0)\\
&\textit{Current control:} && d = f_i(I_{\text{com}} - I_{\text{mes}})\\
&\textit{Voltage control:} && I_{\text{com}} = f_v(V_{\text{ref}} - V_{\text{mes}})\\
&\textit{Current sensor:} && I_{\text{mes}} = I\\
&\textit{Voltage sensor:} && V_{\text{mes}} = V_{\text{bat}}\\
&\textit{Harness:} && I_{\text{bat}} = I\\
&\textit{Battery voltage:} && V_{\text{bat}} = \frac{\alpha_{\text{bat}}}{C_{\text{bat}}} \int_0^t I_{\text{bat}}(t)\,\mathrm{d}t, \quad \alpha_{\text{bat}} \simeq 0.7.
\end{aligned}
\tag{4.4}
$$

1. Define one fault for each of the components: PWM converter, controller block (current and voltage control), current sensor, voltage sensor. Assume that the supply voltage and battery cannot fail.
2. Determine the fault propagation matrices for these components.
3. Determine the closed logical loops in the battery charger.
4. Cut the loop at the signal d_{com} and determine whether the logical loop has a solution. If not, define d_{com} as an ancillary input.
5. Determine the end effects (on I_{bat} and V_{bat}) for the faults you defined.
6. Express the inverse propagation and list the end effects.
7. Suggest how a fault in the current sensor could be accommodated. □

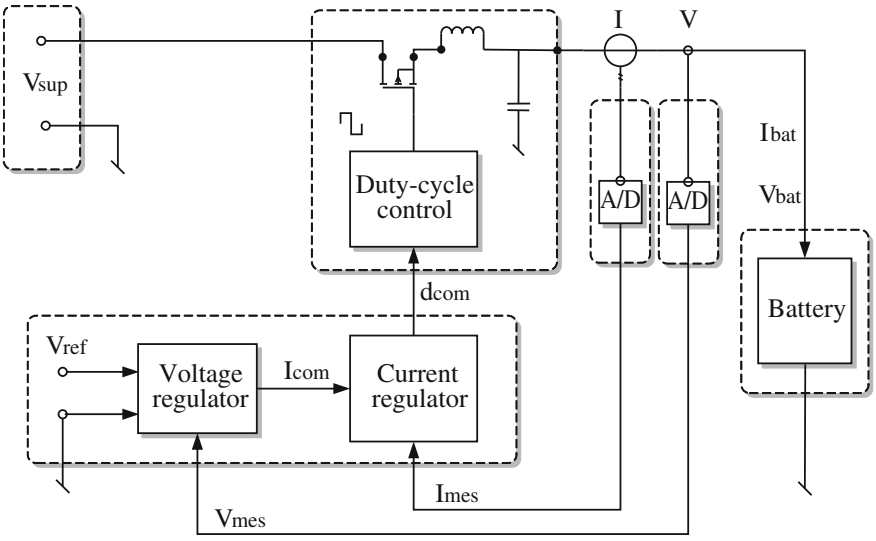


Fig. 4.12 Component diagram of battery charger

4.9 Bibliographical Notes

Architecture models. Basic bibliographical notes on architecture and generic component models were given in Chap. 2. With the purpose of describing complex interconnected systems, modelling extensions have been suggested in form of a bottom-up procedure, which allows to describe high-level devices by the interconnection of low-level ones, see [49, 50].

Generic component models. A semantics for services based on generic component models was developed in [324] where a graphic analysis was found to be very useful. A review of the use of graphical methods was provided in [47].

Failure modes and effects analysis. FMEA is a classical and widely used tool in industry [147]. Presentation of a matrix formulation suited for computational treatment of FMEA schemes was first presented by [192]. The fault propagation analysis was proposed in [22] and further elaborated in [42].

Fault reasoning and data validation. Reasoning and detection of faults, and hierarchical data validation was introduced in [14], while alarm filtering applications were considered in [258]. Systematic analysis of fault propagation [22, 42] was shown to be an essential tool for determination of severity of fault effects and for assessment of remedial actions early in the design phase. Reference [72] presented a comprehensive diagnosis methodology for complex hybrid systems and an application to aircraft power generator diagnosis was included in [358].

Fault tolerance. The analysis of fault propagation has also penetrated to design of industrial systems, Fault-tolerant steering by wire was obtained in [35] and a fault-tolerant three-phase speed drive for AC motors was described in [118].

Distributed systems. Fault tolerance in distributed systems has mainly been based on modular models [165]. The generic component model was used for defining the faulty resources management and the mode management layers [120]. Reconfigurability analysis was developed more recently [121]. Important application areas is prognosis and health management and an application for a distributed medical equipment a distributed concept was illustrated in [102].

Other applications and trends. Consequences of faults and how faults propagate attain continuous attention and documentation of fault effects properties are mandatory in product development in many areas. Transportation systems are among those expected and required by society to be safe. In these and many other systems, the complexity is an obstacle when attempting to conduct covering assessments. The complexity of computer-based safety systems was considered in [271], where a new method was proposed for joint design optimization and engineering failure analysis. The criticality of component failures in the petrochemical processes were treated in [136].

Software. All major functionalities in automation systems depend on software implementation and computer hardware. The complexity of software is immense and

analysis of failure modes and of fault propagation in software is a discipline in itself, which is not within the scope of the present text. Nevertheless it could be mentioned that [135] analysed and experimented with software fault injection aiming at FMEA.

Chapter 5
Structural Analysis

Abstract This chapter uses the structure graph to describe the direct interactions among the signals within a dynamical system. This graph allows to analyse the redundancies within a system, which can be exploited for fault diagnosis and control reconfiguration. A dynamical model is interpreted as a set of constraints, which leads to a bipartite graph representing the system structure. Faults indicate violations of the constraints. The analysis shows how component faults can be found by defining and utilising analytic redundancy relations.

5.1 Introduction

This chapter investigates the structural properties of dynamical systems by analysing their structural model. The structural model of a system is an abstraction of the behavioural model in the sense that instead of the constraints themselves only the structure of the constraints is considered. The structure graph is a representation of the links between constraints and the variables and parameters occurring in each constraint. The structure is represented by a *bipartite graph*, which is independent of the nature of the constraints, and of the variables and parameter values. Structural analysis can hence describe systems described by quantitative or qualitative relations, by equations, by rules or by tabular relations. The structure graph will be shown to represent a qualitative and easy-to-obtain model of the system.

Structural analysis is based on a description of the *normal behaviour* of a system, through describing the normal behaviour of each component and the relation (connection) the components have to variables we wish to consider in the system. Using this approach, we will be able to diagnose whether a *violation of a normal behaviour* has happened. This means we are no longer bound to describe all possible faults that could happen in components of a system, as was the case with FMEA and Fault Propagation Analysis methods presented in Chap. 4. Instead, the starting point of analysis is a set of constraints, a set of nominal input–output relations, which describe the system as a set of components, the normal behaviour of each component and the topology of the components that constitute the system. Any violation of a constraint is considered to be a fault, without specifying the physical reason behind each possible fault.

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8_5

The structural model will be shown to be represented by a structure graph, composed of constraints and variables. This graph is independent of the value of the system parameters, and structural properties are deducted using *graph theory* for graphs that are partitioned into two sets: constraints and variables. The properties of this bipartitioned graph are explored in great detail in this chapter.

In spite of their apparent simplicity, structural models provide significant information for use in fault diagnosis and fault-tolerant control. This approach is able to identify those constraints, and related components of the system, which are—or are not—monitorable, to provide calculation of residuals from the analytic redundancy relations (ARR) that exist in a system, and to find those components whose failure can be tolerated through reconfiguration.

In this chapter, structural investigations concern

- identification of the *monitorable part* of the system, i.e. the subset of the system components whose faults can be detected and possibly isolated,
- direct generation of residuals for a system that is specified by its *structure graph*: the behaviour described through constraints, the known and the unknown variables.
- how *analytical redundancy relations* (ARRs) are calculated with ease for linear or nonlinear systems,
- how *analytical redundancy relations* in structural representation are transformed to *residuals* in analytical form for use in fault diagnosis by algorithmic manipulations on the structure graph (matching and backtracking),
- how to *design residuals* that meet specific fault diagnosis requirements, namely being insensitive to disturbances and can be structured (i.e. sensitive to certain faults and insensitive to others),
- demonstrate structural results for active isolation of possible faults in the structure (violation of constraints) through imposing test signals on inputs to isolate an otherwise non-isolable structural defect,
- discuss possibilities for *reconfiguration* to estimate and to control some variables of interest in case of sensor, actuator or system component failures.

These important properties are found by the analysis of the structure graph and its canonical decomposition. In order to introduce the canonical decomposition, matchings on a bipartite graph are first presented and their interpretation is given. Causality is introduced and adds orientation to the bipartite structure graph. Matching of unknown variables in the structure graph is then investigated and it is shown how ARRs are found among the constraints that are not needed for a particular matching. It is shown how a set of ARRs and the set of constraints through which they are calculated, lead to the important notions of structural detectability and isolability. It is also shown how ARRs generated by structural analysis by design become insensitive to unknown disturbances or to unknown parameters. Further, structural controllability is discussed and fault tolerance is investigated through analysis of the structural properties that exist for reconfiguration of a system in case of component failures. The chapter finally summarises essential design procedures based on the structural analysis methods.

5.2 Structural Model

5.2.1 Structure as a Bipartite Graph

This section introduces the structural model of a system as a bipartite graph which represents the links between a set of variables and a set of constraints. It is an abstraction of the behavioural model, because it merely describes which variables are connected by which constraints, but it does not say how these constraints look like. Hence, the structural model shows the basic features and properties of a system, which are independent of the system parameters.

Behaviour model. The behavioural model of a system is defined by a pair

$$\mathcal{S} = (\mathcal{C},\ \mathcal{Z})$$

where

- $\mathcal{Z} = \{z_1,\ z_2, \ldots, z_N\}$ is a set of variables and parameters and
- $\mathcal{C} = \{c_1,\ c_2, \ldots, c_\mathrm{M}\}$ is a set of constraints.

According to the granularity of the variables (quantitative, qualitative, fuzzy) and of the time (continuous, discrete), the constraints may be expressed in several different forms like algebraic and differential equations, difference equations, rules, etc.

Example 5.1 A differential-algebraic model
Consider the sets

$$\begin{aligned} \mathcal{Z} &= \mathcal{X}_\mathrm{a} \cup \mathcal{X}_\mathrm{d} \cup \mathcal{U} \cup \mathcal{Y} \\ \mathcal{C} &= \{\boldsymbol{g}, \boldsymbol{h}, \boldsymbol{m}\}, \end{aligned}$$

where $\mathcal{X}_\mathrm{a}$ is the set of variables $\boldsymbol{x}_\mathrm{a}$ that appear only in algebraic constraints, and $\mathcal{X}_\mathrm{d}$ the set of variables $\boldsymbol{x}_\mathrm{d}$ whose derivative obeys some differential constraints $\boldsymbol{g}$. A differential-algebraic model is given by

$$\dot{\boldsymbol{x}}_\mathrm{d}(t) = \boldsymbol{g}\left(\boldsymbol{x}_\mathrm{d}(t),\ \boldsymbol{x}_\mathrm{a}(t),\ \boldsymbol{u}(t)\right) \tag{5.1}$$

$$\boldsymbol{0} = \boldsymbol{m}\left(\boldsymbol{x}_\mathrm{d}(t),\ \boldsymbol{x}_\mathrm{a}(t),\ \boldsymbol{u}(t)\right) \tag{5.2}$$

$$\boldsymbol{y}(t) = \boldsymbol{h}\left(\boldsymbol{x}_\mathrm{d}(t),\ \boldsymbol{x}_\mathrm{a}(t),\ \boldsymbol{u}(t)\right). \tag{5.3}$$

Note that it is possible to define a separate set of variables $\dot{\boldsymbol{x}}_\mathrm{d}$ for the derivatives and a separate set of constraints

$$\dot{x}_i(t) - \frac{\mathrm{d}}{\mathrm{d}t} x_i(t) = 0, \quad i = 1, \ldots, n, \tag{5.4}$$

so that the sets of variables and constraints have to be extended:

$$\begin{aligned} \mathcal{Z} &= \mathcal{X}_\mathrm{a} \cup \mathcal{X}_\mathrm{d} \cup \dot{\mathcal{X}}_\mathrm{d} \cup \mathcal{U} \cup \mathcal{Y} \\ \mathcal{C} &= \left\{ \boldsymbol{g}, \boldsymbol{h}, \boldsymbol{m}, \frac{\mathrm{d}}{\mathrm{d}t} \right\}, \end{aligned}$$

where $\frac{\mathrm{d}}{\mathrm{d}t}$ stands for the differential constraints (5.4) and all the constraints (5.1)–(5.3) are algebraic.

The behaviour model of a dynamical system links present and past values of its variables (for discrete time systems) or variables and their time derivatives up to a certain order (for continuous-time systems). Giving two variables the names $\boldsymbol{x}(t)$ and $\dot{\boldsymbol{x}}(t)$ does not guarantee that the second one is the time derivative of the first one. This is only true thanks to the analyst's interpretation, and this fact has to be represented, for automatic treatment, by separate constraints like (5.4). □

Two basic assumptions express the fact that a model defined by some set of constraints is well formed. These assumptions are used in the sequel.

Assumption 5.1

(a) All the constraints in $\mathcal{C}$ are compatible.
(b) All the constraints in $\mathcal{C}$ are independent.

Assumption 5.1(a) means that the set of the constraints is associated with a sound model, namely a model whose set of solutions is not empty. In other words, the constraints do not carry any contradiction.

Assumption 5.1(b) means that the model is minimal in the sense that no constraint defines (at least locally) the same set of solutions as another one, or more generally that in $\mathcal{C}$ there do not exist two different subsets $\mathcal{C}'$ and $\mathcal{C}''$ such that

$$V(\mathcal{C}') \subseteq V(\mathcal{C}'')$$

holds, where $V(\mathcal{C})$ is the set of solutions associated with the constraint set $\mathcal{C}$. It will be seen that this assumption may or may not hold, depending on the redundancy which is present in the system.

Example 5.2 Dependent constraints
Consider the two constraints

$$\begin{aligned} c_1 &: z_1 - 1 = 0 \\ c_2 &: (z_1 - 1)(z_2 - 1) = 0. \end{aligned}$$

They are obviously not independent, since one has $V(c_1) \cap V(c_2) = V(c_1)$. In fact, constraint c_1 is sufficient to describe the set of the system solutions, and one has the implication

$$c_1 \text{ is true } \Rightarrow c_2 \text{ is true.} \square$$

Structure graph. The structure of a system is represented by a bipartite graph. A graph is bipartite if its set of vertices can be separated into two disjoint sets $\mathcal{C}$ and $\mathcal{Z}$ in such a way that every edge has one endpoint in $\mathcal{C}$ and the other one in $\mathcal{Z}$.

Definition 5.1 (*Structural model, structure graph*) The *structural model* of the system $\mathcal{S} = (\mathcal{C},\ \mathcal{Z})$ is a bipartite graph

$$\mathcal{G} = (\mathcal{C},\ \mathcal{Z},\ \mathcal{E}),$$

where $\mathcal{E} \subset \mathcal{C} \times \mathcal{Z}$ is the set of edges defined as follows:

$$(c_i,\ z_j) \in \mathcal{E} \text{ if the variable } z_j \text{ appears in the constraint } c_i.$$

$\mathcal{G}$ is also called the *structure graph* or the *structure*.

In the representation of a system as a pair $\mathcal{S} = (\mathcal{C}, \mathcal{Z})$, the set $\mathcal{C}$ includes the constraints describing the relations among the variables, whereas the vertex set $\mathcal{C}$ of the graph $\mathcal{G}$ includes only the names of these constraints, which are used as the names of vertices. Nevertheless, the same symbol $\mathcal{C}$ is used in $\mathcal{S}$ and $\mathcal{G}$.

The bipartite graph is an undirected graph, which can be interpreted as follows: All the variables and parameters $z_j \in \mathcal{Z}$ that are connected with a given constraint-vertex $c_i \in \mathcal{C}$ have to satisfy the equation or rule that this constraint-vertex represents. The structure graph can be built for rather general models including models of the form of differential and algebraic equations.

In the following figures, the variable-vertices $z_j \in \mathcal{Z}$ will be represented by circles while the constraint-vertices $c_i \in \mathcal{C}$ will be represented by bars. Note that the edges are not oriented. The incidence matrix of the bipartite graph is used to represent the graph as a set $\mathcal{E}$ of edges in an algebraic manner. The rows of this matrix are associated with the constraints and the columns with the variables. A "1" in the intersection of row c_i and column z_j indicates the existence of the edge $\left(c_i,\ z_j\right) \in \mathcal{E}$. For an example, cf. (5.5).

Example 5.3 Structure graph of a linear system
Consider a linear system described by four constraints $\{c_1,\ c_2,\ c_3,\ c_4\}$ with five variables $\{x_1,\ x_2,\ \dot{x}_1,\ \dot{x}_2,\ u\}$ as follows:

$$\begin{aligned}
c_1 &: \dot{x}_1 = \frac{\mathrm{d}x_1}{\mathrm{d}t} \\
c_2 &: \dot{x}_1 = ax_2 \\
c_3 &: \dot{x}_2 = \frac{\mathrm{d}x_2}{\mathrm{d}t} \\
c_4 &: \dot{x}_2 = bx_1 + cx_2 + du.
\end{aligned}$$

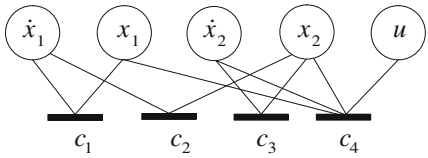


Fig. 5.1 Bipartite graph of the linear system

Its structure graph has the incidence matrix

↗	u	x_1	x_2	$\dot{x}_1$	$\dot{x}_2$
c_1		1		1	
c_2			1	1	
c_3			1		1
c_4	1	1	1		1

(5.5)

leading to the bipartite graph depicted in Fig. 5.1. □

Example 5.4 Tank system
Consider a tank system where the inflow $q_i(t)$ is controlled via a level sensor and an electric pump and the outflow $q_o(t)$ is realised through an output pipe (Fig. 5.2).

The system consists of the components {tank, input valve, output pipe, level sensor, level control algorithm}. A continuous-variable continuous-time model is given by the following constraints:

$$\begin{aligned}
\text{Tank } c_1 &: \dot{h}(t) = q_i(t) - q_o(t) \\
\text{Input valve } c_2 &: q_i(t) = \alpha u(t) \\
\text{Output pipe } c_3 &: q_0(t) = k\sqrt{h(t)} \\
\text{Level sensor } c_4 &: y(t) = h(t) \\
\text{Control algorithm } c_5 &: u(t) = \begin{cases} 1 \text{ if } y(t) \le h_0 - r \\ 0 \text{ if } y(t) \ge h_0 + r. \end{cases}
\end{aligned} \tag{5.6}$$

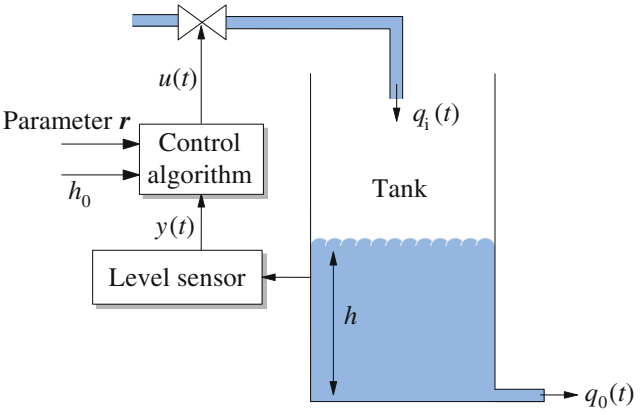


Fig. 5.2 Single-tank system

u is the control variable, y the sensor output, h_0 the given set point, and r and k are given parameters. h denotes the liquid level, q_i and q_o the flow into or out of the tank. α is a valve constant. Each component introduces one constraint. The separate constraint

$$c_6 : \dot{h}(t) = \frac{\mathrm{d}h(t)}{\mathrm{d}t}$$

expresses the fact that $\dot{h}(t)$ is the derivative of the level $h(t)$.

This behavioural model of the tank system without controller leads to the structure graph with the following incidence matrix:

	Input/Output		Internal variables			
↗	$u(t)$	$y(t)$	$h(t)$	$\dot{h}(t)$	$q_i(t)$	$q_o(t)$
c_1				1	1	1
c_2	1				1	
c_3			1			1
c_4		1	1			
c_6			1	1		

Every column of the matrix corresponds to a circle-vertex and every row to a bar-vertex. The structure graph is shown in Fig. 5.3.

If the controller is introduced, the graph is extended by a new bar-vertex for c_5 and two new circle-vertices for h_0 and r. Furthermore, if the parameter k appearing in constraint c_3 is considered now as an important variable (rather than a fixed given parameter like the valve constant α), a circle-vertex is introduced for k and linked with c_3. These steps lead to the following extended incidence matrix:

	Input/Output		Parameters			Internal variables			
↗	$u(t)$	$y(t)$	h_0	r	k	$h(t)$	$\dot{h}(t)$	$q_i(t)$	$q_o(t)$
c_1							1	1	1
c_2	1							1	
c_3					1	1			1
c_4		1				1			
c_5	1	1	1	1					
c_6						1	1		

For simplicity, only the ones appear in this matrix and empty boxes are zero. Figure 5.4 shows the extended graph. □

Remark 5.1 (*Structural representation by digraphs*) For nonlinear systems

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \\ \boldsymbol{y}(t) &= \boldsymbol{g}(\boldsymbol{x}(t)) \end{aligned}$$

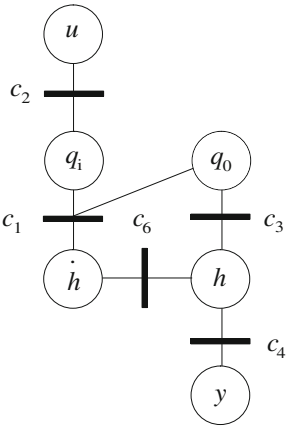


Fig. 5.3 Structure graph of the single-tank system without controller

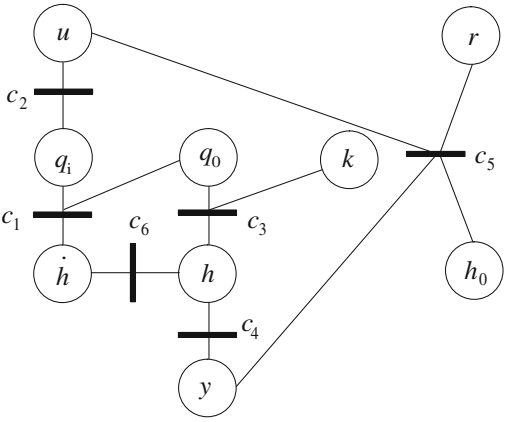


Fig. 5.4 Structure graph of the controlled tank

a popular structural representation uses the directed graph (digraph), whose set of vertices is the set of the input, output and state variables and whose edges are defined by the following rules:

- An edge exists from vertex x_k (resp. from vertex u_1) to vertex x_i if and only if the state variable x_k (resp. the input variable u_1) really occurs in function f_i (i.e. $\frac{\partial f_i}{\partial x_k}$—resp. $\frac{\partial f_i}{\partial u_1}$—is not identically zero).
- An edge exists from vertex x_k to vertex y_j if and only if the state variable x_k really occurs in the function g_j.

In the digraph representation edges are interpreted as "mutual influences" between variables: an edge from x_k to x_i means that the time evolution of the derivative $\dot{x}_i(t)$ depends on the time evolution of $x_k(t)$. Similarly, an edge from x_k to y_j means that the time evolution of the output $y_j(t)$ depends on the time evolution of the state variable $x_k(t)$. In contrast to the bipartite graph, the signals x_i and $\dot{x}_i$ are not distinguished but represented by the same vertex x_i. □

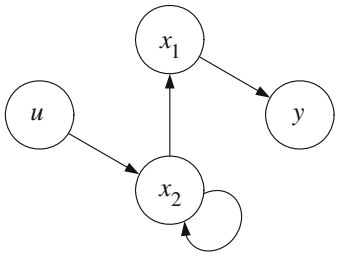


Fig. 5.5 Digraph of the linear system

Example 5.5 Digraph of a linear system
The digraph which describes the structure of the system

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= ax_2(t) + bu(t) \\ y(t) &= x_1(t) \end{aligned} \tag{5.7}$$

is shown in Fig. 5.5. Obviously, the constraints given in the behavioural model are not explicitly represented. □

5.2.2 Subsystems

Instead of considering the whole set of constraints which describe the behavioural model of a system, it is sometimes convenient to consider only subsets of constraints. A subsystem is defined by the set of constraints together with the set of variables that occur in these constraints. This subsection introduces the vocabulary connected with subsets of the constraints.

The symbol $2^{\mathcal{C}}$ denotes the set of all the subsets of $\mathcal{C}$ (also denoted as the power set of $\mathcal{C}$). Let $\mathcal{G} = (\mathcal{C}, \mathcal{Z}, \mathcal{E})$ be the structure graph of the system $\mathcal{S} = (\mathcal{C}, \mathcal{Z})$ and Q be a mapping between a set of constraints and the set of variables used in these constraints:

$$\begin{aligned} Q: \quad & 2^{\mathcal{C}} \to 2^{\mathcal{Z}} \\ & \phi \mapsto Q(\phi) = \{z \in \mathcal{Z};\ \exists c \in \phi \quad \text{s.t. } (c, z) \in \mathcal{E}\}. \end{aligned} \tag{5.8}$$

Q associates with any subset ϕ of constraints, the subset $Q(\phi)$ of those variables which intervene in at least one of them. Correspondingly, the mapping R associates a set of variables with a set of constraints where these variables appear:

$$\begin{aligned} R: \ & 2^{\mathcal{Z}} \to 2^{\mathcal{C}} \\ & \xi \mapsto R(\xi) = \{c \in \mathcal{C};\ \exists z \in \xi \quad \text{s.t. } (c, z) \in \mathcal{E}\}. \end{aligned} \tag{5.9}$$

Definition 5.2 (*Subsystem*) For a system $\mathcal{S} = (\mathcal{C}, \mathcal{Z})$, a subsystem is a pair $(\phi, Q(\phi))$ with $\phi \in 2^{\mathcal{C}}$. The subgraph that is related with subsystem $(\phi, Q(\phi))$ represents the subsystem structure.

According to this definition, a subsystem is any subset ϕ of the system constraints together with the set $Q(\phi) \subset \mathcal{Z}$ of related variables. There are no specific requirements on the choice of the elements in $\phi \subseteq 2^{\mathcal{C}}$. Of course, only some of them are of interest in applications:

- First, subsystems can be associated with some *physical interpretation*. Complex systems are often decomposed into subsystems which have a physical or a functional meaning. For example, a boiler can be decomposed into a steam generator, the instrumentation scheme and a control system. These subsystems are associated with subsets of constraints, so that the fault of one or several subsystems results in some of these constraints being changed.

- Second, subsystems can be associated with *special properties*. For example, fault diagnosis is possible only for subsystems which exhibit redundancy properties as shown later.

Example 5.6 Q **and** R **mappings for the tank example**
Consider the following incidence matrix for the single-tank system.

$\nearrow$	h	$\dot{h}$	q_i	q_o	u	y
c_1		1	1	1		
c_2			1		1	
c_3	1			1		
c_4	1					1
c_5					1	1
c_6	1	1				

Examples for the mappings Q and R are:

$$\begin{aligned} Q(\{c_1, c_3\}) &= \{h, \dot{h}, q_i, q_o\} \\ Q(\{c_5\}) &= \{u, y\} \\ R(\{q_i, q_o\}) &= \{c_1, c_2, c_3\}. \end{aligned}$$

Hence, the pair

$$(\{c_1, c_3\}, \{h, \dot{h}, q_i, q_o\})$$

is a subsystem and its structure is described by the subgraph with the incidence matrix

$\nearrow$	h	$\dot{h}$	q_i	q_o
c_1		1	1	1
c_3	1			1

□

5.2.3 Structural Properties

Two systems which have the same structure are said to be *structurally equivalent*. Consequently, the structure graph $\mathcal{G}$ defines a class $\mathcal{S}(\mathcal{G})$ of structurally equivalent systems. In particular, systems which only differ by the value of their parameters belong to the same class.

The class of systems defined by the structure graph is large, because the structure is independent of the form in which the constraints are expressed. For example, suppose that the level sensor in the single-tank system does not provide an analog output but a quantised one. Then its operation is described by the following table, where α, β, γ are given constants:

h	$\in [0, \alpha[$	$\in [\alpha, \beta[$	$\in [\beta, \gamma[$	$\geq \gamma$
y	empty	low	medium	high

For structural considerations the important information included in this table is the fact that the sensor reading y and the tank level h are connected and, hence, in the structure graph there exist edges between the variable-vertices for y and h towards the constraint-vertex for the sensor. This fact is obviously independent of the quantisation. Hence, the structure of the sensor is exactly the same for analog and for symbolic sensor readings.

Structural properties are properties of the system class $\mathcal{S}(\mathcal{G})$ rather than of a single system $\Sigma \in \mathcal{S}(\mathcal{G})$, because they are properties of the graph $\mathcal{G}$. The relation between the results of the structural analysis and the results of a numerical analysis of a single system is depicted in Fig. 5.6. The arrows from the left to the right part of the figure show the abstraction process, which leads from the numerical values of the system parameters to the links among the variables represented by the structure of the constraints. Accordingly, the properties of the system class are abstractions of properties of the (numerical) systems that are structurally equivalent.

As the aim of structural analysis is to elaborate properties that belong to the graph $\mathcal{G}$, but are relevant for all or at least for most of the systems $\Sigma \in \mathcal{S}(\mathcal{G})$, the analysis usually concerns two similar properties P and P', where P is a property defined for a single system Σ and P' is a property of the graph $\mathcal{G}$. With respect to the figure, one has to ensure that the "inverse abstraction" process from the properties of the class $\mathcal{S}(\mathcal{G})$ towards the single system Σ is known. For most of the structural properties P' that are investigated in structural analysis there exists a property P such that the following relation holds:

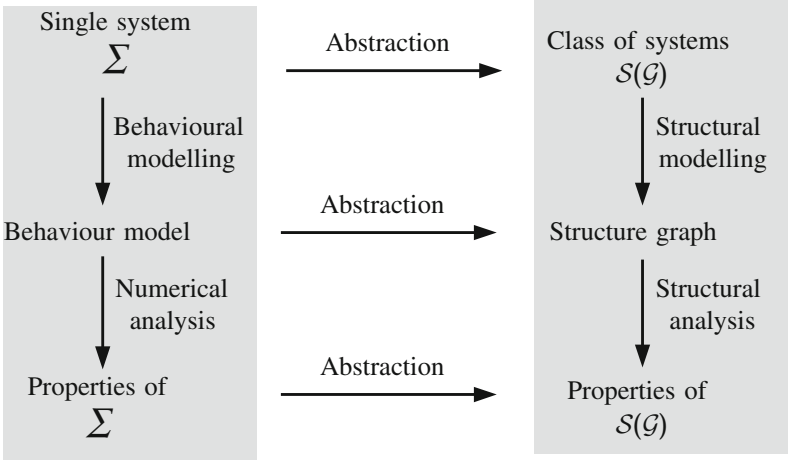


Fig. 5.6 Numerical and structural analysis of dynamical systems

If the system $\Sigma \in \mathcal{S}(\mathcal{G})$ has the property P, then the system class $\mathcal{S}(\mathcal{G})$ has the property P'.

Hence, the requirement that the graph $\mathcal{G}$ possesses the property P' is a necessary condition for the system $\Sigma \in \mathcal{S}(\mathcal{G})$ to have the property P.

Example 5.7 Observability and structural observability
Consider the static system

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \underbrace{\begin{pmatrix} a(\boldsymbol{\theta}) & c(\boldsymbol{\theta}) \\ b(\boldsymbol{\theta}) & d(\boldsymbol{\theta}) \end{pmatrix}}_{\boldsymbol{A}(\boldsymbol{\theta})} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \tag{5.10}$$

for which the internal variables x_1 and x_2 should be determined for measured outputs y_1 and y_2. Every single system Σ is characterised by Eq. (5.10) together with a parameter vector $\boldsymbol{\theta} \in |\mathcal{R}^q$. The system Σ is said to be *observable* if the model (5.10) can be used to determine x_1 and x_2 in terms of y_1 and y_2, which is obviously the case if and only if the matrix $\boldsymbol{A}(\boldsymbol{\theta})$ is invertible. Hence,

$$\operatorname{rank} \boldsymbol{A} = n \tag{5.11}$$

is a necessary and sufficient condition for the observability of the system Σ, where n is the number of unknown variables to be observed. For observability analysis, which concerns the left arrow "Numerical analysis" in Fig. 5.6, Eq. (5.11) has to be checked.

On the other hand, a structural analysis abstracts from the parameter values and uses a graph with the incidence matrix

$$\boldsymbol{E} = \begin{array}{|c||c|c|} \hline \nearrow & x_1 & x_2 \\ \hline\hline c_1 & [a(\boldsymbol{\theta})] & [c(\boldsymbol{\theta})] \\ \hline c_2 & [b(\boldsymbol{\theta})] & [d(\boldsymbol{\theta})] \\ \hline \end{array}$$

where c_1 and c_2 denote the first and the second equation in (5.10) and the symbol $[\,\cdot\,]$ denotes the qualitative value of the matrix element considered. The qualitative value $[a(\boldsymbol{\theta})]$ is equal to 1 if the argument $a(\boldsymbol{\theta})$ does not vanish for *all* parameter vectors $\boldsymbol{\theta} \in |\mathcal{R}^q$; otherwise it is zero.

The system class $\mathcal{S}(\mathcal{G})$ includes all systems Σ described by Eq. (5.10) for arbitrary parameter vectors that are consistent with the entries of the matrix $\boldsymbol{E}$. Hence, if $\boldsymbol{E}$ has a vanishing element, the corresponding element of the matrix $\boldsymbol{A}$ vanishes for all $\Sigma \in \mathcal{S}(\mathcal{G})$. This system class is said to be *structurally observable* if at least one system $\Sigma \in \mathcal{S}(\mathcal{G})$ exists that is observable (according to the definition given above). That is, there has to exist at least one parameter vector $\boldsymbol{\theta}$ for which the relation $\det \boldsymbol{A} \neq 0$ holds. This is obviously the case if and only if the structural rank of the matrix $\boldsymbol{E}$ is two or, more generally,

$$\text{s-rank}\, \boldsymbol{E} = n. \tag{5.12}$$

The structural rank of a matrix $\boldsymbol{E}$ is the maximum number of non-zero elements in different rows and columns of $\boldsymbol{E}$. The arrow "structural analysis" in Fig. 5.6 means to test Eq. (5.12).

The important aspect of this example is the fact that for the single system the notion of observability and for a class of systems the notion of structural observability has been defined in such a way that a system has to belong to a structurally observable system class if the system should be observable. Both definitions are closely related to one another, but these properties are not the same! The structural observability of the system class, which can be tested by the graph, is a necessary condition for the observability of the system.

However, to belong to a structurally observable system class is not sufficient for a system to be observable. Think of the system Σ with $a = b = c = d = 1$. This system violates the condition for observability (as $\det \boldsymbol{A} = 0$) although it belongs to a structurally observable system class. An important aspect of the structural investigations in the next sections concerns the relations among these properties, in particular, the elaboration of conditions under which the structural properties of a system class do not transfer to the (numerical) properties of every single system of this class. For the observability properties considered in this example, this condition is given by

$$\text{s-rank}\, \boldsymbol{E} \geq \text{rank}\, \boldsymbol{A},$$

which means that the rank of a matrix cannot exceed the structural rank of the graph that represents the structure of this matrix. □

An important question asks under what conditions the structural property P' of $\mathcal{S}(\mathcal{G})$ does *not* transfer to the numerical property P of $\Sigma \in \mathcal{S}(\mathcal{G})$. Two cases can be distinguished with respect to the example above:

1. In the first case, parameters $\boldsymbol{\theta}$ *always* satisfy the relation $\det \boldsymbol{A} = 0$ and thus the structural property is *never* translated into an actual property. This situation is excluded in structural analysis, because the parameters are always supposed to be independent, which means that they span the whole space $|\mathcal{R}^q$. As an algebraic relation like $\det \boldsymbol{A} = 0$ defines a manifold in the parameter space $|\mathcal{R}^q$, it cannot be satisfied by all $\boldsymbol{\theta} \in |\mathcal{R}^q$. Otherwise, the equation $\det \boldsymbol{A} = 0$ should have been included in the system model.

2. In the second case, the parameters $\boldsymbol{\theta}$ *of the system under investigation* satisfy the relation $\det \boldsymbol{A} = 0$, and thus the structural property is not translated into an

actual property *for that particular system*. Structural analysis, however, implies the interesting conclusion that under mild assumptions on the functions $a,\ b,\ c,\ d$ there always exists a parameter vector $\tilde{\boldsymbol{\theta}}$ in the neighbourhood of $\boldsymbol{\theta}$ for which the actual property coincides with the structural one.

In conclusion, (numerical) properties P can only occur if the corresponding structural properties P' are satisfied. They can certainly not be true if the structural properties are not satisfied. Furthermore:

> Structural properties are properties which hold for actual systems almost everywhere in the space of their independent parameters.

Hence, it is extremely unlikely that the system under consideration has a parameter vector for which a structural property does not imply the corresponding numerical property.

5.2.4 Known and Unknown Variables

The system variables and parameters can be classified as known and unknown ones. The system inputs and outputs are examples of variables that are usually known. Similarly, model parameters which have been previously identified are known. Unknown variables are not directly measured, though there might exist some way to compute their value from the values of known ones. In the tank example, the last four columns of the incidence matrix $\{h,\ \dot{h},\ q_i,\ q_o\}$ correspond to unknown variables, while the first five ones correspond to known variables and parameters $\{u,\ y,\ h_0,\ r,\ k\}$.

Following that decomposition, the set of the variables is partitioned into

$$\mathcal{Z} = \mathcal{K} \cup \mathcal{X},$$

where $\mathcal{K}$ is the subset of the known variables and parameters and $\mathcal{X}$ is the subset of the unknown ones. Similarly, the set of constraints is partitioned into

$$\mathcal{C} = \mathcal{C}_{\mathcal{K}} \cup \mathcal{C}_{\mathcal{X}},$$

where $\mathcal{C}_{\mathcal{K}}$ is the subset of those constraints which link only known variables and $\mathcal{C}_{\mathcal{X}}$ includes those constraints in which at least one unknown variable appears. $\mathcal{C}_{\mathcal{K}}$ is the largest subset of constraints such that $Q(\mathcal{C}_{\mathcal{K}}) \subseteq \mathcal{K}$. Obviously, the relations defining control algorithms belong to $\mathcal{C}_{\mathcal{K}}$ because they introduce constraints among the sensor output, the control objectives (set points, tracking references, final states) and the control input, which are all known variables.

According to the partition of $\mathcal{Z}$ and $\mathcal{C}$, the graph $\mathcal{G} = (\mathcal{C},\ \mathcal{Z},\ \mathcal{E})$ can be decomposed into two subgraphs which correspond to the two subsystems $(\mathcal{C}_{\mathcal{K}},\ Q(\mathcal{C}_{\mathcal{K}}))$ and $(\mathcal{C}_{\mathcal{X}},\ \mathcal{Z})$. The behavioural model of the subsystem $(\mathcal{C}_{\mathcal{K}},\ Q(\mathcal{C}_{\mathcal{K}}))$ involves only known variables. In some further developments, it will be of interest to focus on the

subsystem $(\mathcal{C}_{\mathcal{X}},\ \mathcal{Z})$ which leads to the *reduced structure graph*. This graph includes only those constraints that refer to at least one unknown variable $z_i \in \mathcal{X}$.

A fundamental question of fault diagnosis concerns the determination of unknown variables from known variables by means of constraints. The question whether this is possible or not depends only upon the structure of the subgraph $(\mathcal{C}_{\mathcal{X}},\ \mathcal{X},\ \mathcal{E}_{\mathcal{X}})$ that results from the complete structure graph $\mathcal{G}$ by deleting all known variables $z_i \in \mathcal{K}$ together with the corresponding edges. Therefore, in all further examples of structure graphs the known variables are marked grey.

Example 5.8 Analysis of the structure graph of the tank system
Consider the tank, whose structure graph is given in Fig. 5.4. Assume that the input u and the output y are known signals and, furthermore, h_0, r and k are known parameters. Then the decomposition of the variable set

$$\mathcal{Z} = \{h,\ \dot{h},\ q_i,\ q_o,\ u,\ y,\ h_0,\ r,\ k\}$$

into set of known and unknown variables yields the sets

$$\mathcal{K} = \{u,\ y,\ h_0,\ r,\ k\} \quad \text{and} \quad \mathcal{X} = \{h,\ \dot{h},\ q_i,\ q_o\}.$$

By selecting all constraints whose variables are all in the set $\mathcal{K}$, the set $\mathcal{C}_{\mathcal{K}} = \{c_5\}$ is obtained. All other constraints are comprised in the set

$$\mathcal{C}_{\mathcal{X}} = \{c_1,\ c_2,\ c_3,\ c_4,\ c_6\}.$$

Obviously, $Q(\mathcal{C}_{\mathcal{K}}) \subseteq \mathcal{K}$ and

$$Q(\mathcal{C}_{\mathcal{X}}) = \{u,\ y,\ q_i,\ q_o,\ h,\ \dot{h}\}$$

hold. The incidence matrix of the structure graph can be reorganised as follows:

	known					unknown			
$\nearrow$	u	y	h_0	r	k	h	$\dot{h}$	q_i	q_o
c_5	1	1	1	1					
c_1							1	1	1
c_2	1							1	
c_3					1	1			1
c_4		1				1			
c_6						1	1		

The known variables are in the left columns and the constraint that refers merely to known variables in the first row. The reduced structure graph which corresponds to the subsystem $(\mathcal{C}_{\mathcal{X}},\ \mathcal{Z})$ is given by the lower part of the incidence matrix. As the variables h_0 and r do not appear in this part of the matrix, their columns are deleted. Hence, the reduced structure graph has the incidence matrix:

	known			unknown			
↗	u	y	k	h	$\dot{h}$	q_i	q_o
c_1					1	1	1
c_2	1					1	
c_3			1	1			1
c_4		1		1			
c_6				1	1		

The reduced graph is shown in Fig. 5.7a.

For diagnosis, another decomposition of the variables into known and unknown ones is used. The parameters like k, h_0 and r are assumed to be fixed and, hence, ignored in the structure graph. The remaining variables represent signals, some of which are measured and the others are unknown. Hence, for the tank system the fixed parameter k is deleted from the structure graph, which results in the following incidence matrix and in the graph depicted in Fig. 5.7b:

↗	u	y	h	$\dot{h}$	q_i	q_o
c_1				1	1	1
c_2	1				1	
c_3			1			1
c_4		1	1			
c_6			1	1		

The diagnosis problem for the tank system can be posed as the problem to decide whether the "grey signals" u and y are consistent with the model whose structure is shown in Fig. 5.7b. □

5.3 Matching in Bipartite Graphs

The basic tool for the structural analysis is the concept of matching in bipartite graphs, which is introduced in this section. In loose terms, a matching is a causal assignment which associates with every unknown system variable a constraint that can be used to determine the variable. Unknown variables which do not appear in a matching cannot be calculated. Variables which can be matched in several ways can be determined in different (redundant) ways. The last situation provides a means for fault detection and for reconfiguration.

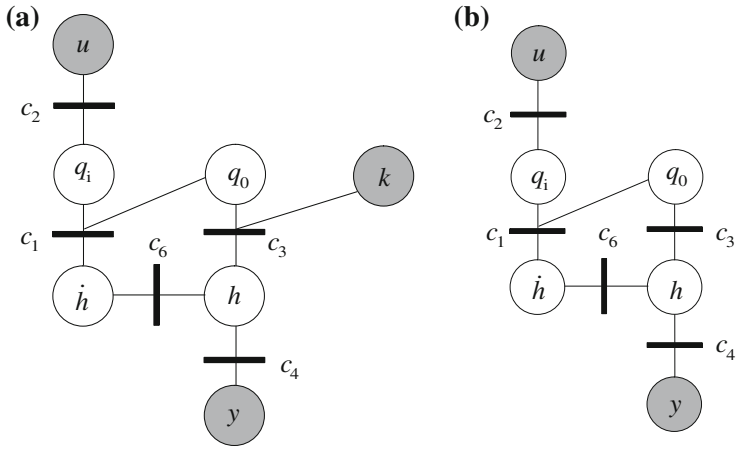


Fig. 5.7 Reduced structure graph of the tank system (**a**) and structure graph used in diagnosis (**b**)

5.3.1 Definitions

Matching is a general notion that has been introduced for bipartite graphs. It is introduced here in general terms for bipartite graphs $\mathcal{G} = (\mathcal{C},\ \mathcal{Z},\ \mathcal{E})$, but illustrated for structure graphs of dynamical systems.

Edges of the graph $\mathcal{G}$ are said to be *disjoint* if they have no vertex in common (neither in $\mathcal{C}$ nor in $\mathcal{Z}$).

Definition 5.3 (*Matching*) A matching $\mathcal{M} \subseteq \mathcal{E}$ is a set of disjoint edges of a bipartite graph $\mathcal{G}$.

In general, different matchings can be defined on a given bipartite graph as illustrated in Fig. 5.8 by the bold edges. These matchings are given by the following set of disjoint edges:

$$\mathcal{M}_1 = \{(c_1, \dot{h}),\ (c_2, q_i),\ (c_3, h),\ (c_4, y)\}$$
$$\mathcal{M}_2 = \{(c_6, \dot{h}),\ (c_1, q_i),\ (c_4, h),\ (c_3, q_o)\}.$$

The examples show that a set $\mathcal{M}$ of edges is called matching even if it does not include a maximum number of disjoint edges. If it does, it is said to be a maximum matching.

A maximum matching is hence a matching such that no edge of the graph $\mathcal{G}$ can be added without violating the requirement that the edges have to be disjoint. Since the set of matchings $\boldsymbol{M}$ is only partially ordered, it follows that there is in general more than one maximum matching. The "size" of a matching $\mathcal{M}$ is its cardinality $|\mathcal{M}|$. In general, the relation

$$|\mathcal{M}| \leq \min\{|\mathcal{C}|\,,\ |\mathcal{Z}|\}$$

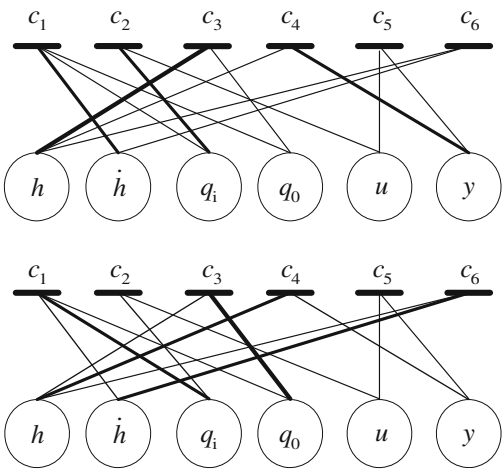


Fig. 5.8 Two matchings for the tank system: The edges $e \in \mathcal{M}$ are drawn by thick lines

holds. The maximum cardinality over the set of matchings is called the *matching number* and is denoted by

$$\nu(\mathcal{G}) = \max_{\mathcal{M} \in \boldsymbol{M}} |\mathcal{M}|.$$

In the incidence matrix, a matching is represented by selecting at most one "1" in each row and in each column and marking it by "①". Each ① represents an edge of the matching. No other edge should contain the same variable (thus it is the only one in the row) or the same constraint (it is the only one in the column). The set $\boldsymbol{M}$ of all matchings of a graph is a subset of $2^{\mathcal{E}}$.

Structural analysis deals with matchings that include all vertices $c \in \mathcal{C}$ or all vertices $z \in \mathcal{Z}$.

Definition 5.4 (*Complete matching*) A matching is called complete with respect to $\mathcal{C}$ if $|\mathcal{M}| = |\mathcal{C}|$ holds. A matching is called complete with respect to $\mathcal{Z}$ if $|\mathcal{M}| = |\mathcal{Z}|$ holds.

For a matching $\mathcal{M}$ that is complete with respect to $\mathcal{C}$, each constraint belongs to exactly one edge of the matching:

$$\forall c \in \mathcal{C}: \quad \exists z \in \mathcal{Z} \text{ such that } (c, z) \in \mathcal{M}.$$

Similarly, for a matching that is complete with respect to $\mathcal{Z}$, every variable belongs to an edge:

$$\forall z \in \mathcal{Z}: \quad \exists c \in \mathcal{C} \text{ such that } (c, z) \in \mathcal{M}.$$

Structural analysis is mainly concerned with $\mathcal{Z}$-complete matchings, because such matchings show a way how to determine all unknown variables of the system.

It is useful to define matchings, maximum matchings and complete matchings by considering either the whole structure of the system or only subgraphs which refer to subsets of the constraint set and the variable set. Since only unknown variables in $\mathcal{X}$ need be determined by a constraint, variables in $\mathcal{K}$ like control input and measurements are already known, the matching can be accomplished for the reduced structure graph containing all unknown variables rather than for the whole structure graph. As the incidence matrices and the graphical representations are given for the complete graph, and since backtracking to known variables is needed at a later stage in order to obtain residuals, matchings are preferably done using the complete structure graph, but we can illustrate some properties of matching by considering the reduced structure graph.

Example 5.9 Matchings on the reduced structure graph of the tank system
To illustrate the notion of maximum and complete matchings, consider the reduced structure graph of the single-tank system. Only the unknown signals and the constraints among them are concerned with. The edges of a matching are identified by a thick line in the drawings and by "①" in the incidence matrices.

↗	h	$\dot{h}$	q_i	q_o
c_1		1	①	1
c_2			1	
c_3	①			1
c_4	1			
c_6	1	①		

Matching (a)

↗	h	$\dot{h}$	q_i	q_o
c_1		1	1	1
c_2			①	
c_3	1			①
c_4	①			
c_6	1	①		

Matching (b)

↗	h	$\dot{h}$	q_i	q_o
c_1		①	1	1
c_2			①	
c_3	1			①
c_4	1			
c_6	①	1		

Matching (c)

As in the matchings unknown variables are associated with a constraint by means of which they can be determined, an intuitive graphical representation of the matchings is given in Fig. 5.9 where the constraints are drawn on the left-hand side and the variables on the right-hand side. The thick edges indicate the matching. The graphs are the same as in Fig. 5.7b.

Figure 5.9a shows an incomplete matching. It is not complete with respect to the constraints because constraints c_2 and c_4 are not matched, nor is it complete with respect to the variables because q_o is not matched. However, no edge can be added to the matching without violating Definition 5.3.

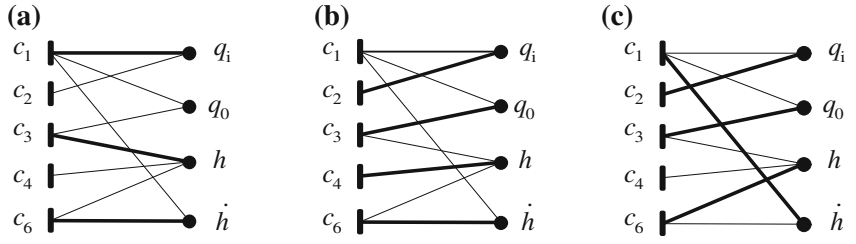


Fig. 5.9 An incomplete matching (**a**) and two matchings (**b**), (**c**) that are complete with respect to $\mathcal{Z}$

Two complete matchings with respect to the set of unknown variables are shown in Fig. 5.9b, c. There is no matching that is complete with respect to $\mathcal{C}_{\mathcal{X}}$, because the number of constraints is larger than the number of variables. Note that it is not guaranteed for the structure graphs that a complete matching exists, neither with respect to $\mathcal{C}_{\mathcal{X}}$ nor with respect to $\mathcal{X}$. □

5.3.2 Oriented Graph Associated with a Matching

Defining a matching on a structure graph introduces some orientations of the edges which, until now, were undirected. Constraints which appear in the system description have no direction, because all variables have the same status. For example, the tank constraint

$$c_1 : q_i(t) - q_o(t) - \dot{h}(t) = 0 \tag{5.13}$$

can be used to compute any of the three variables whenever the two other variables are known. It is written in the non-oriented form to stress that the constraint itself has no preference for any of the three variables. Once a matching is chosen, this symmetry is broken, because each matched constraint is now associated with one matched variable and some non-matched ones.

For a given constraint, matched and non-matched variables are identified in the graph incidence matrix by ① or 1, respectively. For example, according to the matching in Fig. 5.9a, the constraint c_1 described by Eq. (5.13) is used to compute $q_i(t)$. This interpretation of a matching as a set of constraints that can be used to determine the value of unknown variables is valid if there exists an order, in which the constraints can be used for such a calculation. However, a matching may result in an "algebraic loop" (Fig. 5.15), which will be discussed in more detail later, where several constraints together define the value of a set of variables. Then the interpretation of a matching as a correspondence between unknown variables and constraints that can be used to determine the variables is valid only for the set of variables and the set of constraints rather than for single variables and single constraints.

In the graphical representation, the unsymmetries associated with a matching are represented by transforming the originally non-oriented edges into oriented ones. Since some constraints might not be matched, the following rules are applied:

- **Matched constraints**: The edges adjacent to a matched constraint are provided with an orientation
  - from the non-matched (input) variables towards the constraint,
  - from the constraint towards the matched (output) variable (Fig. 5.10a).
- **Non-matched constraints**: All the variables are considered as inputs and, hence, all edges are oriented from the variables to the constraint (Fig. 5.10b).

To understand the reason for these rules, consider a matching $\mathcal{M}$ and choose an edge $(c, x) \in \mathcal{M}$. Then the variable x can be considered as the output of the

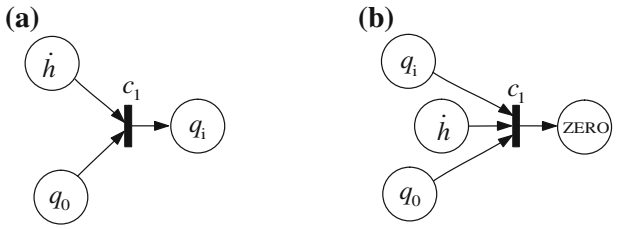


Fig. 5.10 Matched (**a**) and a non-matched constraint (5.13) (**b**)

constraint c while the other variables appearing in the set $Q(c)\backslash\{x\}$ are the inputs.[1] The interpretation is that the matching represents some causality assignment by which the constraint c is used to compute the variable x assuming the other variables to be known. An explicit representation of the constraint c that can be used to determine x is denoted by

$$x = \gamma\left(Q\left(c\right)\backslash\{x\}\right).$$

For non-matched constraints, all variables are considered as inputs and no variable of $Q(c)$ can be considered as an output. Hence, the constraint can be written in the form

$$\gamma\left(Q(c)\right) = 0$$

like Eq. (5.13). If the zero on the right-hand side is considered as output, the constraint can be associated with a ZERO vertex like in Fig. 5.10b. Using no label at all is considered as an implicit ZERO label.

Example 5.10 Determination of unknown variables of the tank system
For the single-tank system, the reduced graph shown in Fig. 5.7b and the three matchings shown in Fig. 5.9 yield the oriented graphs depicted in Fig. 5.11. The directed edges show how the internal variables q_i, q_o, h and $\dot{h}$ can be determined for known values of u and y.

As Matching 1 is incomplete, the unknown variable q_o cannot be computed as shown in the graph. Matchings 2 and 3 are complete with respect to $\mathcal{X}$ but incomplete with respect to $\mathcal{C}_{\mathcal{X}}$. The non-matched constraint c_1 or c_4, respectively, leads to a ZERO output, that is, they have to hold for the variables q_i and $\dot{h}$ or h and y that have been determined by other constraints or have been measured, respectively. □

Note that subgraphs whose input and output nodes are all known provide the system input–output relations. By using Matching 2 in Fig. 5.11 the two following input–output relations are found. The first one is provided by the constraint c_5, which links only known variables and is, therefore, deleted when drawing the reduced graph. The second one results from the non-matched constraint c_1

[1] In Eq. (5.8), Q has been defined as a mapping from the power set of the set of constraints towards the power set of the set of variables. It is used here also for a single constraint c where for notational convenience $Q(\{c\})$ is written as $Q(c)$.

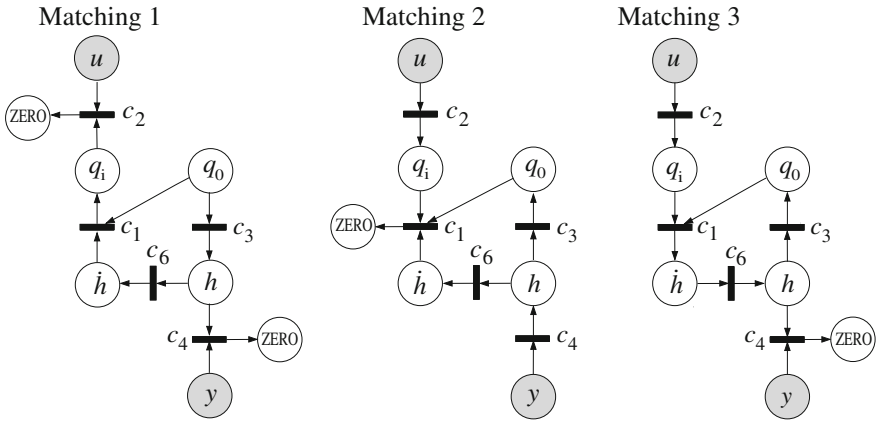


Fig. 5.11 Directed graphs corresponding to the three matchings

$$c_5\,(u,\,y) = 0$$
$$c_1\,(u,\,\gamma_1\,(\gamma_3\,(k,\,\gamma_4(y)),\,\gamma_6\,(\gamma_4(y)))) = 0,$$

where $\gamma_i(z)$ denotes the output of constraint c_i for the input z.

Alternated chains and reachability. The oriented graph that is obtained from the causal interpretation of the structure graph with a matching has the following property: Any existing path between two nodes (variables or constraints) alternates successively variables and constraints nodes. Such a path is called an *alternated chain*. Its length is the number of constraints that are crossed along the path. Note that if a non-matched constraint belongs to an alternated chain, the chain ends with the ZERO variable that is associated with the non-matched constraint.

Alternated chains can be used to define the notion of reachability.

Definition 5.5 (*Reachability*) A variable z_2 is reachable from a variable z_1 if there exists an alternated chain from z_1 to z_2. A variable z_2 is reachable from a subset $\chi \subseteq \mathcal{Z} \setminus \{z_2\}$ of variables if there exists some variable $z_1 \in \chi$ such that z_2 is reachable from z_1. A subset $\mathcal{Z}_2$ of variables is reachable from a subset $\mathcal{Z}_1$ of variables if any variable of $\mathcal{Z}_2$ is reachable from some variable of $\mathcal{Z}_1$.

Example 5.11 Alternated chains in the tank system
Alternated chains associated with the oriented graph of the tank system are the following:

$$y - c_4 - h - c_3 - q_0 - c_1 - q_i$$
$$h - c_6 - \dot{h} - c_1 - q_i.$$

It can be seen that any variable of the set $\{q_i,\, q_o,\, h,\, \dot{h}\}$ is reachable from y. □

5.3.3 Causal Interpretation of Oriented Structure Graphs

The aim of this subsection is to discuss the causal interpretation of the oriented bipartite graph associated with a matching.

As stated above, selecting a pair $(c,\ z)$ to belong to a matching implies a causality assignment, by which the constraint c is used to compute the variable z, assuming the other variables to be known. The oriented bipartite graph which results from a causality assignment is named a *causal graph*. Causal graphs are used in qualitative reasoning, alarm filtering or in providing the computation chain needed for the numerical or formal determination of some variables of interest, as shown by the above interpretation. Although this interpretation is straightforward for simple algebraic constraints, it has to be considered more carefully when strongly coupled subgraphs or differential constraints are present. The following paragraphs deal with these situations.

Algebraic constraints. Let $c \in \mathcal{C}$ be an algebraic constraint, $Q(c)$ the set of the variables occurring in c and $n_c = |Q(c)|$. In the structural analysis, the following assumption is made:

Assumption 5.2 Any algebraic constraint $c \in \mathcal{C}$ defines a manifold of dimension $n_c - 1$ in the space of the variables $Q(c)$.

Since the constraint has to be satisfied at any time t, the variables of the set $Q(c)$ are not independent of each other. Assumption 5.2 means that only $n_c - 1$ variables can be chosen arbitrarily while the remaining variable is given by the constraint c. Hence, there is at least one variable $z \in Q(c)$ such that $\frac{\partial c}{\partial z} \neq 0$ holds almost everywhere in the space of the variables $Q(c)$.[2] Therefore, from the implicit function theorem, its value can be deduced (at least locally) from the constraint c and the values of the $n_c - 1$ other variables. This is exactly the causal interpretation of matching the variable z with constraint c. Stated differently, the constraint c decreases by one the degrees of freedom associated with the variables $Q(c)$.

Example 5.12 Algebraic constraints
Consider the constraint

$$c_1 : a_1 x_1 + b_1 x_2 - y_1 = 0, \tag{5.14}$$

where x_1 and x_2 are two unknowns, a_1 and b_1 are parameters, and y_1 is known. This constraint obviously defines a one-dimensional surface in the space of all vectors $(x_1, x_2)^{\mathrm{T}}$. Thus only one degree of freedom is left because only one of the unknowns can be chosen arbitrarily, the possible value(s) of the other one being deduced from (5.14).

The set $Q(c_1)$ of variables is given by

$$Q(c_1) = \{x_1, x_2, y_1\}$$

[2]The term $\frac{\partial c}{\partial z}$ denotes the derivative of the left-hand side of the constraint $c(z_1, z_2 \ldots) = 0$.

because, for example,

$$\frac{\partial c_1}{\partial x_1} = a_1 \neq 0$$

holds, which illustrates the use of the derivative $\frac{\partial c}{\partial z}$ used above.

Note that the structural point of view considers the most general case of any pair of parameters a_1 and b_1. Particular cases result if a_1 or b_1 equals to zero, where Eq. (5.14) would still define a one-dimensional manifold, or if a_1 and b_1 both equal to zero. In the latter case c_1 would not define a one-dimensional manifold when $y_1 = 0$, because any point $(x_1, x_2)^{\mathrm{T}}$ in the two-dimensional space would satisfy the constraint, and there would be no solution when $y_1 \neq 0$. □

The fact that *at least one* variable can be matched in a given constraint under the causal interpretation does not mean that *any* variable has this property. An obvious situation in which (c, x) cannot be matched is when c is not invertible with respect to x. The constraint shown in Fig. 5.12 defines a manifold of dimension 1 in $|\mathcal{R}^2$, and it is always possible to compute x_2 once x_1 is given. Matching x_2 with this constraint can obviously be interpreted as explained above. However, the interpretation does not apply to the matching of x_1, because $\frac{\partial c}{\partial x_1}$ is not different from zero almost everywhere, thus, the constraint c cannot be used to compute x_1 whatever be the value of x_2.

Differential constraints. The case of differential constraints has to be considered carefully. Differential constraints can always be represented as

$$d : x_2(t) - \frac{\mathrm{d}}{\mathrm{d}t} x_1(t) = 0, \tag{5.15}$$

which means that the functions $x_1(t)$ and $x_2(t)$ cannot be chosen independently of one other. This differential constraint has two possible matchings:

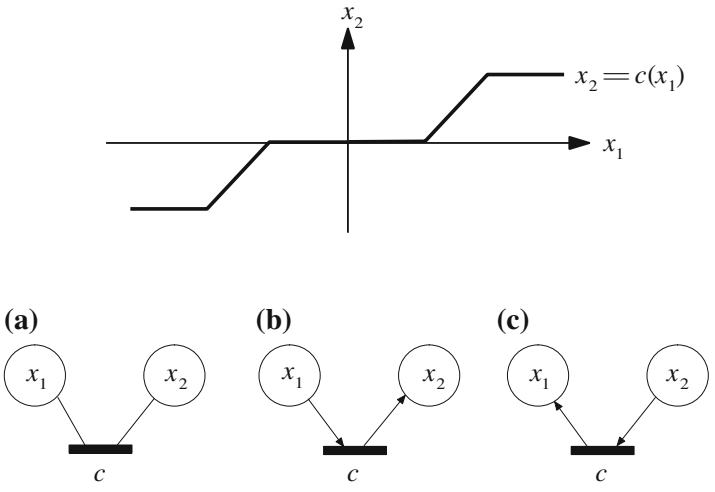


Fig. 5.12 Structure graph (**a**), possible (**b**) and impossible matching (**c**)

- If the trajectory $x_1(t)$ is known, its derivative can always be computed (from an analytical point of view, derivatives are here supposed to exist, and from a numerical point of view, there might be problems rised by the presence of noise, which are not considered here). It follows that the constraint can always be matched for x_2 which is then uniquely defined. This is called *derivative causality.*

- If, on the other hand, $x_2(t)$ is known, matching this constraint for x_1 (which is called *integral causality*) leads to the computation

$$x_1(t) = x_1(0) + \int_0^t x_2(\sigma)\,\mathrm{d}\sigma, \tag{5.16}$$

  which does not determine $x_1(t)$ uniquely, unless the initial condition $x_1(0)$ is known.

Let $(x_1(t),\ x_2(t))^{\mathrm{T}}$ be two functions which satisfy the constraint d. Then, any linear combination $(x_1(t) + \alpha,\ x_2(t))^{\mathrm{T}}$, where α is any constant function, also satisfies the constraint d. Thus, computing x_1 from constraint d may be possible or impossible, depending on the context. Initial values are known in a simulation context, since they are under the control of the user, but this is generally not true in a fault diagnosis context. Hence, the use of integral causality needs to be carefully considered or just avoided.

Remark 5.2 (*Consequences for residual generation*) Parity space or identification-based residual generation approaches aim at eliminating the unknown initial values by using the system input–output relations which are obtained through derivative causality. The observer-based approaches use integral causality by implementing an auxiliary system—the observer—which provides results that are (asymptotically) independent of the estimate of the initial state. □

In summary, different cases have to be considered as far as the counterpart of Assumption 5.2 is concerned stating that a differential constraint (5.15) defines a manifold of dimension $n_c - 1$ in the space of the variables $Q(c)$:

- If $x_1(t)$ is known, $x_2(t)$ can be matched with constraint d which leads to differential causality. This provides a unique result for $x_2(t)$. Assumption 5.2 is satisfied since constraint d leaves only one degree of freedom in the determination of $(x_1(t),\ x_2(t))$.

- If $x_2(t)$ and the initial value $x_1(0)$ are known, $x_1(t)$ can be matched with constraint d using integral causality. This situation leads to a unique result obtained from Eq. (5.16). Assumption 5.2 is satisfied since constraint d leaves only one degree of freedom in the determination of $(x_1(t),\ x_2(t))$.

- If only $x_2(t)$ is known, Assumption 5.2 is not satisfied, because whatever matching is used, two degrees of freedom (the constant function α, and the input function $x_2(t)$) remain for the determination of $(x_1(t),\ x_2(t))$.

Direction of calculability in the structure graph. To show direction of calculability (causality) in a structure graph, the symbol **x** is used in position (i, j) indicates that the variable in column j cannot be calculated from the constraint in row i. This is illustrated in Example 5.14 where h cannot be calculated from c_6.

Example 5.13 First-order system
A model whose solution exists but is not unique, as the result of Assumption 5.2 being not satisfied, is given by the following single-input first-order system:

$$c_1 \ : \ x_2 - ax_1 - bu = 0$$
$$c_2 \ : \ x_2 - \frac{\mathrm{d}}{\mathrm{d}t}x_1 = 0.$$

Constraint c_1 is algebraic and expresses the fact that the vector $(x_1, \ x_2)^{\mathrm{T}}$ lives in a linear manifold of dimension one for every known input u. Constraint c_2 does not allow to decrease the dimension of the unknown vector. If x_1 were known (which is not the case), one could compute its derivative x_2, but the knowledge of x_2 (which could be obtained as a function of x_1 and u in constraint c_1) is of no help to compute x_1 because one should proceed by integration and the initial value $x_1(0)$ is unknown. □

Example 5.14 Derivative causality in the tank system
Consider the following matching in the tank structural model.

↗	h	$\dot{h}$	q_i	q_o	u	y
c_1		①	1	1		
c_2			①		1	
c_3	1			①		
c_4	1					1
c_5					1	1
c_6	①	1				

Although it is complete with respect to the variables $\{h, \ \dot{h}, \ q_i, \ q_o\}$, it cannot be used for the computation of these variables because it introduces an integral causality, where h should be computed from $\dot{h}$ by constraint c_6, while its initial value is not known because constraint c_4 is not matched.

Derivative causality can be forced, if necessary. To represent this situation, the symbol **x** is used, which forbids integral matchings. The previous matching will not be obtained if the tank structural model is written as

↗	h	$\dot{h}$	q_i	q_o	u	y
c_1		1	1	1		
c_2			①		1	
c_3	1			①		
c_4	①					1
c_5					1	1
c_6	**x**	①				

where **x** means that although there is an edge between c_6 and h, h cannot be matched with c_6. Instead, c_6 is used to match $\dot{h}$. □

Strongly connected subgraphs. In the oriented graph associated with a matching, strongly connected parts may occur for which the stepwise causal interpretation does not lead to a sequence of calculations of the unknown variables, but another approach is needed to obtain a matching. A subset of vertices is said to be *strongly connected* if there exists a path between any pair of vertices belonging to this subset.

Strongly connected subgraphs are structures within the graph, which consist of constraints and unknown variables that need be solved simultaneously.

The causal interpretation of a strongly connected subgraph is that the constraints and variables belonging to the subgraph can be matched when all the other variables (not matched in the strongly connected subgraphs) are known. Suppose that n_v variables are constrained by a subsystem of n_l constraints, and there is a matching such that they form a cyclic structure (loop). Then, n_l variables are internal (matched within the loop), and $n_\mathrm{v} - n_\mathrm{l}$ variables are external (not matched within the strongly connected subgraph).

An example of strongly connected constraints are three linear equations with two variables and both variables enter into each of the equations. Since there is no equation (constraint) with only one unknown variable, two of the equations need be solved simultaneously to determine the two unknowns. Alternatively, one constraint is chosen to express one of the variables by the other, and this result is inserted in one of the other constraints to solve for the second variable.

In more general terms, in structural analysis, an algebraic loop is always supposed to have a unique solution (more precisely: a finite number of solutions), which in the space of unknown variables corresponds to the intersection of n_l manifolds of dimension $n_\mathrm{l} - 1$, if the external variables are known (by Assumption 5.2). The loop is associated with a subset of n_l constraints that is written here as vector equation

$$\boldsymbol{h}_\mathrm{l}(\boldsymbol{x}_\mathrm{l}, \boldsymbol{x}_\mathrm{e}) = \boldsymbol{0},$$

where $\boldsymbol{x}_\mathrm{l}$ and $\boldsymbol{x}_\mathrm{e}$ are the vectors of the internal and the known external variables, and each component of $\boldsymbol{x}_\mathrm{l}$ is matched with one constraint in $\boldsymbol{h}_\mathrm{l}$.

It is worth noticing that the interpretation associated with causality in single constraints is not directly extendable to strongly connected subgraphs, as shown by the following example:

Example 5.15 Non-invertible constraints
Consider the non-invertible constraint from Example 5.12 and suppose now that there are two constraints $\{c_1, c_2\}$ of the same form, but with different parameters. The incidence matrix of the structure graph of this system is

↗	x_1	x_2
c_1	1	1
c_2	1	1

A complete matching is given by

↗	x_1	x_2
c_1	①	1
c_2	1	①

The matching illustrated in Fig. 5.13 of x_1 with c_1 is obtained by choosing the variable that should be computed from one specific constraint.

The correct interpretation comes from the fact that each constraint defines a (different) manifold dimension one in $|\mathcal{R}^2$, and that, in general, such two manifolds intersect in a finite number of points. To get no solution at all would be a particular case (which would not satisfy Assumption 5.1), and an infinite number of solutions would be the result of the two manifolds were the same one (at least locally). □

The uniqueness of the solution associated with a cyclic structure that contains differential constraints depends upon the context of the problem. Consider a set of $n_l + n_e$ variables which is constrained by n_l differential equations

$$\begin{aligned} z_l &= \boldsymbol{g}_l(\boldsymbol{x}_l, \boldsymbol{x}_e, \boldsymbol{u}) \\ z_l &= \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}_l, \end{aligned} \tag{5.17}$$

where $\boldsymbol{x}_l$ is the vector of the variables in the loop, $\boldsymbol{g}_l$ are the constraints in the loop, and $\boldsymbol{x}_e$ are the external variables, which are supposed to be known. The system (5.17) has a unique solution only if the initial value $\boldsymbol{x}_l(0)$ is known. If this is not the case, the

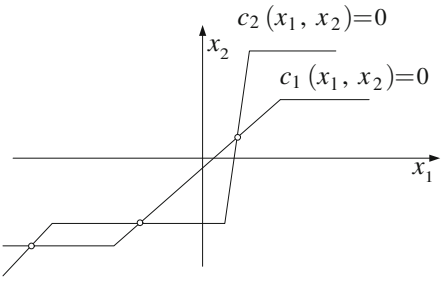


Fig. 5.13 Two algebraic constraints with two unknowns

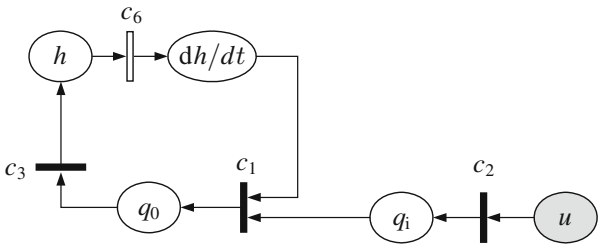


Fig. 5.14 A matching with a differential loop

solution will depend on the n_l unknowns $\boldsymbol{x}_l(0)$ and thus it will belong to a manifold of dimension n_l. Such a differential loop is called *non-causal*.

Example 5.16 Differential loop in the tank example
Consider the following matching

↗	h	$\dot{h}$	q_i	q_o	u	y
c_1		1	1	①		
c_2			①		1	
c_3	①			1		
c_4	1					1
c_5					1	1
c_6	1	①				

which is complete with respect to the variable set $\{h, \dot{h}, q_i, q_o\}$, and in which differential causality is now used for constraint c_6. The matching results in the differential loop $h - c_6 - \dot{h} - c_1 - q_0 - c_3 - h$, which is shown in Fig. 5.14. Although the matching is complete with respect to the set of unknown variables, it is impossible to determine $h(t)$, because the initial value $h(0)$ is unknown. □

Following a classical graph-theoretic approach, a loop can be condensed into one single node, which represents a subsystem of constraints to be solved simultaneously. Another approach is to avoid loops (whenever possible) by some transformation of the constraints, leading to diagonal or triangular system structures.

Example 5.17 Treatment of loops
Consider a subsystem with $\mathcal{Z} = \{x_1, x_2, y_1, y_2\}, \mathcal{C} = \{c_1, c_2\}$. The variables are real numbers, the constraints are linear, y_1, y_2 are supposed to be known, and the problem to be solved concerns the computation of x_1, x_2 by using the constraints

$$\begin{aligned} c_1 &: a\,y_1 + b\,x_1 + c\,x_2 = 0 \\ c_2 &: \alpha\,y_2 + \beta\,x_1 + \gamma\,x_2 = 0. \end{aligned} \tag{5.18}$$

The incidence matrix of the structure graph and a complete matching w.r.t. $\{x_1, x_2\}$ is given as follows:

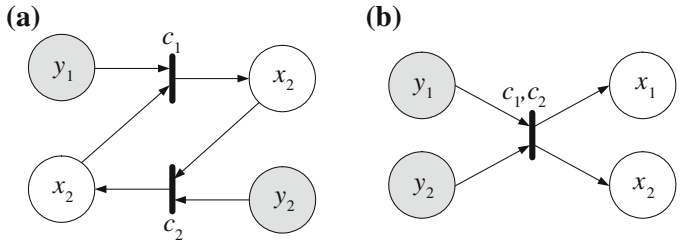


Fig. 5.15 An algebraic loop

↗	x_1	x_2	y_1	y_2
c_1	①	1	1	
c_2	1	①		1

Figure 5.15a shows the resulting loop in the associated oriented graph. In the structure graph it is supposed that b and γ are non-zero. The linear equations c_1 and c_2 are solvable under the condition $b\gamma - c\beta \neq 0$, which cannot be seen from structural considerations.

Figure 5.15b illustrates the condensation in which the loop is "condensed" into one single node, which means that the two equations with two unknowns are solved simultaneously, but no detail is given by the condensed structure graph about how this is done.

Transforming the constraints may also lead to a loop-free oriented graph, because it may give the system a diagonal or a triangular structure. For example, the two following systems are equivalent to (5.18):

$$\begin{aligned} c_1' &: a\gamma y_1 - \alpha c y_2 + (b\gamma - \beta c) x_1 = 0 \\ c_2' &: a\beta y_1 - \alpha b y_2 + (c\beta - b\gamma) x_2 = 0 \end{aligned} \tag{5.19}$$

and

$$\begin{aligned} c_1' &: a\gamma y_1 - \alpha c y_2 + (b\gamma - \beta c) x_1 = 0 \\ c_2'' &: a y_1 + b x_1 + c x_2 = 0. \end{aligned} \tag{5.20}$$

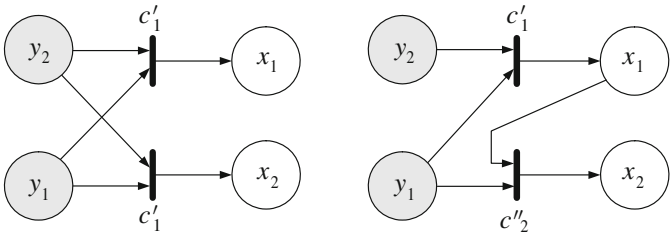


Fig. 5.16 Two equivalent loop-free oriented graphs

Figure 5.16 illustrates the loop-free graphs associated with the transformed systems (5.19) and (5.20). Note that the new systems result from manipulations which are not purely structural, but which are done on the behaviour model. □

5.4 Structural Decomposition of Systems

5.4.1 Canonical Subsystems

This section recalls a classical result from bipartite graph theory, which states that any finite-dimensional graph can be decomposed into three subgraphs with specific properties: an over-constrained, a just-constrained and an under-constrained subgraph. This decomposition is canonical, i.e. for a given system, it is unique. The three subgraphs and the associated subsystems play a major role in the structural analysis and lead to the important structural properties of observability, controllability, monitorability and reconfigurability.

The following definition shows the consequences of the existence of complete matchings for the solution of constraint sets that are structurally described by a bipartite graph $\mathcal{G}$.

Definition 5.6 (*Over-constrained, just-constrained, under-constrained graph*) A graph $\mathcal{G} = (\mathcal{C}, \mathcal{Z}, \mathcal{E})$ is called

- *over-constrained* if there is a complete matching on the variables $\mathcal{Z}$ but not on the constraints $\mathcal{C}$,
- *just-constrained* if there is a complete matching on the variables $\mathcal{Z}$ and on the constraints $\mathcal{C}$,
- *under-constrained* if there is a complete matching on the constraints $\mathcal{C}$ but not on the variables $\mathcal{Z}$.

In an over-constrained graph, there remains a complete matching on $\mathcal{Z}$ after any single constraint has been removed from the set $\mathcal{C}$.

Example 5.18 Property of the reduced graph of the tank system.
Matching 2 of the structure graph of the tank system shown in Example 5.9 is complete with respect to the variables, but there is still one non-matched constraint. Hence, the reduced graph of the tank system is over-constrained.

↗	h	$\dot{h}$	q_i	q_o	u	y
c_1		1	①	1		
c_2			1		1	
c_3	1			①		
c_4	①					1
c_6	1	①				

It can be furthermore noticed that any of the five constraints can be removed, and there still is a complete matching on the resulting graph. □

A graph $\mathcal{G}$ may fail to conform to any of the three properties defined above. In this case, it can be proved that there exists a unique decomposition of $\mathcal{G}$ into three subgraphs, which are defined by the partitions

$$\begin{aligned} \mathcal{C} &= \mathcal{C}^- \cup \mathcal{C}^0 \cup \mathcal{C}^+ \\ \mathcal{Z} &= \mathcal{Z}^- \cup \mathcal{Z}^0 \cup \mathcal{Z}^+ \end{aligned} \tag{5.21}$$

of the sets $\mathcal{Z}$ and $\mathcal{C}$. The subgraphs are denoted by

$$\begin{aligned} \mathcal{G}^+ &= (\mathcal{C}^+, \mathcal{Z}^+, \mathcal{E}^+) \\ \mathcal{G}^0 &= (\mathcal{C}^0, \mathcal{Z}^0, \mathcal{E}^0) \\ \mathcal{G}^- &= (\mathcal{C}^-, \mathcal{Z}^-, \mathcal{E}^-), \end{aligned}$$

where $\mathcal{E}^-$, $\mathcal{E}^0$ and $\mathcal{E}^+$ are the subsets of $\mathcal{E}$ with the edges that connect vertices of $\mathcal{C}^-$ with $\mathcal{Z}^-$, $\mathcal{C}^0$ with $\mathcal{Z}^0$ or $\mathcal{C}^+$ with $\mathcal{Z}^+$, respectively. Note that the sets $\mathcal{E}^-$, $\mathcal{E}^0$ and $\mathcal{E}^+$ do not represent a partition of the edge set $\mathcal{E}$ of the overall graph, but a subset of it. As the important fact of this decomposition, the graphs $\mathcal{G}^-$, $\mathcal{G}^0$ and $\mathcal{G}^+$ are under-constrained, just-constrained or over-constrained, respectively. This decomposition has been introduced by DULMAGE and MENDELSOHN in 1958 and is, therefore, also called the *DM decomposition*.

Theorem 5.1 (DM decomposition of bipartite graphs) *Each bipartite graph* $\mathcal{G} = (\mathcal{C}, \mathcal{Z}, \mathcal{E})$ *can be decomposed into three subgraphs, which have the following properties:*

- **Over-constrained subgraph** $\mathcal{G}^+$*, which possesses a* $\mathcal{Z}$*-complete matching that is not* $\mathcal{C}$*-complete,*
- **Just-constrained subgraph** $\mathcal{G}^0$*, which possesses a complete matching,*
- **Under-constrained subgraph** $\mathcal{G}^-$*, which possesses a* $\mathcal{C}$*-complete matching that is not* $\mathcal{Z}$*-complete.*

As the choice of matchings for a graph is not unique it is important to state that the DM decomposition is unique. That is, the freedom in choosing matchings with the completeness properties mentioned in the theorem is restricted to the subsets of the vertices, which result from the decomposition (5.21).

As a consequence of the graph decomposition, the corresponding system $\mathcal{S} = (\mathcal{C}, \mathcal{Z})$ can be decomposed into three subsystems:

$$\begin{aligned}\mathcal{S}^+ &= \left(\mathcal{C}^+,\, \mathcal{Z}^+\right)\\ \mathcal{S}^0 &= \left(\mathcal{C}^0,\, \mathcal{Z}^+ \cup \mathcal{Z}^0\right)\\ \mathcal{S}^- &= \left(\mathcal{C}^-,\, \mathcal{Z}^+ \cup \mathcal{Z}^0 \cup \mathcal{Z}^-\right).\end{aligned}$$

In analogy with the corresponding subgraphs, these subsystems are classified as follows:

- $\mathcal{S}^+$ is called the *over-constrained subsystem* (also called the over-determined subsystem) and abbreviated as SO. It has more constraints than variables.
- A structurally over-constrained system $\mathcal{S}$ is said to be *proper structurally over-constrained* (PSO) if $\mathcal{S} = \mathcal{S}^+$.
- $\mathcal{S}^0$ is called the *just-constrained subsystem*. It has the same number of unknown variables and constraints if the variables of the set $\mathcal{Z}^0$ are interpreted as known variables ($|\mathcal{Z}^0| = |\mathcal{C}^0|$).
- $\mathcal{S}^-$ is called the *under-constrained subsystem* (under-determined subsystem). It has less constraints than variables ($|\mathcal{Z}^-| < |\mathcal{C}^-|$).

Subsystems which cannot be decomposed into smaller ones are said to be minimal subsystems.

Example 5.19 DM decomposition of a bipartite graph
The example graph in Fig. 5.17 illustrates the situation that a system $\mathcal{S}$ simultaneously can comprise the three subgraphs mentioned above:

$$\begin{aligned}\mathcal{S}^+ &= (\{c_1, c_2, c_3\}, \{z_2, z_3\})\\ \mathcal{S}^0 &= (\{c_4, c_5, c_6, c_7\}, \{z_4, z_5, z_6, z_7\})\\ \mathcal{S}^- &= (\{c_8, c_9\}, \{z_8, z_9, z_{10}\})\end{aligned}$$

with the associated incidence matrices

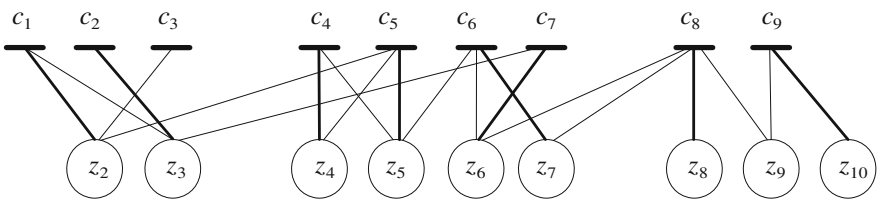


Fig. 5.17 Example of the canonical decomposition of a bipartite graph

$\boldsymbol{E}^+ =$

↗	z_2	z_3
c_3	1	
c_1	①	1
c_2		①

$\boldsymbol{E}^0 =$

↗	z_4	z_5	z_6	z_7
c_4	①	1		
c_5	1	①		
c_7			①	
c_6		1	1	①

and $\boldsymbol{E}^- =$

↗	z_8	z_{10}	z_9
c_8	①		1
c_9		①	1

The subgraph $\mathcal{G}^+$ drawn on the left part has a $\mathcal{Z}$-complete matching marked by the thick edges. Hence, this graph is over-constrained. The middle subgraph $\mathcal{G}^0$ possesses a complete matching and is just-constrained, whereas the right subgraph $\mathcal{G}^-$ has a $\mathcal{C}$-complete matching and is under-constrained. Note that the vertices are ordered such that the edges belonging to the matchings form a kind of "main diagonal" of the matrices. The edges that connect two subgraphs do not contribute to these matchings.

The incidence matrix of the overall graph $\mathcal{G}$ can be ordered in such a way that it contains the sub-matrices $\boldsymbol{E}^+$, $\boldsymbol{E}^0$ and $\boldsymbol{E}^-$ together with further entries, which represent the edges connecting the subgraphs,

$\boldsymbol{E} =$

↗	z_2	z_3	z_4	z_5	z_6	z_7	z_8	z_{10}	z_9
c_3	1								
c_1	①	1							
c_2		①							
c_4			①	1					
c_5	1		1	①					
c_7		1			①				
c_6				1	1	①			
c_8					1	1	①		1
c_9								①	1

Due to the ordering of the vertices in both sets, the matchings in all the three subgraphs build a diagonal line, in which the more abstract representation of Fig. 5.18 is drawn as the diagonal black line. The figure shows in an intuitive way those regions of the incidence matrix where the non-zero elements appear (grey) and those which include only zero entries (white). It further shows that the subgraph $\mathcal{G}^+$ has more vertices from the set $\mathcal{C}$ than from the set $\mathcal{Z}$, because the relation

$$|\mathcal{C}^+| > |\mathcal{Z}^+|$$

holds. Hence, the corresponding subsystem has more constraints than variables and is, hence, over-determined. For the other two subgraphs the relations

$$|\mathcal{C}^0| = |\mathcal{Z}^0| \quad \text{and} \quad |\mathcal{C}^-| < |\mathcal{Z}^-|$$

hold.

Whether or not all the three subgraphs appear in an overall graph $\mathcal{G}$ is not directly related to the cardinalities of $\mathcal{C}$ and $\mathcal{Z}$. That is, even if the graph has more $\mathcal{Z}$-vertices than $\mathcal{C}$-vertices, it may still comprise a part that is over-constrained.

The DM decomposition includes more information about the graph, namely the connection among the variables and the constraints. The DM decomposition is unique, which means that the partition (5.21) of the sets $\mathcal{C}$ and $\mathcal{Z}$ is unique. Whatever complete matchings are used, the same vertices appear in the three subgraphs in all resulting decompositions. What depends upon the matchings used is the order of the vertices in the incidence matrix after the ① entries have been brought into the main diagonal. For example, if in the subgraph $\mathcal{G}^+$ the matching

$$\mathcal{M} = \{(c_3, z_2),\ (c_2, z_3)\}$$

is used, which is $\mathcal{Z}$-complete as well, the order of the rows for c_1 and c_3 in the matrix $\boldsymbol{E}$ has to be exchanged, but the DM decomposition remains the same. □

For later use, it is convenient to define a measure of structural redundancy, which is associated with the over-constrained part of a system, $\mathcal{S}^+$.

Definition 5.7 (*Structural redundancy measure*) Given a set of constraints $\mathcal{C}$, and let $Q(\mathcal{C}) \subseteq \mathcal{Z}$ be the subset of variables in $\mathcal{Z}$ connected to at least one constraint in $\mathcal{C}$. The structural redundancy measure is

$$\varrho(\mathcal{C}) = |\mathcal{C}^+| - |Q(\mathcal{C}^+)|. \tag{5.22}$$

Example 5.20 DM decomposition of the single-tank system
Rearranging the rows and columns related to unknown variables of the structure graph for the single-tank system introduced in Example 5.4 on p. 124, the incidence matrix becomes:

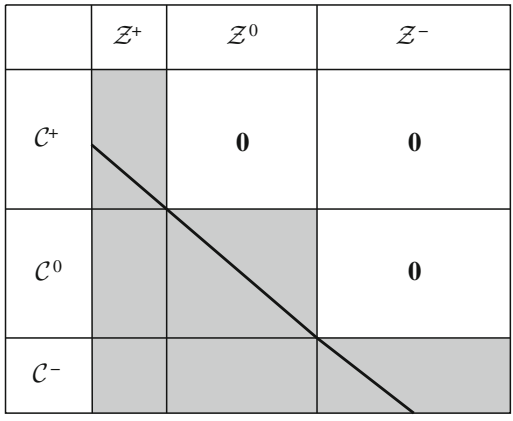


Fig. 5.18 Canonical decomposition of the structure graph

↗	h	$\dot{h}$	q_o	q_i	u	y
c_1		1	1	1		
c_4	①					1
c_6	1	①				
c_3	1		①			
c_2				①	1	

As the ①-elements of a maximum matching show, the single-tank system is over-constrained:

$$\mathcal{S}^+ = \{\{c_1, c_4, c_6, c_3, c_2\}, \{h, \dot{h}, q_\mathrm{o}, q_\mathrm{i}\}\}.$$

The decomposition is independent of the known variables, which are added to the right of the table for completeness. The structural redundancy measure is

$$\begin{aligned} \varrho(\mathcal{C}) &= |\mathcal{C}^+| - |Q(\mathcal{C}^+)| && (5.23)\\ &= |\{c_1, c_4, c_6, c_3, c_2\}| - |\{h, \dot{h}, q_\mathrm{o}, q_\mathrm{i}\}| = 1 && (5.24) \end{aligned}$$

If c_2 was removed from the system, the modified system would be just-constrained with

$$\mathcal{S}^0 = \{\{c_4, c_6, c_3, c_1\}, \{h, \dot{h}, q_\mathrm{o}, q_\mathrm{i}\}\}.$$

and $\varrho(\mathcal{C}) = 0$. □

Further decomposition of the just-constrained subgraph. The graph $\mathcal{G}^0 = (\mathcal{C}^0, \mathcal{Z}^0, \mathcal{E}^0)$ can be further decomposed as it will be explained in this paragraph. As stated above, this graph includes a set of n^0 constraints that can be used to determine the same number of variables. The decomposition introduced now splits these sets of constraints and variables in such a way that the smaller constraint sets can be used consecutively to determine the associated unknown variables. What happens in this decomposition can be seen in Fig. 5.19, where the incidence matrix of the just-constrained subgraph is a lower block triangular matrix.

The decomposition starts after the edges of the just-constrained subgraph have been given the directions described in Sect. 5.3.2, namely the orientation from the $\mathcal{C}$-vertex towards the $\mathcal{Z}$-vertex for all edges belonging to the complete matching and the opposite direction for all remaining edges. Then the resulting oriented graph is decomposed into strongly connected components. The paths have to be built in accordance with the directions of the edges. As the structure graph is bipartite, the sets of strongly connected vertices include vertices of both kinds. If the corresponding subsets of $\mathcal{C}^0$ and $\mathcal{Z}^0$ are enumerated in the same way, the decomposition of the just-constrained subgraph $\mathcal{G}^0$ results in partitions of these sets:

$$\begin{aligned} \mathcal{C}^0 &= \mathcal{C}_1^0 \cup \mathcal{C}_2^0 \cup \cdots \cup \mathcal{C}_q^0 \\ \mathcal{Z}^0 &= \mathcal{Z}_1^0 \cup \mathcal{Z}_2^0 \cup \cdots \cup \mathcal{Z}_q^0, \end{aligned}$$

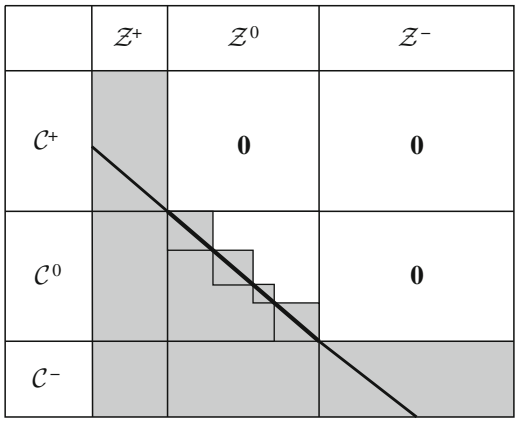


Fig. 5.19 Incidence matrix after the detailed decomposition of the just-constrained subgraph

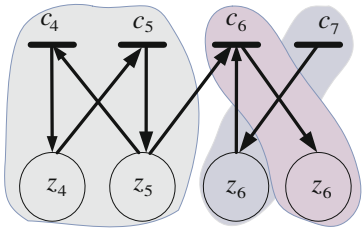


Fig. 5.20 Decomposition of the just-constrained subgraph $\mathcal{G}^0$ into strongly connected components

where q is the number of strongly connected components obtained. If the rows and columns of the incidence matrix belonging to the just-constrained subgraph are ordered accordingly, the lower block-diagonal matrix shown in the middle of Fig. 5.19 results.

Example 5.21 Decomposition of the just-constrained subgraph
Figure 5.20 shows the subgraph $\mathcal{G}^0$ of the bipartite graph introduced in Example 5.19. The edges have the prescribed orientation from the $\mathcal{C}$-vertex towards the $\mathcal{Z}$-vertex for all edges belonging to the complete matching and the opposite direction for the other edges. The grey fields mark the three different strongly connected components. The constraint c_6 and the variable-vertex z_7 belong together, because the edge $c_6 \rightarrow z_7$ belongs to the matching used in the DM decomposition. For the same reason, c_7 and z_6 represent a strongly connected component.

After ordering the vertices according to this decomposition, the following incidence matrix $\boldsymbol{E}^0$ is obtained:

$\boldsymbol{E}^0 =$

↗	z_4	z_5	z_7	z_6
c_4	①	1		
c_5	1	①		
c_7			①	
c_6		1	1	①

Hence, the just-constrained subgraph of this example can be further decomposed into three subgraphs that can be used consecutively for determining the unknown variables. Note that the variables z_5 and z_7 will be determined by the constraint c_5 or z_7, respectively, and these results will influence the determination of the variable z_6 by using the constraint c_6. This fact illustrates that the order of the subsets is important to retain the causality of the graph and, hence, the order of the computation of the unknown variables. □

5.4.2 Interpretation of the Canonical Decomposition

This subsection addresses the canonical subsystems with respect to existence of solutions, thus providing a key for later analysis of structural observability and controllability.

First, it is clear that Assumption 5.1 (a) on p. 122 must be satisfied by each of the subsets of constraints $\mathcal{C}^+$, $\mathcal{C}^0$ and $\mathcal{C}^-$. If this was not true, the system model would have no solution, which contradicts with the fact that it describes the behaviour of a physical system (which indeed has a solution).

Second, from the structural point of view, any algebraic constraint is assumed to satisfy Assumption 5.2 on p. 141, thus, a subset of n variables completely matched within a subset of n constraints is uniquely defined, while the result depends on the causality and on the existence of differential loops when constraints of the form

$$d: \quad z_2(t) - \frac{\mathrm{d}}{\mathrm{d}t} z_1(t) = 0$$

are considered. Finally, it will be seen that there are cases in which Assumption 5.1 (b) cannot hold true.

Static systems. The behavioural model of static systems contains only algebraic constraints. In the **over-constrained subsystem** $(\mathcal{C}^+, Q(\mathcal{C}^+))$ the variables in the set $\mathcal{Z}^+ = Q(\mathcal{C}^+)$ have to satisfy more than $n^+ = |\mathcal{Z}^+|$ constraints. Since there are more manifolds than variables, no solution can exist if they also satisfy Assumption 5.1 (b). As the model should have at least one solution for a given physical system, one concludes that the constraints in $\mathcal{C}^+$ are not independent, i.e. the system description is redundant. In other words, for the system to have a solution, some compatibility conditions must hold. Structural analysis always assumes the most general case, i.e. the minimum number of relations between the system parameters. This means that the number of independent constraints is maximal, thus leading to the following equivalent conclusions:

- The over-constrained subsystem has a unique solution (more generally, it has a finite number of solutions).
- The number of independent constraints in $\mathcal{C}^+$ is n^+.
- The number of compatibility conditions is $\left|\mathcal{C}^+\right| - n^+$.

In the **just-constrained subsystem**, $(\mathcal{C}^0,\ Q(\mathcal{C}^0))$, the n^0 variables in the set $\mathcal{Z}^0$ have to satisfy exactly n^0 constraints, which satisfy Assumptions 5.2 and 5.1(a). A unique solution exists, which is the intersection of the manifolds associated with the constraints $\mathcal{C}^0$, which are assumed to satisfy Assumption 5.1(b). This being the most general case, structural analysis proposes the following conclusions:

- The just-constrained subsystem has a unique solution.
- The number of independent constraints in $\mathcal{C}^0$ is n^0.
- There is no compatibility condition.

In the **under-constrained subsystem**, $(\mathcal{C}^-,\ Q(\mathcal{C}^-))$, the n^- variables in the set $\mathcal{Z}^-$ have to satisfy less than n^- constraints, which satisfy Assumptions 5.2 and 5.1(a). All what the model can tell is that the unique solution of the physical system belongs to the intersection of less than n^- manifolds, and thus the solution is not uniquely defined by the model. It belongs to a manifold of dimension $n^- - \left|\mathcal{C}^-\right|$ if the constraints also satisfy Assumption 5.1(b). This being the most general case, structural analysis proposes the following conclusions:

- The under-constrained subsystem has no unique solution.
- The constraints in $\mathcal{C}^-$ are independent.
- There is no compatibility condition.

Example 5.22 Compatibility conditions in an over-constrained subsystem
Consider the set of linear constraints

$$\begin{aligned} c_1 &: a_1 x_1 + b_1 x_2 - y_1 = 0 \\ c_2 &: a_2 x_1 + b_2 x_2 - y_2 = 0 \\ c_3 &: a_3 x_1 + b_3 x_2 - y_3 = 0, \end{aligned} \tag{5.25}$$

where $\boldsymbol{a} = (a_1,\ a_2,\ a_3)^{\mathrm{T}}$ and $\boldsymbol{b} = (b_1,\ b_2,\ b_3)^{\mathrm{T}}$ are known parameter vectors and $\boldsymbol{y} = (y_1,\ y_2,\ y_3)^{\mathrm{T}}$ is a known signal vector. This system is clearly over-constrained with respect to the unknown variables $(x_1,\ x_2)$. However, whether or not this system of linear equations has a solution depends upon the following cases:

1. rank $(\boldsymbol{a},\ \boldsymbol{b},\ \boldsymbol{y}) = 3$, i.e. $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{y}$ are linearly independent vectors. The system (5.25) has no solution, because the three constraints are incompatible. Assumptions 5.1 and 5.2 cannot hold simultaneously.
2. If rank $(\boldsymbol{a},\ \boldsymbol{b},\ \boldsymbol{y}) = 2$, one solution exists. Note that the parameters and the known variables are no longer independent but the matrix $(\boldsymbol{a},\ \boldsymbol{b},\ \boldsymbol{y})$ has a vanishing eigenvalue and

$$\boldsymbol{y} = \lambda \boldsymbol{a} + \mu \boldsymbol{b}$$

   leads to the unique solution $x_1 = \lambda$ and $x_2 = \mu$.
3. If rank $(\boldsymbol{a},\ \boldsymbol{b},\ \boldsymbol{y}) = 1$, the matrix $(\boldsymbol{a}\ \boldsymbol{b}\ \boldsymbol{y})$ has two null eigenvalues. Any pair (x_1, x_2) such that $x_1 + x_2 - \lambda\mu = 0$ satisfies Eq. (5.25). Note that in that case, two compatibility conditions exist, and Assumption 5.1 does not hold.

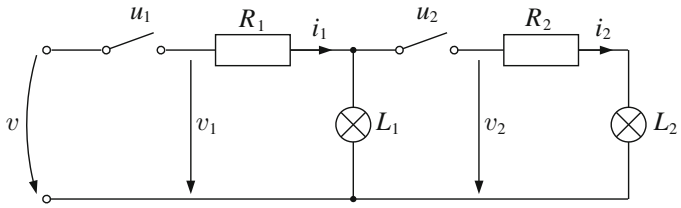


Fig. 5.21 Circuit of a tail lamp

4. The last case is rank $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{y}) = 0$, i.e. $\boldsymbol{a} = \boldsymbol{b} = \boldsymbol{y} = \boldsymbol{0}$. In this case, all parameters are specified and any pair $(x, y)^{\mathrm{T}} \in |\mathcal{R}^2$ satisfies the system of equations. Assumption 5.2 does not hold.

Since Eq. (5.25) is the behavioural model of a physical system, it should exhibit at least one solution. Then obviously the most general situation is Case 2 in which only one relation holds between the parameters. This is what assumed in any structural analysis. □

Example 5.23 Structural analysis of a tail lamp
Figure 5.21 shows the simplified circuit of a tail lamp of a car, which is represented by the following constraints:

$$
\begin{aligned}
c_1: \quad & v_1 = \begin{cases} v & \text{if } u_1 = 1 \\ 0 & \text{if } u_1 = 0 \end{cases} \\
c_2: \quad & v_1 - i_1 R_1 - (i_1 - i_2) R_{\mathrm{L1}} = 0 \\
c_3: \quad & v_2 = \begin{cases} v_1 - i_1 R_1 & \text{if } u_2 = 1 \\ 0 & \text{if } u_2 = 0 \end{cases} \\
c_4: \quad & v_2 - i_2 R_2 - i_2 R_{\mathrm{L2}} = 0
\end{aligned}
$$

with the set of variables

$$\mathcal{Z} = \{v, v_1, v_2, i_1, i_2, u_1, u_2\}.$$

The structure graph is shown in Fig. 5.22a. In the analysis the circuit is considered for closed switches, where the model can be reformulated as a set of linear equations, which is given by

$$
\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & -R_1 - R_{\mathrm{L1}} & 0 & -R_{\mathrm{L1}} \\ -1 & R_1 & 1 & 0 \\ 0 & 0 & 1 & -R_2 - R_{\mathrm{L2}} \end{pmatrix}}_{\boldsymbol{A}} \begin{pmatrix} v_1 \\ i_1 \\ v_2 \\ i_2 \end{pmatrix} = \begin{pmatrix} v \\ 0 \\ 0 \\ 0 \end{pmatrix}, \tag{5.26}
$$

where the variable v is the known input voltage.

For given v, the reduced graph has the canonical decomposition shown by the arrows in Fig. 5.22b. It is interesting to see that for linear static systems this decomposition determines the structural rank of the matrix $\boldsymbol{A}$, which is given by the maximum number of entries that can be chosen in different rows and different columns as indicated by the ① in the following scheme:

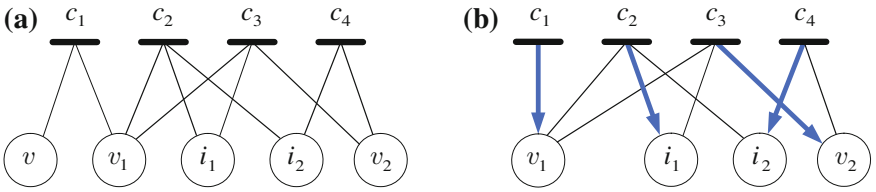


Fig. 5.22 Structure graph of the circuit (**a**) and DM decomposition of the reduced structure graph (**b**)

$$\text{s-rank}\begin{pmatrix} ① & 0 & 0 & 0 \\ 1 & ① & 0 & 1 \\ 1 & 1 & ① & 0 \\ 0 & 0 & 1 & ① \end{pmatrix} = 4.$$

The structural rank is determined for the structure matrix $[\boldsymbol{A}]$ rather than the matrix $\boldsymbol{A}$ itself. Therefore, all non-vanishing elements have been replaced by "1" in the structural rank condition. The ① correspond to the oriented edges in the graph and, hence, make a complete matching. The system is just-constrained.

The DM decomposition also shows how to determine the unknown variables i_1, i_2, v_2 and v_1. The more detailed decomposition leads to the subsystems

$$\begin{aligned} \mathcal{S}_1^0 &= (\{c_1\}, \{v_1\}) \\ \mathcal{S}_2^0 &= (\{c_2, c_3, c_4\}, \{v_1, i_1, i_2, v_2\}). \end{aligned}$$

Accordingly, in the first step, the constraint c_1 is used to determine v_1 in terms of the known variable v

$$v_1 = v.$$

Then the constraint set $\{c_2, c_3, c_4\}$ has to be used to determine the further three variables by solving the linear equation

$$\begin{pmatrix} -R_1 - R_{\mathrm{L}1} & 0 & -R_{\mathrm{L}1} \\ R_1 & 1 & 0 \\ 0 & 1 & -R_2 - R_{\mathrm{L}2} \end{pmatrix} \begin{pmatrix} i_1 \\ v_2 \\ i_2 \end{pmatrix} = \begin{pmatrix} -v_1 \\ v_1 \\ 0 \end{pmatrix}$$

for given v_1. The complete matching of this subsystem leads to an algebraic loop. Hence, the constraints c_2, c_3 and c_4 have to be used simultaneously to determine the three unknown variables (Fig. 5.23).

In this example, again, the difference between structural and numerical properties can be seen. As the structural rank of the matrix $\boldsymbol{A}$ is four, for almost all parameters occurring in the matrix the inverse $\boldsymbol{A}^{-1}$ exists and the linear equations have a unique solution. For exceptional cases, for which the determinant of $\boldsymbol{A}$ vanishes, the constraint set has no solution. For the tail lamp, this exceptional case is given by the equality

$$0 = \det \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & -R_1 - R_{\mathrm{L}1} & 0 & -R_{\mathrm{L}1} \\ -1 & R_1 & 1 & 0 \\ 0 & 0 & 1 & -R_2 - R_{\mathrm{L}2} \end{pmatrix}.$$

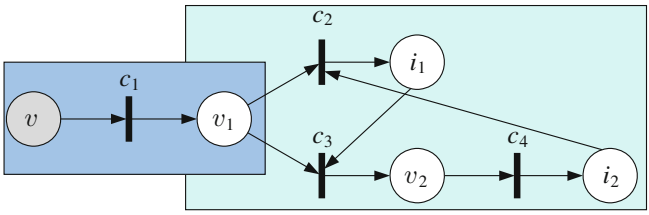


Fig. 5.23 Scheme for determining the unknown variables of the tail lamp for given input voltage v

Hence, for all parameter values that do not satisfy this equality, that is, for all parameters with

$$R_2 \neq \frac{R_1 R_{\mathrm{L}1}}{R_1 + R_{\mathrm{L}1}} - R_{\mathrm{L}2}$$

the structural rank transfers to the numerical rank of $\boldsymbol{A}$ and the model of the circuit has a unique solution for any given input voltage v. □

Dynamical systems. Remember that, when differential constraints are considered, matching all the variables in a subsystem guarantees that there is a unique solution under integral causality, i.e. if the initial conditions are known. Under derivative causality, the solution is unique if and only if there is a matching which avoids differential loops.

Let n_1^+ (respectively, n_1^0, n_1^-) be the maximum number of variables which can be matched in the over-constrained subsystem (respectively, in the just-constrained or the under-constrained subsystems) without introducing any differential loop. One obviously has $n_1^+ \leq n^+,\ n_1^0 \leq n^0,\ n_1^- \leq n^-$.

An over-constrained or a just-constrained subsystem is called *causal* if there exists a complete matching with respect to the variables $\mathcal{Z}^+$ and $\mathcal{Z}^0$ which do not contain any differential loop, i.e. if $n_1^+ = n^+$ or $n_1^0 = n^0$ holds. The under-constrained subsystem cannot be causal, because there does not exist any complete matching with respect to $\mathcal{Z}^-$.

Example 5.24 Causal over-constrained system
The following system

$$\begin{aligned} c_1 &: \ x_2 - a\,x_1 - bu = 0 \\ c_2 &: \ x_2 - \alpha\,x_1 - \beta u = 0 \\ c_3 &: \quad x_2 - \tfrac{\mathrm{d}}{\mathrm{d}t} x_1 = 0 \end{aligned}$$

is over-constrained with respect to the variables $(x_1,\ x_2)$, where u is supposed to be known. The system is causal because $(x_1,\ x_2)$ can be matched with $(c_1,\ c_2)$ without introducing a differential loop. Thus, there is a unique solution, which is obtained from the intersection of the two manifolds associated with $(c_1,\ c_2)$:

$$x_1 = \frac{\beta - b}{a - \alpha} u$$
$$x_2 = \left(\frac{a\beta - \alpha b}{a - \alpha} \right) u.$$

$a - \alpha$ is assumed not to be zero. Moreover, the constraint c_3 is redundant, and acts as a compatibility condition which has to be satisfied for the system solution to exist, namely

$$\left(\frac{a\beta - \alpha b}{a - \alpha} \right) u - \frac{\beta - b}{a - \alpha} \dot{u} = 0.$$

If the constraint c_2 does not exist, then the system is just-constrained but not causal. Its solution is defined up to the constant $x_1(0)$, which is unknown under differential causality. □

5.5 Matching Algorithms

From the definition, a matching can be represented in the incidence matrix of the bipartite graph by selecting at most one "1" in each row and in each column. This subsection shows how the selection should be done in order to find maximum matchings. An intuitive simple algorithm, referred to as *ranking*, is first introduced. This algorithm uses the causal interpretation of matchings and is well suited to understand the matching process but it cannot handle cases with strongly coupled subgraphs. As a general approach, the classical maximum matching algorithm is introduced and is followed by a maximum flow algorithm that can find all matchings in an elegant way that, however, is computationally heavy. A very efficient algorithm is then reviewed, which is based on directly finding all minimal structurally over-determined sets (MSO sets) in a structure graph. Examples are provided to show the use of the different algorithms.

5.5.1 Ranking Algorithm

According to the causal interpretation described above, a complete causal matching over the unknown variables identifies the computations to be done in order to express the unknown variables as a function of the variables that are known or have already been determined. If the matching is not complete with respect to the constraints, non-matched constraints exist that must be satisfied by the variables obtained. These facts are the basis of the following constraint propagation (or ranking) algorithm, which can be used to find a matching. The idea of this intuitive algorithm is to start with a known variable and to "propagate" the knowledge, step by step, by matching the variables which are present in constraints where all other involved variables are matched or known. The algorithm is not able to provide a matching in cases where subsystems are so closely coupled that a set of constraints and variables need be

solved simultaneously. Such problems require more elaborate algorithms that are introduced later in this section.

Algorithm 5.1 *Ranking of the constraints*

Given: Incidence matrix of a bipartite graph

1. Mark all known variables with rank 0
   $i = 1$
2. Find all constraints in the current table with exactly one unmarked variable. Associate rank i with these constraints and mark these constraints as well as the corresponding variable.
3. Set $i := i + 1$.
4. If there are unmarked constraints whose variables are all marked, associate them with rank i, mark them and connect them with the pseudo-variable ZERO.
5. If there are unmarked variables or constraints, continue with Step 2.

Result: Ranking of the constraints.

In the first step, all known variables in the set $\mathcal{K}$ are marked and all unknown variables remain unmarked. In the second step, every constraint that contains at most one unmarked variable is assigned rank 1. It is matched for the unmarked variables (or for ZERO, if there is none), and the variable is marked. This step is repeated with an increasing rank number, until no new variables can be matched.

If every matched variable is also given a number, the rank can be interpreted as the number of steps needed to calculate the corresponding variable from the known ones.

The ranking algorithm stops before a complete matching is obtained if there exist unmarked variables still to be determined, but if all constraints include more than one unmarked variable. This situation occurs, for example, if two constraints have to be used simultaneously to determine two variables (e. g. the constraints c_1 and c_2 for the variables x_1 and x_2 in Example 5.17).

Example 5.25 Ranking of constraints for the single-tank system
The ranking algorithm is applied to the tank example as follows. As $u,\ y$ are known, only the variable set $\{q_i,\ q_o,\ h,\ \dot{h}\}$ has to be matched.

Starting set (rank 0): $\{u,\ y\}$
First step (rank 1): match q_i with c_2, match h with c_4
Second step (rank 2): match q_0 with c_3, match $\dot{h}$ with c_6
End (every variable is matched)

The obtained matching is the following one:

↗	h	$\dot{h}$	q_i	q_o	u	y
c_1		1	1	1		
c_2			①		1	
c_3	1			①		
c_4	①					1
c_5					1	1
c_6	1	①				

Hence, the ranking algorithm can be used to get a complete matching for the tank system. □

Example 5.26 Two-tank system
The two-tank system introduced in Sect. 2.1 will be considered with u as the known control input and q_{m} as the measured outflow. The following equations lead to the structure graph in Fig. 5.24:

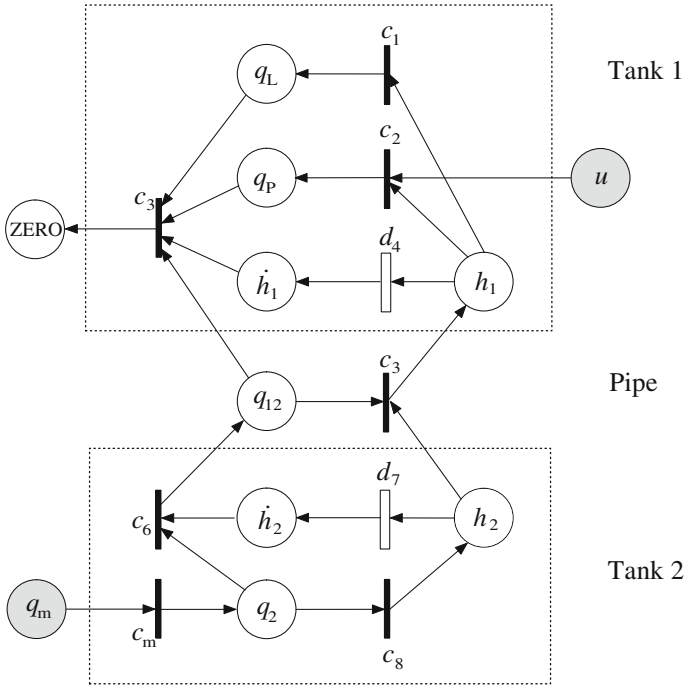


Fig. 5.24 Structure graph of the two-tank system

$$
\begin{aligned}
c_1: &\quad q_\mathrm{L} = c_\mathrm{L}\sqrt{h_1} \\
c_2: &\quad q_\mathrm{P} = u \cdot f(h_1) \\
c_3: &\quad \dot{h}_1 = \frac{1}{A}\,(q_\mathrm{P} - q_\mathrm{L} - q_{12}) \\
d_4: &\quad \dot{h}_1 = \frac{\mathrm{d}}{\mathrm{d}t}\,h_1 \\
c_5: &\quad q_{12} = k_1\sqrt{h_1 - h_2} \\
c_6: &\quad \dot{h}_2 = \frac{1}{A}\,(q_{12} - q_2) \\
d_7: &\quad \dot{h}_2 = \frac{\mathrm{d}}{\mathrm{d}t}\,h_2 \\
c_8: &\quad q_2 = k_2\sqrt{h_2} \\
m_1: &\quad q_\mathrm{m} = k_\mathrm{m}\,q_2 \\
m_2: &\quad q_{m12} = k_\mathrm{m}\,q_{12}
\end{aligned}
$$

A, k_1, k_2 and k_m are known parameters. c_L is the unknown parameter describing the size of the fault. It can be assumed to be zero for the faultless case. In the structure graph the constraints c_1, c_2, c_3 and d_4 representing the Tank 1 are separated from constraints c_6, d_7, c_8 and c_m describing the Tank 2.

The following matching is found using the ranking algorithm, where the last column shows the rank of the constraints obtained.

$$
\begin{aligned}
q_\mathrm{L} &= c_\mathrm{L}\sqrt{h_1} \\
q_\mathrm{P} &= u \cdot f(h_1) \\
0 &= -q_\mathrm{L} + q_p - q_{12} - A\dot{h}_1 \\
\dot{h}_1 &= \frac{\mathrm{d}}{\mathrm{d}t}\,h_1 \\
h_1 &= h_2 + \left(\frac{q_{12}}{k_1}\right)^2 \\
q_{12} &= A\,\dot{h}_2 + q_2 \\
\dot{h}_2 &= \frac{\mathrm{d}}{\mathrm{d}t}h_2 \\
h_2 &= \left(\frac{q_2}{k_2}\right)^2 \\
q_2 &= \frac{\mathrm{m}}{k_\mathrm{m}}
\end{aligned}
$$

↗	q_L	q_P	$\dot{h}_1$	h_1	q_{12}	$\dot{h}_2$	h_2	q_2	R
c_1	①			1					5
c_2		①		1					5
c_3	1	1	1		1				6
d_4			①	1					5
c_5				①	1		1		4
c_6					①	1		1	3
d_7						①	1		2
c_8							①	1	1
c_m								①	0

The matching obtained can alternatively be represented as follows:

	c_1	c_2	c_3	d_4	c_5	c_6	d_7	c_8	m_1	c_m
1	q_L	q_P	0	$\dot{h}_1$	h_1	$\dot{h}_2$	0	h_2	q_{12}	q_2

□

As the ranking algorithm may stop when encountering strongly connected subgraphs, more generic approaches to matching are introduced below.

5.5.2 General Matching Algorithm

Let $\mathcal{M}$ be a matching on a graph $\mathcal{G}$. An edge is said to be *weak* with respect to $\mathcal{M}$ if it does not belong to $\mathcal{M}$. A vertex is *weak* with respect to $\mathcal{M}$ if it is only incident to weak edges. An $\mathcal{M}$-*alternating path* is a path whose edges are alternating in $\mathcal{M}$ and not in $\mathcal{M}$ (or conversely). An $\mathcal{M}$-*augmenting path* is an alternating path whose end vertices are both weak with respect to $\mathcal{M}$. An $\mathcal{M}$-*alternating tree* with root v is a collection of disjoint $\mathcal{M}$-*alternating paths* with the common root v.

The basic matching algorithm is built on the following theorem:

Theorem 5.2 (BERGE 1957) *A matching $\mathcal{M}$ in a graph $\mathcal{G}$ is maximum if and only if there exists no $\mathcal{M}$-augmenting path in $\mathcal{G}$.*

The idea of the proof of the theorem is that if an augmenting path would exist, a new matching of size $|\mathcal{M}| + 1$ would be obtained by exchanging the roles of the matched and non-matched edges in the path, as illustrated by the following example. This step is called the *transfer* from the old to the new matching along the $\mathcal{M}$-augmenting path.

Example 5.27 An $\mathcal{M}$-augmenting path
A matching $\mathcal{M}$ of size 3 is given by the bold edges in the bipartite graph of Fig. 5.25 (left).

It can be checked that there exists an $\mathcal{M}$-*augmenting path*, namely

$$\underbrace{c_1 - x_1}_{\text{weak}} - \underbrace{x_1 - c_2}_{\text{matched}} - \underbrace{c_2 - x_3}_{\text{weak}} - \underbrace{x_3 - c_3}_{\text{matched}} - \underbrace{c_3 - x_4}_{\text{weak}} - \underbrace{x_4 - c_6}_{\text{matched}} - \underbrace{c_6 - x_5}_{\text{weak}}$$

and, therefore, this matching is not maximum. By exchanging weak (dashed lines) and matched (solid line) edges, the following matching of size 4 is found:

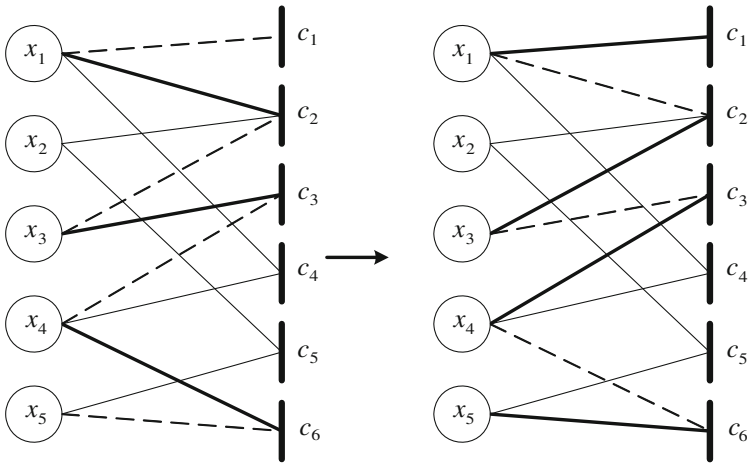


Fig. 5.25 Finding a new matching by using an augmenting path

$$\underbrace{c_1 - x_1}_{\text{matched}} - \underbrace{x_1 - c_2}_{\text{weak}} - \underbrace{c_2 - x_3}_{\text{matched}} - \underbrace{x_3 - c_3}_{\text{weak}} - \underbrace{c_3 - x_4}_{\text{matched}} - \underbrace{x_4 - c_6}_{\text{weak}} - \underbrace{c_6 - x_5}_{\text{matched}} . \square$$

Based on the theorem above, the following algorithm extends an initially given matching step-by-step by finding an augmenting path and augmenting the size of the current matching by transferring it, until no further augmenting path can be found and, therefore, the latest determined matching is maximum.

Algorithm 5.2 *Algorithm for finding a maximum matching*

Given: A bipartite graph $\mathcal{G}$ and an initial matching $\mathcal{M}_0$.

1. Let $\mathcal{M}$ be the current matching. If the number of weak vertices with respect to $\mathcal{M}$ is less than or equal to one, the current matching is maximum. Otherwise, let v be any weak vertex. Build an alternating tree with root v.

2. If the tree contains an $\mathcal{M}$-*augmenting path* then perform a transfer along this path and update the matching on the initial graph. Go back to 1.

Result: A maximum matching.

The initial matching can be the empty matching $\mathcal{M}_0 = \{\}$. In Step 2, the cardinality of the matching is increased by one.

Example 5.28 Maximum matching algorithm
Let $\mathcal{M}_0 = \{(x_1, c_2), (x_3, c_3), \ (x_4, c_6)\}$ be the initial matching shown on Fig. 5.25. The weak edges are

$$\{(x_1, c_1), (x_1, c_4), (x_2, c_2), (x_2, c_5), (x_3, c_2), (x_4, c_3), (x_4, c_4), (x_5, c_5), (x_5, c_6)\} .$$

There are four weak vertices, namely $\{c_1, x_2, c_4, c_5\}$. For the first iteration, choosing c_1 as the root gives the alternating tree shown on Fig. 5.26a where the current matching is shown in dashed lines. It is easily seen that there are two $\mathcal{M}_0$-augmenting paths, namely $c_1 - x_1 - c_2 - x_2$ and $c_4 - x_4 - c_6 - x_5$. Since these paths are disjoint, the two transfers can be done simultaneously, resulting in the matching

$$\mathcal{M}_1 = \{(x_1, c_1), (x_2, c_2), (x_4, c_4), (x_5, c_6)\} .$$

The weak edges are now

$$\{(x_1, c_2), (x_1, c_4), (x_2, c_5), (x_3, c_2), (x_3, c_3), (x_4, c_3), (x_4, c_6), (x_5, c_5)\}$$

and the weak vertices are $\{x_3, c_3, c_5\}$. Choosing c_3 as the root results in the alternated tree of Fig. 5.26b, which exhibits the $\mathcal{M}_1$-augmenting path $x_3 - c_2 - x_2 - c_5$. Performing the transfer gives the new matching

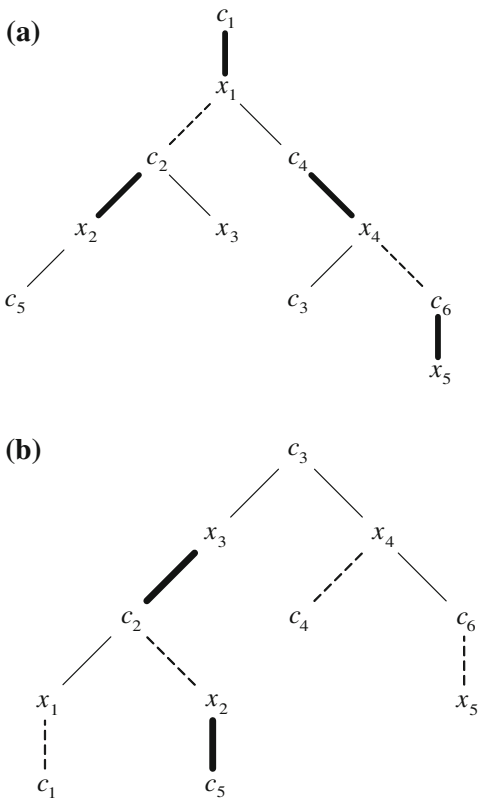


Fig. 5.26 Alternating tree with root c_1 (**a**) and with root c_3 (**b**)

$$\mathcal{M}_2 = \{(x_1, c_1), (x_3, c_2), (x_2, c_5), (x_4, c_4), (x_5, c_6)\}$$

which is maximum, because the set of weak edges is now

$$\{(x_1, c_2), (x_1, c_4), (x_2, c_2), (x_3, c_3), (x_4, c_3), (x_4, c_6), (x_5, c_5)\}$$

and there remains only one single weak vertex c_3. □

Example 5.29 Application to the single-tank system
The aim is to search for a maximum matching with respect to the reduced structure graph of the single-tank system.

↗	h	$\dot{h}$	q_i	q_o	u	y
c_1		1	1	1		
c_2			1		1	
c_3	1			1		
c_4	1					1
c_5					1	1
c_6	1	1				

Let $\mathcal{M}_0 = \{\}$. Then all vertices are weak. Selecting, e.g. h as the root gives the alternated tree

$$\begin{array}{lllllllllll} & \nearrow & c_6 & \to & \dot{h} & & & & & & \\ h & \longrightarrow & c_4 & & & & & & & & \\ & \searrow & c_3 & \to & q_o & \to & c_1 & \to & q_i & \to & c_2, \end{array}$$

where two disjoint $\mathcal{M}_0$-augmenting paths are given by $c_6 - \dot{h}$ and $h - c_3 - q_o - c_1 - q_i - c_2$ providing the new matching $\mathcal{M}_1 = \left\{\left(\dot{h}, c_6\right), (h, c_3), (q_o, c_1), (q_i, c_2)\right\}$ which is complete with respect to the unknown variables and, hence, the algorithm ends. □

5.5.3 Maximum Flow Algorithm

Finding a maximum matching in a bipartite graph can be transformed into a maximum flow problem. The procedure is as follows: Construct a network $\mathcal{N}$ associated with the graph $\mathcal{G} = (\mathcal{C}, \mathcal{Z}, \mathcal{E})$ by orienting all edges from $\mathcal{Z}$ to $\mathcal{C}$, by inserting a source vertex S with arcs to all vertices of $\mathcal{Z}$ and a sink vertex T with arcs from all vertices of $\mathcal{C}$, and by connecting T to S as shown on Fig. 5.27. Furthermore, assign the capacity of all arcs from S to $\mathcal{Z}$ and from $\mathcal{C}$ to T as 1. The capacities of all other arcs are set to ∞. Then, the maximum flow on $\mathcal{N}$ is associated with a maximum matching as stated in the following theorem.

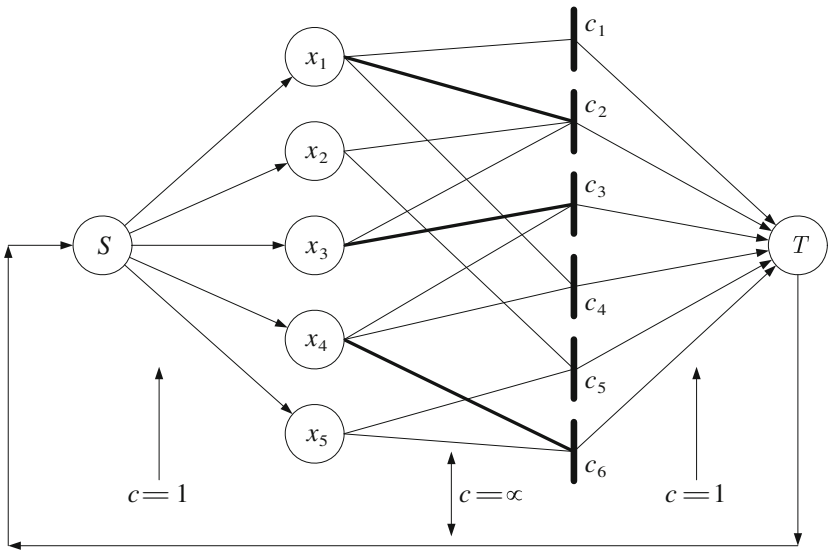


Fig. 5.27 Setting the maximum matching problem as a maximum flow problem. Flow in edges from $\mathcal{Z}$ to $\mathcal{C}$ is 1 if an edge is used (labelled), otherwise 0

Theorem 5.3 *In a bipartite graph $\mathcal{G}$, the matching number $\nu(\mathcal{G})$ equals the maximum flow through the network $\mathcal{N}$ that is associated to the graph $\mathcal{G}$.*

Therefore, a maximum matching can be found by applying the classical maximum flow algorithm of FORD and FULKERSON, which in the case of bipartite graphs is called "the Hungarian method". Like the preceding algorithm, this algorithm assumes a given matching $\mathcal{M}$ is known and attempts to extend $\mathcal{M}$ by finding an augmenting path. This is done by marking vertices on weak edges so as to follow possible augmenting paths.

Algorithm 5.3 *Hungarian method for determining maximum matchings*

Given: A bipartite graph and an initial matching $\mathcal{M}_0$.

1. Denote the current matching by $\mathcal{M}$. Label with an * all vertices of $\mathcal{Z}$ that are weak with respect to $\mathcal{M}$, and alternately apply Steps 2 and 3 until no further labelling is possible.

2. Select a newly labelled vertex in $\mathcal{Z}$, say z_i, and label with z_i all unlabelled vertices of $\mathcal{C}$ that are connected to z_i by an edge that is weak with respect to $\mathcal{M}$. Repeat this step on all vertices of $\mathcal{Z}$ that were labelled in the previous step.

3. Select a newly labelled vertex of $\mathcal{C}$, say c_j and label with c_j the vertex of $\mathcal{Z}$ which is connected to c_j in $\mathcal{M}$. Repeat this process on all vertices of $\mathcal{C}$ labelled in the previous step.

4. The labellings will continue to alternate until one of two possibilities occurs:

END 1: A weak vertex $\mathcal{C}$ has been labelled. Then an $\mathcal{M}$-augmenting path has been found, and it can be constructed by working backwards through the labels until the vertex of $\mathcal{Z}$ which is labelled by a *. Transferring this path gives an extended matching and the algorithm is repeated by going back to Step 2.

END 2: It is not possible to label more vertices and END 1 has not occured. Then $\mathcal{M}$ is a maximum matching.

Result: A maximum matching.

Example 5.30 Determination of a maximum matching by the Hungarian method
Let $\mathcal{M}_0 = \{(x_1, c_2), (x_3, c_3), (x_4, c_6)\}$ be the initial matching shown on Fig. 5.25. The table below shows the initial labelling and the sequence of labels obtained as Steps 2 and 3 alternate. The bar over vertices c_1, c_4 and c_5 indicate that these vertices are weak with respect to the current matching.

	x_1	x_2	x_3	x_4	x_5	$\bar{c}_1$	c_2	c_3	$\bar{c}_4$	$\bar{c}_5$	c_6
Step 1		*			*						
Step 2-1							x_2			x_2	x_5
Step 3-1	c_2			c_6							
Step 2-2						x_1		x_4	x_1		
Step 3-2			c_3								

The table demonstrates the Steps 2 and 3 of the algorithm. However, in this example, the first iteration would stop as early as after Step 2-1 because the weak vertex c_5 has been labelled (END 1). Tracking the labels backwards until a *-vertex is found gives the $\mathcal{M}_0$-augmenting path $c_5 - x_2$, which results in the augmented matching

$$\mathcal{M}_1 = \{(x_1, c_2), (x_2, c_5), (x_3, c_3), (x_4, c_6)\}$$

from which the next iteration starts. □

Example 5.31 Hungarian method applied to the single-tank system
With $\mathcal{M}_0 = \{(\dot{h}, c_6)\}$ being the initial matching, the first iteration gives the labels

	h	$\dot{h}$	q_i	q_o	$\bar{c}_1$	$\bar{c}_2$	$\bar{c}_3$	$\bar{c}_4$	$\bar{c}_5$	c_6
Step 1	*		*	*						
Step 2					q_i	q_i	h	h		h

before END 1 occurs and the matching can be updated as

$$\mathcal{M}_1 = \{(\dot{h}, c_6), (h, c_3), (q_i, c_1)\}.$$

The second iteration gives

	h	$\dot{h}$	q_i	q_o	c_1	$\bar{c}_2$	c_3	$\bar{c}_4$	$\bar{c}_5$	c_6
Step 1				*						
Step 2-1					q_0		q_0			
Step 3-1	c_3		c_1							
Step 2-2						q_i		h		h

before END 1 occurs. The $\mathcal{M}_1$-augmenting path is given by $c_4 - h - c_3 - q_0$ and leads to the new matching

$$\mathcal{M}_2 = \{(\dot{h}, c_6), (h, c_4), (q_i, c_1), (q_o, c_3)\}.$$

This matching is complete with respect to the unknown variables and stops the algorithm. Note that the solution is different from the one previously found, which illustrates the fact that maximum matchings are not unique. □

The above algorithm is computationally very heavy and experience showed that alternative algorithms were needed to cope with industrial scale systems.

5.5.4 Minimal Over-Determined Subsystems Approach

The Minimal Structurally Over-determined (MSO) set approach offers another way to find all analytical redundancy relations. The idea in the method is to calculate all subsets $\mathcal{M}_{\text{MSO}} \subseteq \mathcal{S}^+$ of an over-constrained structure graph, which have exactly one constraint more than the just-constrained subsystem. The structural redundancy measure for such subset is $\varrho(\mathcal{M}_{\text{MSO}}) = 1$ according to Eq. (5.22). Therefore, each MSO set will comprise at least one constraint that can be used as an ARR. The number of ARRs generated in this way will be larger than the set of ARRs found from a single complete matching, but less or equal to the number of ARRs generated by the brute-force approach of generating all possible complete matchings and get a set of ARRs for each of these matchings.

The reason to generate more than the minimal set of ARRs available from a single complete matching is that structural isolability can be enhanced by considering more than the minimal number of ARRs.

Definition 5.8 (*Minimal structurally over-constrained subsystem*) A minimal structurally over-determined subsystem (MSO subsystem) is a part of the over-constrained part of a system graph from which removal of one constraint will make the subsystem to become just-constrained.

The procedure to find MSO sets is based on examining the set $\mathcal{M}$ of constraints of a proper structurally over-constrained structure graph. The PSO property means $\mathcal{M} = \mathcal{C}^+$. Denoting the set of unknown variables in $\mathcal{X}$ that are connected to at least one constraint in $\mathcal{M}$ by $Q(\mathcal{M})$, then

$$\varrho(\mathcal{M}) = |\mathcal{M}^+| - |Q(\mathcal{M}^+)|$$

(cf. Eq. (5.22)). Removing one constraint c_i from the set $\mathcal{M}$ reduces the structural redundancy by one,

$$\varrho(\mathcal{M} \backslash \{c_i\}) = \varrho(\mathcal{M}) - 1.$$

The set of constraints $\mathcal{M}$ is an MSO set if $\mathcal{M}$ is PSO and $\varrho(\mathcal{M}) = 1$.

These observations led to a computationally very efficient way to determine the set of all possible ARRs for a system. The following computational procedure is used recursively [187]:

Algorithm 5.4 *Determination of the set of all ARRs*

Require $\mathcal{M}$ is a PSO set
Procedure $\mathcal{M}_{\text{MSO}}$:= findmso($\mathcal{M}$)
if $\varrho(\mathcal{M}) = 1$ **then**
$\mathcal{M}_{\text{MSO}} := \{\mathcal{M}\}$;
else
$\mathcal{M}_{\text{MSO}} := \emptyset$;
for any $c_i \in \mathcal{M}$
$\mathcal{M}' := (\mathcal{M} \setminus \{c_i\})^+$;
$\mathcal{M}_{\text{MSO}} = \mathcal{M}_{\text{MSO}} \cup$ findmso($\mathcal{M}'$);
end
endif
return $\mathcal{M}_{\text{MSO}}$
end

This procedure is used in the following algorithm to determine all MSO sets in a structure graph.

Algorithm 5.5 *Determination of minimal structurally over-determined sets*

Given: Reduced structure graph $\mathcal{S}$ of a system

1. Using DM decomposition, select the constraints that form the part of $\mathcal{S}$ which is PSO. Denote this set of constraints by $\mathcal{M}$
2. Perform a complete search using the recursive procedure $\mathcal{M}_{\text{MSO}}$:= findmso($\mathcal{M}$)

Result: Set $\mathcal{M}_{\text{MSO}}$ of all possible MSO sets

The above algorithm finds MSO sets more than once. This can be avoided by finding *equivalence classes* of constraints and make an extension of the basic algorithm that is described in [187].

Example 5.32 Determination of MSO sets
This example shows how MSO sets and thereby ARRs are generated for the single-tank example with extended measurements. Assume that the flow q_o is measured in addition to the input u and the inflow q_i, which leads to an additional measurement constraint

$$c_7 : \quad y_2 = q_o.$$

↗	h	$\dot{h}$	q_i	q_o	u	y_1	y_2
c_1		1	1	1			
c_2			1		1		
c_3	1			1			
c_4	1					1	
c_6	1	1					
c_7				1			1

This structure graph is over-constrained with $\varrho(\mathcal{M}) = 2$.

Algorithm 5.5 determines the four MSO sets listed below.

	c_1	c_2	c_3	c_4	c_6	c_7
$\mathcal{M}_1$	-	-	0	h	-	q_o
$\mathcal{M}_2$	$\dot{h}$	q_i	-	h	0	q_o
$\mathcal{M}_3$	$\dot{h}$	q_i	h	-	0	q_o
$\mathcal{M}_4$	$\dot{h}$	q_i	q_o	h	0	-

The table has to be interpreted as follows: The MSO set $\mathcal{M}_1$ includes the constraint $c_3(h, q_o) = 0$ as an ARR and uses c_4 to calculate h and c_7 to calculate q_o. Each of the MSO sets is, by the definition of the MSO subsystem, also an ARR.

Algorithm 5.5 finds four MSO sets for this example. By comparison, the Ranking Algorithm 5.1 finds one complete matching of the unknown variables and two ARRs, c_1 and c_3, according to the following matching table:

↗	h	$\dot{h}$	q_i	q_o	u	y_1	y_2
c_1		1	1	1			
c_2			①		1		
c_3	1			1			
c_4	①					1	
c_6	1	①					
c_7				①			1

□

5.6 Structural Diagnosability and Isolability

A system is said to be *structurally diagnosable* or *monitorable* if it is possible to test whether the system constraints are satisfied or not. This section is concerned with the analysis of system monitorability and with fault detection and isolation algorithms based on *Analytical Redundancy Relations* (ARRs).

Analytical redundancy occurs and analytical redundancy relations become available when there are constraints that are not needed to match the unknown variables in a system. These additional constraints, as well as all others, need be satisfied when the system obeys normal behaviour, so the additional, or redundant, constraints can test whether the system behaviour is normal. Violation of a constraint that is used to calculate an ARR would leave the ARR as not satisfied.

Residuals are derived from ARRs. A residual will only depend on known variables but the ARR might not represent causal computations. The analytical form of a residual is a signal $r(t)$ that can be calculated by causal operations in real time by inserting the instantaneous values of known variables: the input $u(t)$ and measurements $y(t)$. A residual signal $r(t)$ is therefore obtained from the corresponding $arr(t)$ through filtering of the entire $arr(t)$.

The terms *analytical redundancy relation* and *residual generator* are often used as synonyms in the literature, although strictly speaking, the ARRs are found without any consideration to stability and causality while a residual generator needs to be both stable and causal to generate a signal $r(t)$ that has the properties needed for fault diagnosis. This is further elaborated in Chap. 6.

Analytical redundancy-based fault diagnosis tries to identify faults by comparing the actual behaviour of the system, which is observed through the time evolution of the known variables, with the behaviour described by the system constraints. This comparison can be performed only if some redundant information exists. For diagnosis it is not sufficient that the known variables and the set of constraints allow to determine all unknown variables. There must be available at least one constraint more with which one can test whether the obtained variables are consistent with the model representing the faultless behaviour of the system. ARRs are the constraints that express this redundancy.

In this section, the analytical redundancy relation-based approach to fault diagnosis is first briefly recalled and stated in the frame of structural analysis, leading to characterise the structurally monitorable part of the system. Finding residuals that are robust, meaning they are insensitive to disturbances or to unknown parameters, are then discussed and residuals that are sensitive to certain structural faults, but not to others (structured residuals) are then addressed.

5.6.1 Analytical Redundancy-Based Fault Detection and Isolation

Analytical redundancy relations are static or dynamical constraints that will be satisfied (equal to zero) when the system operates according to its normal operation model. Once ARRs are found, the fault detection procedure checks whether they are satisfied or not, and if not, the fault isolation procedure identifies the system components which are to be suspected. The existence of ARR is thus a prerequisite to the

elaboration of fault diagnosis procedures. Moreover, in order for the fault diagnosis procedure to work properly, ARR should have the following properties:

- *Robust*, i.e. insensitive to unknown input and unknown parameters. This insures that they are satisfied when no fault is present, so that false alarms are not issued.
- *Sensitive to faults*: This insures that they are not satisfied when a constraint is violated, i.e. a fault is present, so that faults are detected.
- *Structured*: This insures that in the presence of a given fault, only a subset of the ARRs is not satisfied, thus allowing to recognise the fault that occurred from the subset of ARRs that are satisfied and the subset that is not satisfied.

Faults. In structural analysis, a fault is defined as *a violation in a constraint*. A system is the interconnection of a number of components, each of which is described by its behavioural model in normal operation. Let $\{C_i, i = 1, 2, \ldots, N\}$ be the set of the system components. Each of them is a subsystem $(\phi_i,\ Q(\phi_i))$ which imposes the set of constraints ϕ_i to the system variables $Q(\phi_i)$, where $Q(\phi_i) \cap \mathcal{X}$ are unknown (unmeasured state variables, unknown input, unknown parameters) while $Q(\phi_i) \cap \mathcal{K}$ are known (input, output, known parameters). A fault in component C_i is defined as a change in at least one of the constraints $\varphi \in \phi_i$.[3] Note that this general definition of faults allows to consider different fault modes associated with the same component. Each subset of ϕ_i can in fact be considered as a fault mode of C_i. Note also that since only the structure is of interest, there is no need to define, nor to model the nature of the change (e.g. using additive or multiplicative fault models).

Example 5.33 Representation of faults in an insulated pipe
Consider an insulated pipe and suppose that one is interested in modelling the mass and the heat transfers. A simple model is given by the two constraints

$$\begin{aligned} \varphi_1 &: q_i(t) - q_o(t) = 0 \\ \varphi_2 &: q_i(t)\,\theta_i(t) - q_o(t)\,\theta_o(t) = 0, \end{aligned}$$

where q_i and q_o are the input and the output flow of the (incompressible) fluid, and θ_i (respectively, θ_o) is the input (respectively, the output) fluid temperature. A defect in the insulation would obviously result in φ_2 being violated, while a leak in the pipe would be modelled by φ_1 and φ_2 being violated. □

Direct redundancy. Consider a constraint $\varphi \in \mathcal{C}_\mathcal{K}$, where $\mathcal{C}_\mathcal{K}$ is the subset of constraints such that $Q(\mathcal{C}_\mathcal{K}) \subseteq \mathcal{K}$ and let C be the component to which φ belongs. This constraint is an ARR because it links only known variables, and it can be checked in real time if it is satisfied or not, by taking the numerical values of the known variables, putting them into constraint φ, and testing whether the result is ZERO or not. If the constraint is not satisfied, it can be concluded that the system is not

[3]The notation φ is used—as a mnemonic for "fault"—instead of c which was a mnemonic for "constraint".

in normal operation, while if the constraint is satisfied it can only be said that the normal operation hypothesis is not falsified by the values of the observations.

In practical situations, variables are not very precisely known, measurements are corrupted by noise, and models only approximate the system behaviour. Thus, the obtained value for the constraint will never be exactly zero, even in normal operation. Let $r_\varphi(\mathcal{K})$ be the obtained value. $r_\varphi(\mathcal{K})$ is called the *residual* associated with ARR φ, and fault detection boils down to decide whether the residual is small enough so that the ZERO hypothesis can be accepted. Fault isolation obviously follows fault detection because only a fault in component C could cause constraint φ not to be satisfied.

In all systems, the control algorithms are direct ARRs, because the subset $\mathcal{C}_\mathcal{K}$ includes the constraints which describe them. Hence, they can be used to check whether the controller is working properly. Although this might be of practical interest, such direct redundancy relations are of little interest as far as structural analysis is concerned, because the result is obvious. Therefore, the aim of the following part of this chapter is to find ARRs in the subsystem $(\mathcal{C}_\mathcal{X}, \mathcal{Z})$ which includes unknown variables.

Deduced redundancy. Consider some constraint $\varphi \in \mathcal{C}_\mathcal{X}$ and again let C be the component to which φ belongs. Let $\mathcal{X}_\varphi = Q(\varphi) \cap \mathcal{X}$ be the subset of unknowns which appear in constraint φ, and suppose that

$$\mathcal{X}_\varphi \subseteq \mathcal{X}_{\text{obs}} \tag{5.27}$$

holds, where $\mathcal{X}_{\text{obs}}$ is the subset of the observable variables. Then, any variable $x \in \mathcal{X}_\varphi$ can be expressed as a function of the known ones (possibly including their derivatives) using the model. Suppose that there exists at least one alternated chain with target x which does not include constraint φ. This means that even if constraint φ is removed, x can still be matched and computed as a function of the known variables, which indicates that constraint φ belongs to an over-constrained subsystem, as it will be seen later. Then, this alternated chain can be used to compute x as a function of the known variables, and one can put the obtained expression into φ, which produces an ARR. The associated residual $r_\varphi(\mathcal{K})$ should be ZERO when the system operates properly.

However, fault isolation will be slightly different because the residual associated with φ will be non-zero not only if C is not performing well, but also if the actual values of the $\mathcal{X}_\varphi$ variables are different from those computed from the observations via the normal operation model. This may happen when the fault changes some constraint which belongs to an alternated chain whose target is in $\mathcal{X}_\varphi$. The conclusion is that when $r_\varphi(\mathcal{K})$ is non-zero, there is an associated set of components to be suspected instead of a single one.[4] It can be easily determined from the graph-based interpretation.

[4]This set is called the *structure of the residual* in the control community and it is called a *conflict* in the Artificial Intelligence community.

Example 5.34 Single-tank system
Consider the tank whose structure graph is shown in Fig. 5.3. There are two redundancy relations for this system. The first one is given by constraint c_5 and is of no interest because it is a direct redundancy relation which only duplicates the control algorithm. The second one is given by c_2 which should be satisfied when the system operates normally and which will be false if one of the constraints $\{c_1,\ c_2,\ c_3,\ c_4\}$ is not satisfied (c_6 is a mathematical constraint which is not linked with any hardware or software component and thus it cannot be faulty). □

5.6.2 Structurally Monitorable Subsystems

Unfortunately, not every fault can be detected. Therfore, it is important to find ways for distinguishing diagnosable faults or diagnosable subsystems from undiagnosable ones. Such ways will be described in this subsection.

Definition 5.9 (*Structurally monitorable subsystem*) The structurally diagnosable (monitorable) part of the system is the subset of the constraints for which there exists ARRs that are structurally sensitive to their change.

Such subsystems can be characterised by the following theorem:

Theorem 5.4 (Structural monitorability) *The following two necessary conditions for a fault φ to be structurally diagnosable (monitorable) are equivalent:*

(i) $\mathcal{X}_\varphi$ *is structurally observable—according to (5.27)—in the system* $(\mathcal{C}\backslash\{\varphi\},\ \mathcal{Z})$.
(ii) φ *belongs to the structurally observable over-constrained part of the system* $(\mathcal{C},\ \mathcal{Z})$.

Let $(\mathcal{C}_\mathcal{X},\ \mathcal{X})$ be a structurally observable over-constrained subsystem. Then there exists a subset $\mathcal{S}_\mathcal{X} \subset \mathcal{C}_\mathcal{X}$ of $n = |\mathcal{X}|$ constraints which (from a structural point of view) can be solved uniquely for the variables $\mathcal{X}$.[5] These variables can thus be computed as functions of the known variables $\mathcal{K}$. Putting the obtained values into the remaining constraint set $\mathcal{R}_\mathcal{X} = \mathcal{C}_\mathcal{X}\backslash\mathcal{S}_\mathcal{X}$ (the symbol $\mathcal{R}$ is used as a mnemonic for Remaining, or Redundant), one obtains $|\mathcal{C}_\mathcal{X}| - |\mathcal{X}|$ relations which link only known variables and which are, therefore, redundancy relations. For a more convenient notation the function

$$\mathcal{X} = \Gamma_\mathcal{X}(\mathcal{K}) \tag{5.28}$$

is introduced for the computation of the unknown variables, leading to expressions for the set of constraints $\mathcal{C}_\mathcal{X}$ in the equivalent form

$$\begin{aligned} \mathcal{S}_\mathcal{X}:&\quad \mathcal{X} - \Gamma_\mathcal{X}(\mathcal{K}) = 0 \\ \mathcal{R}_\mathcal{X}:&\quad (\mathcal{C}_\mathcal{X}\backslash\mathcal{S}_\mathcal{X}) \circ \Gamma_\mathcal{X}(\mathcal{K}) = 0, \end{aligned} \tag{5.29}$$

[5]The symbol $\mathcal{S}$ is used as a mnemonic for "solve".

where $\circ$ means the substitution of $\mathcal{X}$ by $\Gamma_{\mathcal{X}}(\mathcal{K})$.

In general, several different complete matchings can be found in a given causal over-constrained subsystem, which lead to different means of computing the unknown variables $\mathcal{X}$ from the known ones. This fact will be used for the elaboration of fault-tolerant observation schemes but it can also provide another interpretation of redundancy, since obviously the unknown variables $\mathcal{X}$ have to be the same for all matchings. For example, suppose that two matchings exist such that $\mathcal{X}$ is associated with $\mathcal{S}_{\mathcal{X}} \subset \mathcal{C}_{\mathcal{X}}$ in the first one, leading to the relation $\mathcal{X} = \Gamma_{\mathcal{X}}(\mathcal{K})$, and with $\mathcal{P}_{\mathcal{X}} \subset \mathcal{C}_{\mathcal{X}}$ in the second one, leading to $\mathcal{X} = \Lambda_{\mathcal{X}}(\mathcal{K})$. The redundancy relations

$$\Gamma_{\mathcal{X}}(\mathcal{K}) - \Lambda_{\mathcal{X}}(\mathcal{K}) = 0$$

directly follow from the fact that the two results should be the same.

Example 5.35 Sensor redundancy
A good illustration of this idea is provided by sensor hardware redundancy. Suppose that two sensors measure the same unknown variable x. The measurement equations are given by

$$\begin{aligned} &\text{Sensor 1} \quad c_1: \quad y_1 - x - \varepsilon_1 = 0 \\ &\text{Sensor 2} \quad c_2: \quad y_2 - x - \varepsilon_2 = 0, \end{aligned}$$

where ε_1 and ε_2 denote measurement noise with known distribution. The structure graph has the following incidence matrix.

	known				unknown
$\nearrow$	y_1	y_2	ε_1	ε_2	x
c_1	1		1		1
c_2		1		1	1

Here, ε_1 and ε_2 are considered as known variables because their probability distribution is known. This system is over-constrained with $\mathcal{C}_{\mathcal{X}} = \{c_1, c_2\}$ and $\mathcal{X} = \{x\}$. The unknown x can be matched with each of the two constraints and, hence, be calculated by each of the sensor equations. This is not only true from the structural point of view but x can be determined numerically if $\frac{\mathrm{d}c_1}{\mathrm{d}x}$ and $\frac{\mathrm{d}c_2}{\mathrm{d}x}$ are both non-zero. Otherwise at least one of the sensors would be completely useless.

For the matching

	known				unknown
$\nearrow$	y_1	y_2	ε_1	ε_2	x
c_1	1		1		①
c_2		1		1	1

the oriented graph is given by Fig. 5.28, in which the unknown x is computed by

$$x = \gamma_1(y_1, \varepsilon_1)$$

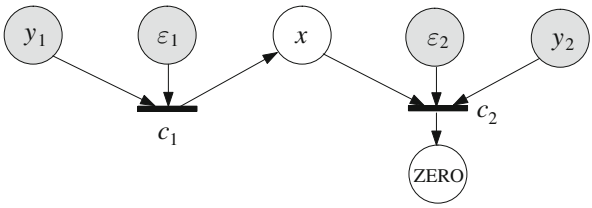


Fig. 5.28 Oriented structure graph for sensor monitoring

and c_2 is used as a redundancy relation which can be written as

$$c_2(\gamma_1(y_1, \varepsilon_1), y_2, \varepsilon_2) = 0.$$

Choosing the second possible matching

	known				unknown
↗	y_1	y_2	ε_1	ε_2	x
c_1	1		1		1
c_2		1		1	①

provides

$$x = \gamma_2(y_2, \varepsilon_2)$$

and the redundancy relation

$$c_1\,(y_1, \varepsilon_1, \gamma_2\,(y_2, \varepsilon_2)) = 0.$$

Since two matchings exist, the value of x can be computed either from the first or from the second one and leads to the redundancy relation

$$\gamma_1\,(y_1, \varepsilon_1) - \gamma_2\,(y_2, \varepsilon_2) = 0. \square$$

5.6.3 Finding Analytic Redundancy Relations

As explained in the preceding sections, redundancy relations are obtained from over-constrained subgraphs of the reduced structure graph. They are composed of alternated chains, which start with known variables and end with non-matched constraints whose output is labelled ZERO. Designing a set of residuals calls for building maximum matchings on the given structure graph, and identifying the redundancy relations as the non-matched constraints in which all the unknowns have been matched, and subsequently expressing the non-matched constraints by known variables through *backtracking* to known variables, according to the matching. This section gives a complete illustration of this procedure.

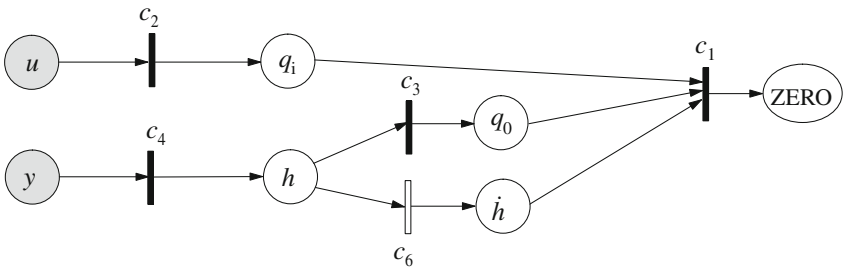


Fig. 5.29 Ranking for the single-tank system

Example 5.36 Finding an analytic redundancy relation for the single-tank system
For the single-tank example, the incidence matrix of its reduced structure graph was given in Example 5.9:

↗	h	$\dot{h}$	q_i	q_o
c_1		1	1	1
c_2			1	
c_3	1			1
c_4	1			
c_6	**x**	1		

The result of the ranking algorithm is shown in the following table and in Fig. 5.29. The matching is identical with the second matching in Example 5.9. Note that a new column has been introduced to mark constraints which have the output ZERO. Since ZERO is not a variable, it may be matched several times.

	unknown				Ranking	
↗	h	$\dot{h}$	q_i	q_o	ZERO	Rank
c_1		1	1	1	①	2
c_2			①			0
c_3	1			①		1
c_4	①					0
c_6	**x**	①				1

Sorted according to the rank, the following constraint set is obtained:

Rank	Constraint	Output
0	c_2	$q_i(t)$
	c_4	$h(t)$
1	c_3	$q_o(t)$
	c_6	$\dot{h}$
2	c_1	ZERO

If the reduced structure graph is redrawn according to the ranking of the constraints, Fig. 5.29 is obtained. The figure shows how the internal variables q_i, h, q_o and $\dot{h}$ can be successively determined. The constraints are ordered according to their associated rank. Finally, the constraint c_1 is used to test whether the variables obtained are consistent with the model.

As all constraints are ranked, the system is fully observable and monitorable. By solving the constraints for the matched variables, the following equations are obtained. The right-hand column shows the path of the matching.

$$\begin{array}{lll} c_2: & q_i(t) = \alpha \cdot u(t) & c_2(u) \rightarrow q_i \\ c_4: & h(t) = y(t) & c_4(y) \rightarrow h \\ c_3: & q_0(t) = k\sqrt{h(t)} & c_6(h) \rightarrow q_o \\ c_6: & \dot{h}(t) = \frac{\mathrm{d}}{\mathrm{d}t}h(t) & c_6(h) \rightarrow \dot{h} \\ c_1: & 0 = \dot{h}(t) + q_o(t) - q_i(t) & c_1(\dot{h}, q_i, q_o) \rightarrow \text{ZERO} \end{array} \tag{5.30}$$

These equations can be simplified to obtain the redundancy relations in one analytic expression:

$$c_1: \; 0 = \frac{\mathrm{d}}{\mathrm{d}t}y(t) + k\sqrt{y(t)} - \alpha u(t).$$

The order of operations on constraints was

$$c_1(c_6(c_4(y)), c_2(h), c_3(c_4(y))) \rightarrow \text{ZERO}.$$

As all variables on the right-hand side of the two equations are known, these equations can be applied to the known variables u and y, which are marked by grey circles in Fig. 5.29, to illustrate this fact. □

5.6.4 Structural Detectability and Isolability

Assume that the over-constrained subsystem has been determined by finding a complete matching on the unknown variables. Then, the main results of structural analysis are obtained from the following steps:

1. List all analytic redundancy relations that exist for the system.
2. For all these relations, determine an explicit form if the constraints are explicitly known.

3. List which violations of constraints are detectable.
4. List which violations of constraints are isolable.

Calculate residuals from structural analysis. After a matching has been found, the set $\mathcal{C}^{(u)} \subset \mathcal{C}$ of unmatched constraints

$$\mathcal{C}^{(u)} = \{c \,:\, c(x_c, k_c) \to 0, x_c \in \mathcal{X},\ k_c \in \mathcal{K}\}$$

is determined. To obtain analytical redundancy relations for diagnosis, also referred to as parity relations, the unknown variables in each $c \in \mathcal{C}^{(u)}$ must be substituted by known ones entering through matched constraints. Backtracking along alternated chains in the matching will facilitate such an elimination of the unknown variables. Finally, each unmatched constraint c will give one parity relation r to be used for diagnosis, and a violation of any constraint that was used in constructing the parity relation will give a non-zero residual when all known variables enter by their real-time values.

Furthermore, analytical redundancy relations show which residuals depend on which constraints. One view on these relations is the Boolean mapping, the *dependability matrix* or *signature matrix*,

$$M : c \to r$$

from which structural detectability can be analysed. It can be checked that the following definition is the practical translation of the monitorability condition in Theorem 5.6.

Lemma 5.1 (Structural detectability) *A violation of a constraint c is structurally detectable if and only if it has a non-zero Boolean signature in some residual r*

$$c \in \mathcal{C}_{\text{detectable}} \Leftrightarrow \exists r :\ c \neq 0 \ \Rightarrow\ r \neq 0.$$

Moreover, since for a given constraint c the set of all parity relations can be partitioned into those in which its Boolean signature is zero and those in which its Boolean signature is non-zero, the following result is straightforward.

Lemma 5.2 (Structural isolability) *A violation of a constraint c_i is structurally isolable if and only if it has a unique signature in the residual vector, i.e. column m_i of M is independent of all other columns in M*

$$c_i \in \mathcal{C}_{\text{isolable}} \Leftrightarrow \forall j \neq i : m_i \neq m_j.$$

Example 5.37 Nonlinear parity relations for ship
Consider the nonlinear model of a ship with dual measurements of heading angle ψ and with no disturbance from waves:

$$
\begin{aligned}
c_1 &: \ \dot{\omega}_3 &&= b(\eta_1\omega_3 + \eta_3\omega_3^3) + b\delta \\
c_2 &: \ \dot{\psi} &&= \omega_3 \\
d_1 &: \ \tfrac{\mathrm{d}\omega}{\mathrm{d}t} &&= \dot{\omega} \\
d_2 &: \ \tfrac{\mathrm{d}\psi}{\mathrm{d}t} &&= \dot{\psi} \\
m_1 &: \ y_1 &&= \psi \\
m_2 &: \ y_2 &&= \psi \\
m_3 &: \ y_3 &&= \dot{\psi} \\
m_4 &: \ y_4 &&= \delta.
\end{aligned}
$$

The set of unknown variables is $\mathcal{X} = \{\delta, \omega_3, \dot{\omega}_3, \psi, \dot{\psi}\}$, the set of known variables is $\mathcal{K} = \{y_1, y_2, y_3, y_4\}$. A complete matching on the unknown variables is traced in the left column below, the right column shows the backtracking to known variables.

$$
\begin{array}{lcl}
m_1(y_1) & \to & \psi \\
m_2(y_2, \psi) & \to & \text{ZERO} \ \Rightarrow \ m_2(y_2, m_1(y_1)) \to \text{ZERO} \\
m_3(y_3) & \to & \dot{\psi} \\
m_4(y_4) & \to & \delta \\
d_2(\psi, \dot{\psi}) & \to & \text{ZERO} \ \Rightarrow \ d_2(m_2(y_2), m_3(y_3)) \to \text{ZERO} \\
c_2(\dot{\psi}) & \to & \omega_3 \\
d_1(\omega_3) & \to & \dot{\psi} \\
c_1(\delta, \omega_3, \dot{\omega}_3) & \to & \text{ZERO} \ \Rightarrow \ c_1(m_4(y_4), c_2(m_3(y_3)), d_1(c_2(m_3(y_3)))) \to \text{ZERO}
\end{array}
\tag{5.31}
$$

The way constraints are used in the three parity relations as follows,

$$
\begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} \leftarrow
\begin{array}{c}
\begin{matrix} m_1 & m_2 & m_3 & m_4 & c_1 & c_2 & d_1 & d_2 \end{matrix} \\
\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}
\end{array}.
$$

As a violation of any constraint is mapped onto the residuals, all faults are detectable. Considering isolability, five columns are independent: m_1, m_2, m_3, d_2. Hence it is only violations in these constraints that are structurally isolable.

The matching obtained is summarised in condensed form in the following table:

	c_1	c_2	d_1	d_2	m_1	m_2	m_3	m_4
1	0	ω	$\dot{\omega}$	0	0	ψ	$\dot{\psi}$	δ

The detectability and isolability properties are conveniently summarised in tabular form as follows, where d and i denote structural detectability and isolability, n that a constraint cannot fail.

	c_1	c_2	d_1	d_2	m_1	m_2	m_3	m_4
1	d	n	n	n	i	i	i	d

The analytical form of the parity relations is obtained from the symbolic expressions from the backtracking. This gives the expected result,

$$\begin{aligned} r_1 &= y_2 - y_1 \\ r_2 &= \dot{y_2} - y_3 \\ r_3 &= \dot{y_3} - b(\eta_1 y_3 + \eta_3 y_3^3) - b y_4 \ . \ \square \end{aligned} \tag{5.32}$$

5.6.5 Design of Robust and Structured Residuals

Robust residuals. The set of constraints that describe the nominal operation of a system might fail to represent all aspects of its actual behaviour. Discrepancies follow from the existence of unknown inputs (disturbances) and from the fact that system parameter values are never exactly known (uncertain parameters). Such discrepancies might result in residuals firing false alarms.

Example 5.38 Residual discrepancies caused by unknown inputs
The unknown variables of the single-tank system were computed from the following constraints:

Component	Constraint	Constraint expression
Pump	c_2 :	$q_i(t) = \alpha \cdot u(t)$
Level sensor	c_4 :	$h(t) = y(t)$
Output pipe	c_3 :	$q_0(t) = k\sqrt{h(t)}$
Derivative constraint	c_6 :	$\dot{h}(t) = \frac{\mathrm{d}}{\mathrm{d}t} h(t)$

Putting these expressions into the constraint

$$c_1 : \dot{h}\,(t) - q_i(t) + q_o\,(t) = 0,$$

the residual

$$r(t) = \dot{y}(t) + k\sqrt{y(t)} - \alpha u(t)$$

is obtained. Assume that the level sensor output is affected by a constant bias δ (unknown input):

Component	Nominal constraint	Actual constraint
Level sensor	$h(t) = y(t)$	$h(t) = y(t) - \delta.$

Simple calculations show that the residual computed using the nominal model constraints would have a non-zero value:

Case	Residual value
Nominal	$\dot{y}(t) + k\sqrt{y(t)} - \alpha u(t) = 0$
Sensor bias	$\dot{y}\,(t) - \alpha \cdot u(t) + k\sqrt{y(t) - \delta} = k\left(\sqrt{y(t) - \delta} - \sqrt{y(t)}\right) \neq 0.$

Hence, although the system is faultless, the residual is non-zero due to the measurement bias δ. □

Example 5.39 Residual discrepancies caused by uncertain parameters
Consider now the two following cases for the single-tank system

Component	Nominal constraint	Actual constraint
Pump	$q_i(t) = \alpha \cdot u(t)$	$q_i(t) = \bar{\alpha} \cdot u(t)$
Output pipe	$q_0(t) = k\sqrt{y(t)}$	$q_0(t) = \bar{k}\sqrt{y(t)}.$

which refer to uncertainties in the pump and output pipe parameters. Then, the residual computed using the nominal model constraints would have the following values:

Case	Residual value
Behaviour without uncertainty	$\dot{y}(t) + k\sqrt{y(t)} - \alpha u(t) = 0$
Uncertainty of the pump model	$\dot{y}\,(t) + k\sqrt{y(t)} - \bar{\alpha} u(t) = (\alpha - \bar{\alpha})\, u(t) \neq 0$
Uncertainty of the output pipe model	$\dot{y}(t) - \alpha u(t) + \bar{k}\sqrt{y(t)} = (\bar{k} - k)\sqrt{y(t)} \neq 0.$

Again, a non-zero residual results not from a fault, but from uncertainties of a parameter. □

Robustness refers to the property that residuals would not fire any false alarm as the result of unknown inputs acting on the system or as the result of uncertainties in the values of the system parameters. One means of designing robust residuals is the exact decoupling approach, in which the designed residuals are insensitive to unknown input and unknown or uncertain parameters. Therefore, they are satisfied when no fault is present for any value of the unknown input or uncertain parameters. Note that the robustness problem is automatically solved in structural analysis, using the exact decoupling approach presented in Chap. 6, because it exhibits ARRs which are, by definition, only dependent on known variables. Unknown variables which affect the structurally monitorable subsystem are eliminated so that no residual can depend on them. When unknown variables cannot be eliminated, the part of the system they affect is not monitorable. When uncertain parameters are present, the solution to the exact decoupling problem is simply to design the fault diagnosis system considering them as unknown variables (this boils down to use the subset of residuals in which no uncertain parameter intervenes). The consequence is that the number of ARRs will in that case be smaller.

Structured residuals. As defined above, the structure of a residual is the set of the constraints which can be suspected when this residual is not ZERO. Let $\mathcal{R}$ be a set

of residuals, and let $\Phi(r) \in 2^{\mathcal{C}}$ be the structure of residual $r \in \mathcal{R}$. This means that r is expected to be non-zero when at least one of the constraints in $\Phi(r)$ is faulty. Similarly, when some constraint $\varphi \in \mathcal{C}$ is faulty, then all the residuals whose structure contains φ are expected to be non-zero. The pattern of ZERO and non-zero residuals associated with a given fault is called its *signature*.

Faults which have different signatures are isolable from each other, while faults which share the same signature are non-isolable. Let $\mathcal{R} = \mathcal{R}_0(t) \cup \mathcal{R}_1(t)$ be the decomposition of the set of residuals provided at some given time t by the decision procedure, where $\mathcal{R}_0(t)$ is the subset of the ZERO residuals and $\mathcal{R}_1(t)$ is the subset of non-zero ones. The subset of *suspected* constraints (the constraints which might be unsatisfied) at time t is given by

$$\mathcal{C}_{\text{susp}}(t) = \cap_{r \in \mathcal{R}_1(t)} \Phi(r).$$

Note that it is possible to define the subset of *exonerated* constraints (the constraints which are certainly satisfied) at time t by

$$\mathcal{C}_{\text{exo}}(t) = \cup_{r \in \mathcal{R}_0(t)} \Phi(r),$$

but one must be aware that this supposes all faults to be detectable. Exoneration is based on the assumption that if a constraint is not satisfied then it will necessarily show through the residuals whose structure it belongs to. The diagnosis at time t is

$$\mathcal{C}_{\text{diag}}(t) = \mathcal{C}_{\text{susp}}(t) \setminus \mathcal{C}_{\text{exo}}(t).$$

In order to obtain good isolability properties, it may be of interest to find residuals with given structure. Suppose that one wishes to have residuals which are insensitive to the structural faults of a subset of constraints $\mathcal{C}'$ and are sensitive to the structural faults of the subset of constraints $\mathcal{C} \setminus \mathcal{C}'$. A direct approach towards such residuals is to consider only the system $(\mathcal{C} \setminus \mathcal{C}', \mathcal{Z})$ in the design process. However, from the structural monitorability condition, it is seen that the residuals can be made sensitive only to the faults in the monitorable subsystem of $(\mathcal{C} \setminus \mathcal{C}', \mathcal{Z})$, which may be smaller than that of $(\mathcal{C}, \mathcal{Z})$, because the former contains less constraints.

Example 5.40 Two-tank system
The two-tank system introduced in Sect. 2.1 will first be considered with u as the known control input and q_{m} as the measured outflow. The following equations lead to the structure graph in Fig. 5.24.

$$
\begin{aligned}
c_1: &\quad q_{\mathrm{L}} = c_{\mathrm{L}}\sqrt{h_1} \\
c_2: &\quad q_{\mathrm{P}} = u \cdot f(h_1) \\
c_3: &\quad \dot{h}_1 = \frac{1}{A}\,(q_{\mathrm{P}} - q_{\mathrm{L}} - q_{12}) \\
d_4: &\quad \dot{h}_1 = \frac{\mathrm{d}}{\mathrm{d}t}\,h_1 \\
c_5: &\quad q_{12} = k_1\sqrt{h_1 - h_2} \\
c_6: &\quad \dot{h}_2 = \frac{1}{A}\,(q_{12} - q_2) \\
d_7: &\quad \dot{h}_2 = \frac{\mathrm{d}}{\mathrm{d}t}\,h_2 \\
c_8: &\quad q_2 = k_2\sqrt{h_2} \\
c_{\mathrm{m}}: &\quad q_{\mathrm{m}} = q_2.
\end{aligned}
$$

A, k_1, k_2 are known parameters. c_{L} is the unknown parameter describing the size of the fault. It can be assumed to be zero for the faultless case. In the structure graph the constraints c_1, c_2, c_3 and d_4 representing the Tank 1 are separated from constraints c_6, d_7, c_8 and m_1 describing the Tank 2.

The following matching is found using the ranking algorithm, where the last column shows the rank of the constraints obtained.

$$
\begin{aligned}
&q_{\mathrm{L}} = c_{\mathrm{L}}\sqrt{h_1} \\
&q_{\mathrm{P}} = u \cdot f(h_1) \\
&0 = -q_{\mathrm{L}} + q_p - q_{12} - A\dot{h}_1 \\
&\dot{h}_1 = \frac{\mathrm{d}}{\mathrm{d}t}\,h_1 \\
&h_1 = h_2 + \left(\frac{q_{12}}{k_1}\right)^2 \\
&q_{12} = A\dot{h}_2 + q_2 \\
&\dot{h}_2 = \frac{\mathrm{d}}{\mathrm{d}t}\,h_2 \\
&h_2 = \left(\frac{q_2}{k_2}\right)^2 \\
&q_2 = q_{\mathrm{m}}
\end{aligned}
$$

↗	q_{L}	q_{P}	$\dot{h}_1$	h_1	q_{12}	$\dot{h}_2$	h_2	q_2	R
c_1	①			1					5
c_2		①		1					5
c_3	1	1	1		1				6
d_4			①	1					5
c_5				①	1		1		4
c_6					①	1		1	3
d_7						①	1		2
c_8							①	1	1
m_1								①	0

The equations shown on the left are already solved for the matched variable. The corresponding oriented graph is shown in Fig. 5.30. Simplifying these equations results in the following redundancy relation,

$$arr(t) = u(t) \cdot f\,(h_1(t)) - A\dot{h}_2(t) + q_{\mathrm{m}}(t) - A\dot{h}_{1(t)} - c_{\mathrm{L}}\sqrt{h_1(t)} \tag{5.33}$$

with

$$h_1(t) = h_2(t) + \left(\frac{A\dot{h}_2(t)}{k_1} + \frac{q_{\mathrm{m}}(t)}{k_1}\right)^2 \tag{5.34}$$

$$h_2(t) = \left(\frac{q_{\mathrm{m}}(t)}{k_2}\right)^2. \tag{5.35}$$

Equations (5.33)–(5.35) can be used to monitor the two-tank system. By using Eq. (5.35), $h_2(t)$ and, hence, $\dot{h}_2$(t) can be determined for given measurement $q_m(t)$. Then Eq. (5.34) yields $h_1(t)$ and $\dot{h}_1(t)$. Finally, Eq. (5.33) is checked for known $u(t)$, $q_m(t)$ and for $h_1(t)$, $\dot{h}_1(t)$ and $\dot{h}_2(t)$ just obtained.

After redrawing the structure graph, Fig. 5.31 is obtained. This graph shows in which order the constraints can be used to determine all internal variables for given measurement q_m. Finally, constraint c_3 is used to test the consistency of the variables with the model. The resulting value is denoted by $r(t)$. This residual should vanish to indicate that the measured values $q_m(t)$ and $u(t)$ at time t are consistent with the set of constraints and, hence, we must assume that no fault is present. For this example, the residual has the physical meaning of the loss of liquid through a leakage.

A simulation result is depicted in Fig. 5.32 which shows from top to bottom the signals $u(t)$, $x_1(t)$ and $x_2(t)$, the measurement $q_m(t)$ and the right-hand side of Eq. (5.33). Note that the states are reconstructed very nicely. The residual shows the occurrence of the fault very precisely and without any delay. The little spike at time 155 s is due to the reversal of the flow direction in the connection pipe, which represents a singular point in the linearised system.

The signal $arr(t)$ is non-causal due to the two differentiations. To construct a residual, low-pass filtering need be applied to get a causal residual generator. In Laplace transform notation,

$$r(s) = \frac{1}{(1+s\tau)^2} arr(s) \tag{5.36}$$

here illustrated by a second-order low-pass filter with two real eigenvalues. It is essential that it is never the signals $\dot{h}_1$ and $\dot{h}_2$ from h_1 or h_2, respectively, which are low-pass filtered, but the

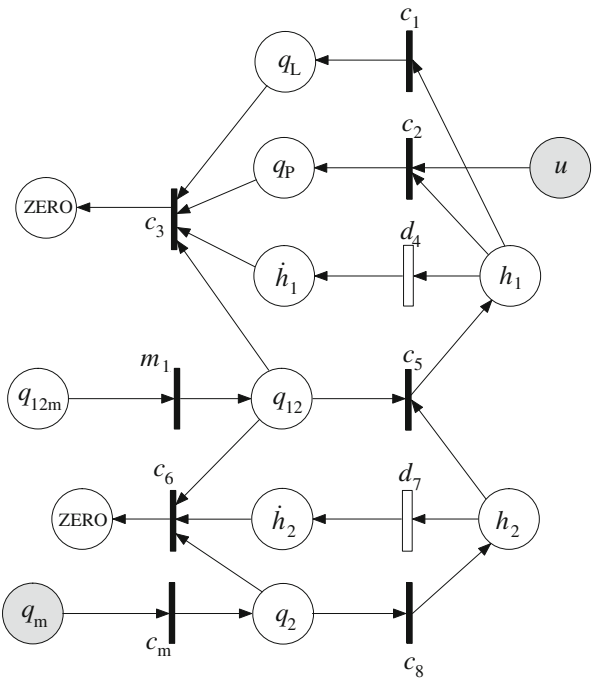


Fig. 5.30 Oriented graph of the two-tank system

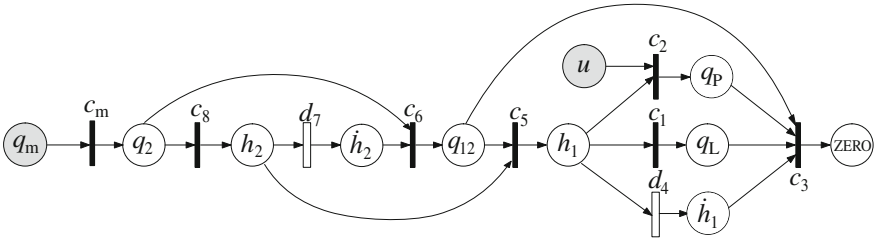


Fig. 5.31 Graph showing the order in which the unknown variables can be determined for given q_m

entire ARR expression. Otherwise, due to the phase lag introduced by filtering, the residual given by Eq. (5.33) might no longer be zero for the faultless case.

Structured residuals. Assume now that the flow q_{12} between the two tanks can be measured in addition to the input u and the outflow q_m, which leads to the additional measurement constraint

$$m_2: \quad q_{12} = q_{12,m} k_m.$$

The system is over-constrained with two remaining constraints that lead to two residuals:

$$
\begin{aligned}
q_L &= c_L \sqrt{h_1} \\
q_P &= u \cdot f(h_1) \\
0 &= -q_L + q_p - q_{12} - A\dot{h}_1 \\
\dot{h}_1 &= \frac{d}{dt} h_1 \\
h_1 &= h_2 + \left(\frac{q_{12}}{k_1}\right)^2 \\
0 &= q_{12} - A\dot{h}_2 - q_2 \\
\dot{h}_2 &= \frac{d}{dt} h_2 \\
h_2 &= \left(\frac{q_2}{k_2}\right)^2 \\
q_2 &= q_m \\
q_{12} &= q_{12,m}
\end{aligned}
$$

↗	q_L	q_P	$\dot{h}_1$	h_1	q_{12}	$\dot{h}_2$	h_2	q_2	R
c_1	①			1					3
c_2		①		1					3
c_3	1	1	1		1				4
d_4			①	1					3
c_5				①	1		1		2
c_6					1	1		1	4
d_7						①	1		2
c_8							①	1	1
m_1								①	0
m_2					①				0

This matching results in the oriented graph shown in Fig. 5.33. Following the orientation of the edges, it is easy to see that the first parity relation depends only on the variables

$$\{u,\ q_L,\ q_P,\ \dot{h}_1,\ h_1,\ q_{12},\ q_{12,m},\ h_2,\ q_2,\ q_m\},$$

while the second depends on

$$\{q_{12},\ q_2,\ h_2,\ \dot{h}_2,\ q_{12,m},\ q_m\}.$$

These two conditions can be used to selectively monitor Tank 1 and Tank 2. Only a fault in the connection flow q_{12} or its measurement would affect both constraints.

From the graph two ARRs $arr_1(t)$ and $arr_2(t)$ are obtained:

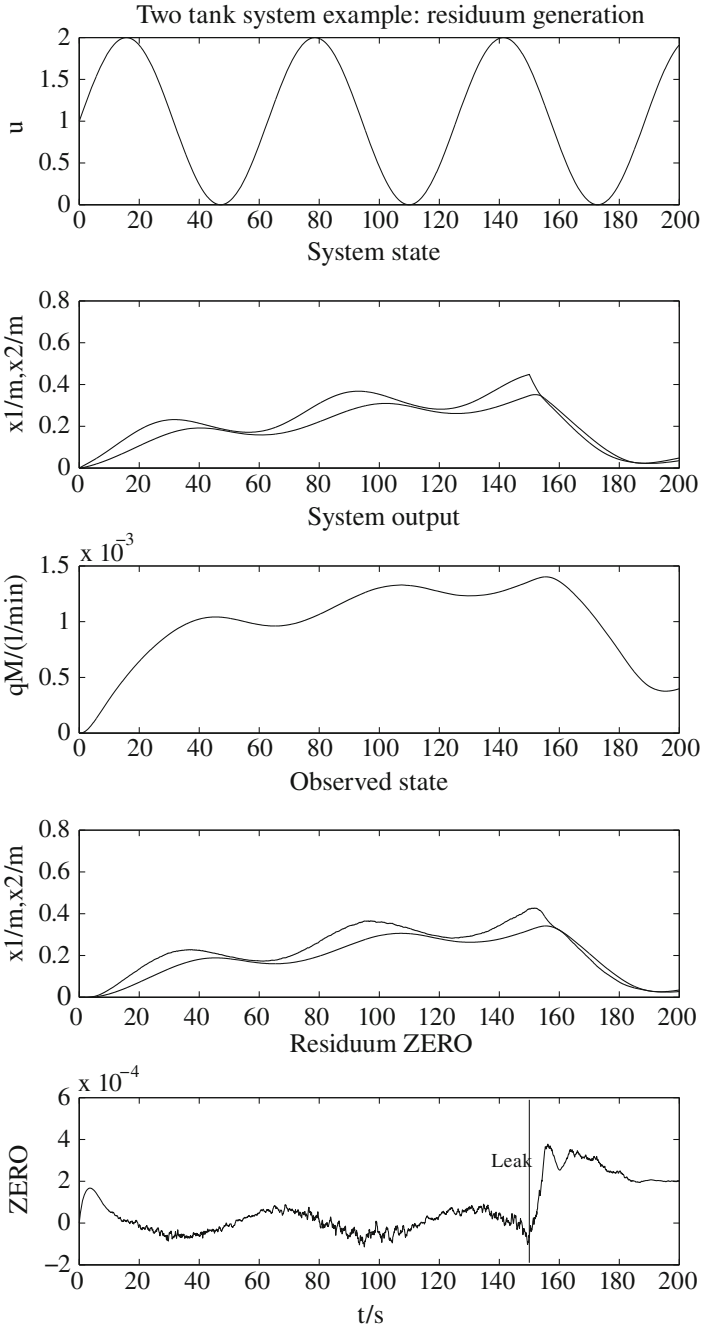


Fig. 5.32 Simulation results of the two-tank system. From *top* to *bottom* input u; tank levels h_1, h_2; measured q_m; reconstructed levels h_1, h_2; right-hand side of Eq. (5.33)

$$arr_1(t) = u(t) \cdot f(h_1(t)) - q_{12,m}(t) - A\dot{h}_1(t) - c_L\sqrt{h_1(t)}$$
$$arr_2(t) = q_{12,m}(t) - A\dot{h}_2(t) - \frac{q_m(t)}{k_m}$$

with

$$h_1(t) = h_2(t) + \left(\frac{q_{12,m}(t)}{k_1}\right)^2$$
$$h_2(t) = \left(\frac{q_m(t)}{k_m k_2}\right)^2.$$

These residuals are structured in the sense that they become non-zero if Tank 1 or Tank 2 is affected by some fault. Hence, the additional measurement makes it possible not only to detect a fault in the overall system but to identify the affected component.

Structural isolability using the MSO approach. The ranking algorithm obtained one complete matching of the unknown variables. The matching can be represented in condensed form as follows, where 0 in a column denotes that the constraint is unmatched and used as an ARR.

	c_1	c_2	c_3	d_4	c_5	c_6	d_7	c_8	m_1	m_2
1	q_L	q_P	0	$\dot{h}_1$	h_1	$\dot{h}_2$	0	h_2	q_2	q_{12}

Giving rise to the two ARRs listed above, structural detectability and isolability is shown in

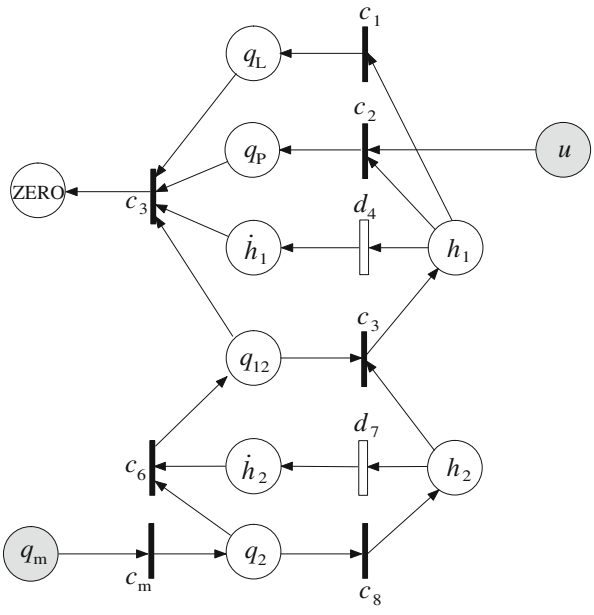


Fig. 5.33 Oriented graph, in which the arrows indicate the order of matching

	c_1	c_2	c_3	d_4	c_5	c_6	d_7	c_8	m_1	m_2
1	d	d	d	n	d	i	n	d	d	d

where d denotes structurally detectable, i structurally isolable and n that the particular constraint cannot fail. Differential constraints cannot fail as these are just definitions that relate a variable $\dot{h}$ with h through a differential operator. The result is that only one constraint is structurally isolable. Using the MSO set approach, the following MSO sets are received:

	c_1	c_2	c_3	d_4	c_5	c_6	d_7	c_8	m_1	m_2
$\mathcal{M}_1$						0	$\dot{h}_2$	h_2	q_2	q_{12}
$\mathcal{M}_2$	q_L	q_P	0	$\dot{h}_1$	h_1			h_2	q_2	q_{12}
$\mathcal{M}_3$	q_L	q_P	0	$\dot{h}_1$	h_1	q_2	$\dot{h}_2$	h_2		q_{12}
$\mathcal{M}_4$	q_L	q_P	0	$\dot{h}_1$	h_1	q_{12}	$\dot{h}_2$	h_2	q_2	

The fields either contain the matched unknown variables, zeros to indicate an unmatched constraints or nothing if constraints are not used in the MSO set. Four MSO sets are determined.

Using all four MSO sets and ARRs, the resulting detectability and isolability properties are shown in the following table:

	c_1	c_2	c_3	d_4	c_5	c_6	d_7	c_8	m_1	m_2
$\mathcal{M}_1$				n		d	n	d	d	d
$\mathcal{M}_2$	d	d	d	n	d		n	d	d	d
$\mathcal{M}_3$	d	d	d	n	d	d	n	d		d
$\mathcal{M}_4$	d	d	d	n	d	d	n	d	d	
all	d	d	d	n	d	i	n	i	i	i

Using all four ARR's from MSO sets 1–4 results in enhanced isolability, now including both measurements. □

It is a general finding that use of several ARR's in parallel can enhance isolability. Violation of some constraints remain only detectable, and isolating these as possible sources of a fault requires another approach, referred to as *active fault isolation*.

5.6.6 Active Fault Isolation

Active structural isolation is an extension of the passive technique considered so far, where residuals were formed from ARRs by backtracking to known variables, input $u(t)$ and measurements $y(t)$, and evaluating the residual $r(t)$ in real time. This approach was seen to lead to cases where some violations of constraints could only be detected but not isolated. We also encountered cases where violation of one of the constraints within a group could be pinpointed as the possible source of violation but

isolation could not be achieved, i.e. we could not distinguish which of the constraints within the group had been violated (groupwise isolability).

Active fault isolation employs a perturbation in one or more of the input signals, once it has been detected that some fault is present, to attempt to determine which individual constraints have been violated.

Active isolation is needed if faults are groupwise isolable, i.e. within the group individual faults are detectable but not structurally isolable to an individual constraint. This does not necessarily imply that isolation cannot be achieved in other ways. Exciting the system with an input signal perturbation may make it possible to discriminate different responses of the same residual set, or from input to output in the system, when different constraints within the group are faulty. The following observation is obvious:

Lemma 5.3 *Active structural isolation is possible if and only if both a structural condition and a quantitative condition are true.*

- *Structural condition: The known variables in the set of residuals associated with a group of non-structurally isolable constraints include at least one control input.*
- *Quantitative condition 1: The transfer from control inputs to residuals is affected differently by faults on different constraints.*
- *Quantitative condition 2: The transfer from control inputs to outputs is affected differently by faults on different constraints.*

Active structural isolation is possible if the structural condition and one or both of the quantitative conditions are met. In order to express the quantitative condition in rigorous terms, we need the following definitions, which are based on reachability and monitorability.

Definition 5.10 (*Presence in path from input to residual or to output*) Let z_j denote residual r_j or output y_j. Let $p^{(i,j)} = \{c_f, c_g, \ldots, c_h\}$ be a path through the structure graph from input u_i to z_j and $\prod^{(i,j)}$ the union of valid paths from u_i to z_j. Let

$$C_{\text{reach}}^{(i,j)} = \left\{ c_g \mid c_g \in \prod^{(i,j)} \right\}.$$

A constraint c_h is present in a path from u_i to z_j, and the path includes the constraint $c_h \in C_{\text{reach}}^{(i,j)}$ if c_h is reachable from u_i and is monitorable from z_j.

Lemma 5.4 *Active structural isolability is from input to residual or to output. Two constraints c_g and c_h are actively isolable from residual, respectively, output signatures if*

$$\exists i, j, k, l : c_g \in C_{\text{reach}}^{(i,j)}, c_h \in C_{\text{reach}}^{(k,l)} \quad \textit{and} \quad \left\{c_g, c_h\right\} \notin C_{\text{reach}}^{(i,j)} \cap C_{\text{reach}}^{(k,l)}.$$

This Lemma advises an easily verifiable way to determine whether one or more constraints, which are only groupwise isolable with the passive approach outlined earlier, could be subjected to active isolation.

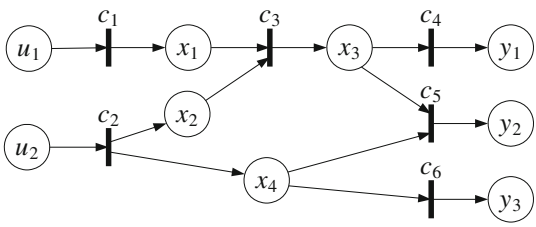


Fig. 5.34 Structure graph for the active diagnosis example

Active isolation is employed once a fault has been detected but the exact location could not be determined because the event only possess groupwise structural isolability with the set of residuals used.

Algorithmic aspects. A path through a graph can be determined from the adjacency matrix (cf. Chap. 4)

$$\boldsymbol{A} : [C, K_i, K_m] \rightarrow [C, K_i, K_m]$$

to show which nodes in a graph are connected. As the graph is bipartite, the adjacency matrix is easily obtained from the incidence matrix $\boldsymbol{S}$ as

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{O} & \boldsymbol{S} \\ \boldsymbol{S}^{\mathrm{T}} & \boldsymbol{O} \end{pmatrix}.$$

The adjacency matrix shows the result of a walk of length 1. A walk of length n will be described by $\boldsymbol{A}^n$. Reachability of element i from element j in the graph is determined by investigating the element (i, j) in the sequence of matrices

$$\boldsymbol{A}^1, \boldsymbol{A}^2, \boldsymbol{A}^3, \ldots, \boldsymbol{A}^{2cn}$$

where cn is the number of elements in $\{C, K_i, K_m\}$. With the ith column of $\boldsymbol{A}$ being an input, and the jth row an output, or the residual associated with the zero variable belonging to an unmatched constraint, a path of length m exists from i to j if and only if $\boldsymbol{A}^m_{ij} \neq 0$. The nodes passed on the walk are determined by tracing the non-zero elements of $\boldsymbol{A}^m, \boldsymbol{A}^{m-1}, \ldots, \boldsymbol{A}^1$. While this algebraic method is intuitive and is related to the structure graph $\boldsymbol{S}$, it is computationally inefficient for large systems and algorithmic methods exist that can find all paths from a given input to any variable in a graph.

Example 5.41 Active diagnosis
Let a system be given by the structure graph shown in Fig. 5.34. The set of inputs is $\mathcal{K}_i = \{u_1, u_2\}$, the set of outputs $\mathcal{K}_m = \{y_1, y_2, y_3\}$, unknown variables are $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$. The associated incidence matrix is shown in the following table:

$\nearrow$	u_1	u_2	y_1	y_2	y_3	x_1	x_2	x_3	x_4
c_1	1	0	0	0	0	1	0	0	0
c_2	0	1	0	0	0	0	1	0	1
c_3	0	0	0	0	0	1	1	1	0
c_4	0	0	1	0	0	0	0	1	0
c_5	0	0	0	1	0	0	0	1	0
c_6	0	0	0	0	1	0	0	0	1

A complete matching on the unknown variables can be achieved using the ranking algorithm, leaving c_6 and c_3 as unmatched constraints. The path found by the matching is the following:

$$\begin{aligned} c_1(u_1) &\rightarrow x_1; c_4(y_1) \rightarrow x_3; \\ c_5(x_3) &\rightarrow x_4; c_2(u_2, x_4) \rightarrow x_2 \\ &\Rightarrow c_3(x_1, x_2, x_3) = 0 \\ &\Leftrightarrow c_3(c_1(u_1), c_2(u_2, x_4), c_4(y_1)) = 0 \\ &\Leftrightarrow c_3(c_1(u_1), c_2(u_2, c_5(c_4(y_1))), c_4(y_1)) = 0 \end{aligned}$$

and

$$\begin{aligned} c_6(y_3, x_4) = 0 \Leftrightarrow c_6(y_3, c_5(x_3)) &= 0 \\ \Leftrightarrow c_6(y_3, c_5(c_4(y_1))) &= 0. \end{aligned}$$

The analytical redundancy relations associated with c_3 and c_6 constitute two parity relations for the system considered in the example and two residual generators are

$$\begin{aligned} r_1 &= c_3(c_1(u_1), c_2(u_2, c_5(c_4(y_1))), c_4(y_1)) \\ r_2 &= c_6(y_3, c_5(c_4(y_1))). \end{aligned}$$

The dependency matrix between residuals and constraints shown in

$\nearrow$	c_1	c_2	c_3	c_4	c_5	c_6
r_1	1	1	1	1	1	0
r_2	0	0	0	1	1	1

imply the detectability and the isolability as achievable from the two residuals. Linearly independent columns show that violation of constraint c_6 can be isolated. The sets

$$\{c_4, c_5\} \quad \text{and} \quad \{c_1, c_2, c_3\}$$

are blockwise isolable but violation of any of the individual constraints will only be detectable.

In a fault-tolerant control setting, inputs u_1 and u_2 can be individually perturbed by the control system. The set of paths through constraints from u_1 to the outputs are represented in the reachability table

$u_1 \downarrow$	c_1	c_2	c_3	c_4	c_5	c_6
y_1	1	0	1	1	0	0
y_2	1	0	1	0	1	0
y_3	0	0	0	0	0	0

The reachability from u_2 is shown in

$u_2 \downarrow$	c_1	c_2	c_3	c_4	c_5	c_6
y_1	0	1	1	1	0	0
y_2	0	1	0	0	1	0
y_3	0	1	0	0	0	1

Following Lemma 5.4, it is easily seen that $\{c_1, c_2, c_3\}$ are structurally isolable when active isolation is employed, while c_4 remains detectable. □

5.7 Structural Controllability and Structural Observability

Structural controllability and structural observability are two notions that have been introduced long ago with the aim to show that dynamical systems have the properties of controllability and of observability mainly for structural reasons. The well-known rank conditions on the controllability matrix or the observability matrix can only be satisfied if the non-zero entries of these matrices satisfy structural conditions.

This short section should show that as far as controllability and observability are concerned, structural results obtained by the analysis methods explained in this chapter by means of a bipartite graph are rather similar to those results that have been derived in control theory by a structural representation of linear dynamical systems by directed graphs.

5.7.1 Observability and Computability

Known and unknown variables. As before, the set of system variables $\mathcal{Z}$ is decomposed into the sets $\mathcal{K}$ of known variables and the set $\mathcal{X}$ of unknown variables. Known variables are available in real time, while unknown variables are not directly measured. *Observability* is the system property that allows to determine all unknown variables from all known variables. Analysing the system observability coincides with identifying ways in which those unknown variables can be calculated.

Consider the general system described by the Eqs. (5.1)–(5.4)

$$\dot{\boldsymbol{x}}_{\mathrm{d}}(t) = \boldsymbol{g}\left(\boldsymbol{x}_{\mathrm{d}}(t), \boldsymbol{x}_{\mathrm{a}}(t), \boldsymbol{u}(t)\right) \tag{5.37}$$

$$\boldsymbol{0} = \boldsymbol{m}\left(\boldsymbol{x}_{\mathrm{d}}(t), \boldsymbol{x}_{\mathrm{a}}(t), \boldsymbol{u}(t)\right) \tag{5.38}$$

$$\boldsymbol{y}(t) = \boldsymbol{h}\left(\boldsymbol{x}_{\mathrm{d}}(t), \boldsymbol{x}_{\mathrm{a}}(t), \boldsymbol{u}(t)\right) \tag{5.39}$$

$$\dot{\boldsymbol{x}}_{\mathrm{d}}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}_{\mathrm{d}}(t) \tag{5.40}$$

with the set of known variables $\mathcal{K} = \{\boldsymbol{u}, \boldsymbol{y}\}$, the set of unknown variables $\mathcal{X} = \{\boldsymbol{x}_{\mathrm{a}}, \boldsymbol{x}_{\mathrm{d}}, \dot{\boldsymbol{x}}_{\mathrm{d}}\}$ and the set of constraints $\mathcal{C} = \{\boldsymbol{g}, \boldsymbol{m}, \boldsymbol{h}, \frac{\mathrm{d}}{\mathrm{d}t}\}$. According to the decomposition of $\mathcal{Z}$ into $\mathcal{K} \cup \mathcal{X}$, $\mathcal{C}$ is decomposed into $\mathcal{C}_{\mathcal{K}} \cup \mathcal{C}_{\mathcal{X}}$:

$$\begin{aligned} \mathcal{C}_{\mathcal{K}} &= \{c \in \mathcal{C};\ Q(c) \cap \mathcal{X} = \emptyset\} \\ \mathcal{C}_{\mathcal{X}} &= \{c \in \mathcal{C};\ Q(c) \cap \mathcal{X} \neq \emptyset\}\,. \end{aligned}$$

$\mathcal{C}_{\mathcal{K}}$ is the largest subset of constraints such that $Q(\mathcal{C}_{\mathcal{K}}) \subseteq \mathcal{K}$. For the aim to analyse the possibility of computing the unknowns in $\mathcal{X}$, only the subgraph $(\mathcal{C}_{\mathcal{X}}, \mathcal{X}, \mathcal{E}_{\mathcal{X}})$ needs to be decomposed.

5.7.2 Structural Observability Conditions

For the canonical decomposition

$$\begin{aligned} \mathcal{S}^{+} &= (\mathcal{C}^{+}_{\mathcal{X}}, \mathcal{X}^{+}) \\ \mathcal{S}^{0} &= (\mathcal{C}^{0}_{\mathcal{X}}, \mathcal{X}^{+} \cup \mathcal{X}^{0}) \\ \mathcal{S}^{-} &= (\mathcal{C}^{-}_{\mathcal{X}}, \mathcal{X}^{+} \cup \mathcal{X}^{0} \cup \mathcal{X}^{-}) \end{aligned}$$

of the subgraph $(\mathcal{C}_{\mathcal{X}}, \mathcal{X}, \mathcal{E}_{\mathcal{X}})$ associated with the system (5.37)–(5.40), structural observability can be characterised as follows:

Theorem 5.5 (Structural observability) *A necessary and sufficient condition for system (5.37)–(5.40) to be structurally observable is that, under derivative causality,*

1. *all the unknown variables are reachable from the known ones,*
2. *the over-constrained and the just-constrained subsystems are causal,*
3. *no under-constrained subsystem exists.*

Condition 1 says that there does not exist any subsystem whose behaviour is not reflected in the behaviour of the known variables, while Conditions 2 and 3 impul that all the variables can be matched using causal matchings and thus are uniquely defined once the known variables are given.

Example 5.42 Non-reachability
Consider the following incidence matrix, in which the variable x_3 is not reachable from the output.

↗	x_1	x_2	x_3	$\dot{x}_1$	$\dot{x}_2$	$\dot{x}_3$	u	y
c_1	1	1		1			1	
d_1	**x**			1				
c_2	1	1			1			
d_2		**x**			1			
c_3			1			1		
d_3			**x**			1		
m	1							1

The constraint set associated with such a structure graph has the form

$$\begin{array}{lrcl} & \dot{x}_1(t) & = & g_1\,(x_1(t),\, x_2(t), u(t)) \\ \text{Subsystem 1:} & \dot{x}_2(t) & = & g_2\,(x_1(t),\, x_2(t)) \\ & y(t) & = & h\,(x_1(t)) \\ \text{Subsystem 2:} & \dot{x}_3(t) & = & g_3\,(x_3(t)). \end{array} \tag{5.41}$$

It is seen that Subsystem 2 can by no means be observable. □

Example 5.43 Observability of a nonlinear system
Consider the following nonlinear dynamical system with two state variables, two input signals, one parameter θ and one sensor:

$$\begin{aligned} c_1: &\quad \dot{x}_1(t) = (\theta - 1)x_2(t)\,u_1(t) \\ c_2: &\quad \dot{x}_2(t) = u_2(t) \\ m: &\quad y(t) = x_1(t). \end{aligned}$$

This system is over-constrained and satisfies the three conditions of the above theorem. The following matching allows to compute the state.

↗	$\dot{x}_1$	$\dot{x}_2$	x_1	x_2	u_1	u_2	y
c_1	1			①	1		
c_2		1				1	
d_1	①		**x**				
d_2		①		**x**			
m			①				1

The variable x_2 can be reached from the known variables if and only if the matching $(c_1,\, x_2)$ can be used, which means that the two conditions

$$u_1 \neq 0 \quad \text{and} \quad \theta \neq 1$$

simultaneously have to hold. If not, the system is not observable, because there is no matching by means of which x_2 could be computed under derivative causality.

This example illustrates the fact that structural properties provide results which are valid for almost every value of the system parameters *and variables*. □

5.7.3 Observability and Structural Observability of Linear Systems

Let us consider the linear time-invariant system

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) \tag{5.42}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t), \tag{5.43}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are of dimensions n and p. In linear system theory it has been proved that the state is observable if and only if the following condition holds

$$\text{rank} \begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \vdots \\ \boldsymbol{CA}^{n-1} \end{pmatrix} = n, \tag{5.44}$$

for which a necessary condition is

$$\text{rank} \begin{pmatrix} \boldsymbol{A} \\ \boldsymbol{C} \end{pmatrix} = n. \tag{5.45}$$

Equation (5.45) means, in structural terms, that the unknown variable $\boldsymbol{x}$ belongs to a causal just-constrained or over-constrained subsystem, when derivative causality is imposed. The structure graph is

↗	$\dot{\boldsymbol{x}}$	$\boldsymbol{y}$	$\boldsymbol{x}$
$\boldsymbol{d}$	$\boldsymbol{I}$		x
$\boldsymbol{m}$		$\boldsymbol{I}$	$\boldsymbol{S}_C$
$\boldsymbol{c}$	$\boldsymbol{I}$		$\boldsymbol{S}_A$

where $\boldsymbol{d}$ are the derivative constraints, which express that dots mean derivatives, $\boldsymbol{m}$ are the constraints (5.43) from the measurement, and $\boldsymbol{c}$ are the system constraints (5.42). $\boldsymbol{S}_C$ and $\boldsymbol{S}_A$ are the structures associated with matrices $\boldsymbol{C}$ and $\boldsymbol{A}$. Since no variable in $\boldsymbol{x}$ can be matched from any constraint in $\boldsymbol{d}$, the system $(\{\boldsymbol{c}, \boldsymbol{m}\}, \{\dot{\boldsymbol{x}}, \boldsymbol{x}, \boldsymbol{y}\})$ must be over-constrained with respect to $\boldsymbol{x}$. It can be noted that this requirement does not constitute a sufficient condition, because the system parameters might have values such that (5.44)—or (5.45)—is not satisfied.

Example 5.44 Observability of linear systems
Consider the unobservable linear time-invariant system

$$\begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{pmatrix} = \begin{pmatrix} 0 & 0 & c \\ 0 & 0 & d \\ a & b & e \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{pmatrix} \tag{5.46}$$

$$y(t) = \begin{pmatrix} 0 & 0 & f \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{pmatrix}, \tag{5.47}$$

where the parameters a, b, c, d, e, f can take any real value. Its structure graph has the incidence matrix

$\nearrow$	$\dot{x}_1$	$\dot{x}_2$	$\dot{x}_3$	x_1	x_2	x_3	y
c_1	1					1	
c_2		1				1	
c_3			1	1	1	1	
d_1	1			**x**			
d_2		1			**x**		
d_3			1			**x**	
m						1	1

where the constraints c_1, c_2, c_3 represent the system (5.46), the constraints d_1, d_2, d_3 express the derivative link between the x_1, x_2, x_3 and the $\dot{x}_1$, $\dot{x}_2$, $\dot{x}_3$ and m is the measurement Eq. (5.47). This system can be decomposed into a just-constrained part $\mathcal{C}^0_{\mathcal{X}} = \{c_1, c_2, d_3, m\}$, $\mathcal{X}^0 = \{\dot{x}_1, \dot{x}_2, \dot{x}_3, x_3\}$ from which $\dot{x}_1$, $\dot{x}_2$, $\dot{x}_3$ and x_3 can be computed as functions of y for almost all values of the parameters, and an under-constrained part $\mathcal{C}^-_{\mathcal{X}} = \{c_3, d_1, d_2\}$, $\mathcal{X}^- = \{x_1, x_2\}$ in which x_1 and x_2 should both be computed from the single constraint c_3. It can be checked that adding $\dot{y}$ and the associated constraints, the subsystem $(\{c_3, d_1, d_2\}, \{x_1, x_2\})$ remains under-constrained and that this will always be the case when higher derivatives $y^{(i)}$ will be considered. Consequently, the information available from the sensor is enough to place the vector $(x_1, x_2)^{\mathrm{T}}$ in a subspace of dimension one (since they are linked by one constraint which is known to be linear), but is not enough to compute this vector completely. The observability matrix

$$\begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \boldsymbol{CA}^2 \end{pmatrix} = \begin{pmatrix} 0 & 0 & f \\ af & bf & ef \\ aef & bef & (ac + bd + e^2)f \end{pmatrix}$$

is not full rank, whatever the coefficients a, b, c, d, e, f are, and it can be checked that no more than the linear form $ax_1 + bx_2$ can be determined from the observation $(y, \dot{y}, \ldots, y^{(s)})$ for any $s \geq 1$.

Consider now the case that the second state variable is measured:

$$\begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{pmatrix} = \begin{pmatrix} 0 & 0 & c \\ 0 & 0 & d \\ a & b & e \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{pmatrix} \tag{5.48}$$

$$y(t) = (0 \quad f \quad 0) \begin{pmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{pmatrix}. \tag{5.49}$$

Then the system is observable. The structure graph has the incidence matrix

$\nearrow$	$\dot{x}_1$	$\dot{x}_2$	$\dot{x}_3$	x_1	x_2	x_3	y
c_1	1					1	
c_2		1				1	
c_3			1	1	1	1	
d_1	1			**x**			
d_2		1			**x**		
d_3			1			**x**	
m					1		1

and the following causal matching shows that all the components of the state can be computed from y and its derivatives.

$\nearrow$	$\dot{x}_1$	$\dot{x}_2$	$\dot{x}_3$	x_1	x_2	x_3	y
c_1	1					1	
c_2		1				①	
c_3			1	①	1	1	
d_1	①			**x**			
d_2		①			**x**		
d_3			①			**x**	
m					①		1

□

5.7.4 Graph-Based Interpretation and Formal Computation

Since an oriented graph can be associated with each matching, the observability property can be analysed from a graph-theoretical point of view. Let x be an observable variable. Then x can be matched with a constraint the input of which is either known or a set of observable variables. By repeating this argument, it follows that for x to be observable, it is necessary that there exists at least one subgraph (a set of alternated chains) which links this variable with the known variables $\boldsymbol{u}$ and $\boldsymbol{y}$ and where no unobservable variable acts as an input in any constraint of this subgraph.

This subgraph with the observable target variable x may contain algebraic loops, but it does not contain any differential loop.

The constraints along the alternated chains show the computations which are to be performed in order to compute x. If these constraints are combined, a formal expression of x in terms of known variables can be obtained. A simple algebraic constraint in the chain means that the matched variable is computed as a function of the non-matched ones. An algebraic loop shows that a set of constraints has to be solved simultaneously. A derivative constraint means that the non-matched variable has to be derivated in order to obtain the matched variable (remember that only derivative causality is allowed). The number of derivative constraints which are included between a given input and the target variable shows the maximum order of derivations needed on this input for computing this target.

Note that this interpretation expresses that x belongs to a just- or an over-constrained causal subsystem. If x were to belong to an under-constrained subsystem, the corresponding subgraph would have less constraints than variables, i.e. some unknown variables would be input signals to constraints while being output of no other constraint.

For example, Fig. 5.35 shows the two graphs associated with the linear systems (5.46), (5.47) and (5.48), (5.49) which are non-observable or observable, respectively. It can be seen that in the first case, either x_2 or x_1 stands as an unknown input of constraint c_3 while in the second case, both can be matched thus providing all the states with known predecessors at some level.

When different estimation subgraphs with the same target variable exist, they provide different computation schemes for the same variable. This feature is of interest when monitorability and reconfigurability are considered as discussed in the next section.

5.7.5 Structural Controllability

Controllability is a property which describes the links between the unknown variables and the input variables, independently of the fact that some unknown variables might be measured or not. Thus, it can be analysed from the structure graph in which the measurement constraints have been removed. Roughly speaking, controllability is concerned with the possibility of finding controls so as to achieve objectives, which are defined in terms of the values one wishes the system variables to be given.

The reachable set of a system is the set of states in which the system can be brought by an appropriate control input. Global controllability is a strong property, which states that the reachable set is the whole state space. Local controllability is a weaker property, which requires that any point in the open ball around a reachable point is also reachable. For linear systems, local and global properties coincide.

Let us first consider static systems $(\mathcal{C},\ \mathcal{Z})$ like

$$\mathbf{0} = \boldsymbol{h}\,(\boldsymbol{x}_{\mathrm{a}},\ \boldsymbol{u}), \tag{5.50}$$

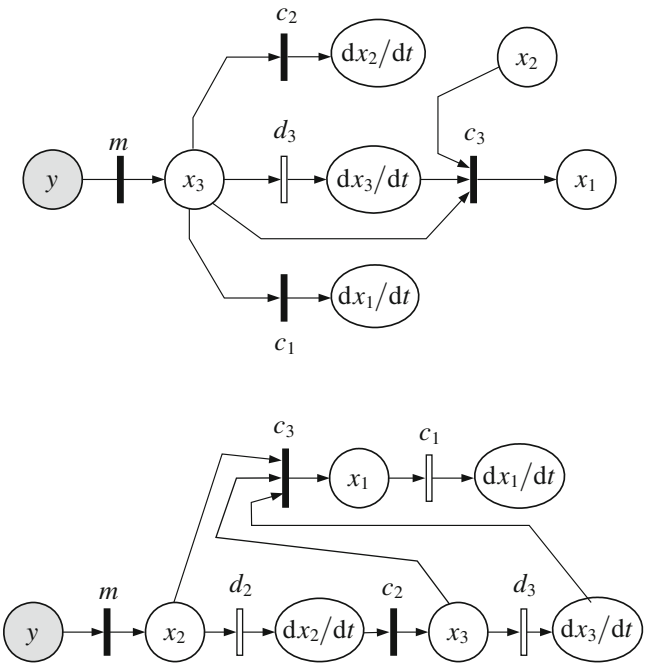


Fig. 5.35 Graph-based interpretation of the observability property

where $\mathcal{C} = \{\boldsymbol{h}\}$, $\mathcal{Z} = \{\boldsymbol{x}_\mathrm{a}, \boldsymbol{u}\}$. For such systems, global controllability means that Eq. (5.50) can be solved for the unknown variables $\boldsymbol{u}$ for any value of the known (wished) variables $\boldsymbol{x}_\mathrm{a}$, thus justifying the decomposition of $\mathcal{Z}$ into $\mathcal{Z} = \mathcal{K} \cup \mathcal{X}$, with $\mathcal{K} = \{\boldsymbol{x}_\mathrm{a}\}$, $\mathcal{X} = \{\boldsymbol{u}\}$.

Theorem 5.6 (Controllability of static systems) *Necessary and sufficient conditions for system (5.50) to be structurally controllable are the following:*

(i) The vertices of $\mathcal{K}$ are reachable in the structure graph from the input,
(ii) The canonical decomposition of $(\mathcal{C}_\mathcal{X}, \mathcal{X}, \mathcal{E}_\mathcal{X})$ contains no over-constrained subsystem.

If $\mathcal{K}$ were not reachable from the input, there would be a decomposition of $\boldsymbol{x}_\mathrm{a}$ into $\boldsymbol{x}'_\mathrm{a}$ (the reachable part), and $\boldsymbol{x}''_\mathrm{a}$ (the unreachable part), such that the model can be written as

$$\begin{aligned} \mathbf{0} &= \boldsymbol{h}'(\boldsymbol{x}'_\mathrm{a}, \boldsymbol{u}) \\ \mathbf{0} &= \boldsymbol{h}''(\boldsymbol{x}''_\mathrm{a}). \end{aligned}$$

There is no solution to this model for *any* $\boldsymbol{x}_\mathrm{a}$, namely when $\boldsymbol{x}_\mathrm{a}$ is such that the part $\boldsymbol{x}''_\mathrm{a}$ does not satisfy the second equation. On the other hand, if the canonical

decomposition contains an over-constrained subsystem, the known variables satisfy some compatibility condition, which results in the existence of some manifold

$$\alpha(\boldsymbol{x}_\mathrm{a}) = 0$$

and in the impossibility to find any control $\boldsymbol{u}$ when the wished system states lie out of this manifold.

The case of dynamical systems is more complex, and except for linear systems, only the reachability condition of the above result can be extended. Consider the general system

$$\dot{\boldsymbol{x}}_\mathrm{d}(t) = \boldsymbol{g}\left(\boldsymbol{x}_\mathrm{d}(t),\, \boldsymbol{x}_\mathrm{a}(t),\, \boldsymbol{u}(t),\, t\right) \tag{5.51}$$

$$\boldsymbol{0} = \boldsymbol{m}\left(\boldsymbol{x}_\mathrm{d}(t),\, \boldsymbol{x}_\mathrm{a}(t),\, \boldsymbol{u}(t),\, t\right) \tag{5.52}$$

$$\dot{\boldsymbol{x}}_\mathrm{d}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}_\mathrm{d}(t), \tag{5.53}$$

where the known variables are $\mathcal{K} = \{\boldsymbol{x}_\mathrm{a}, \dot{\boldsymbol{x}}_\mathrm{d}\}$, the unknown variables are $\mathcal{X} = \{\boldsymbol{x}_\mathrm{d}, \boldsymbol{u}\}$ and the constraints are $\mathcal{C} = \{\boldsymbol{g}, \boldsymbol{m}, \frac{\mathrm{d}}{\mathrm{d}t}\}$. As the initial conditions $\boldsymbol{x}_\mathrm{d}(0)$ are known, derivative as well as integral causality can be used.

Theorem 5.7 (Reachability condition) *A necessary condition for system (5.51)–(5.53) to be structurally controllable is that the vertices of $\mathcal{K}$ can be reached in the structure graph from the input $\boldsymbol{u}$.*

This condition says that there does not exist any subsystem whose dynamical behaviour is independent of the input. The "no over-constrained subsystem" condition cannot be extended to the general case, but it holds for linear systems. For simplicity, let us drop algebraic equations, and consider the system (5.54), (5.55) with the known variables $\mathcal{K} = \{\dot{\boldsymbol{x}}_\mathrm{d}\}$, the unknown variables $\mathcal{X} = \{\boldsymbol{x}_\mathrm{d}, \boldsymbol{u}\}$, and the constraints $\mathcal{C} = \{\boldsymbol{g}, \frac{\mathrm{d}}{\mathrm{d}t}\}$.

$$\dot{\boldsymbol{x}}_\mathrm{d}(t) = \boldsymbol{g}\left(\boldsymbol{x}_\mathrm{d}(t),\, \boldsymbol{u}(t),\, t\right) \tag{5.54}$$

$$\dot{\boldsymbol{x}}_\mathrm{d}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}_\mathrm{d}(t). \tag{5.55}$$

Theorem 5.8 (Linear continuous systems) *If the constraints $\boldsymbol{g}$ are linear, necessary and sufficient conditions for system (5.54), (5.55) to be structurally controllable are the following:*

(i) The vertices of $\mathcal{K}$ are reachable in the structure graph from the input,
(ii) the canonical decomposition of $(\mathcal{C}_\mathcal{X},\ \mathcal{X},\ \mathcal{E}_\mathcal{X})$ contains no over-constrained subsystem.

The existence of an over-constrained subsystem would imply that the known variables (here $\dot{\boldsymbol{x}}_\mathrm{d}$) satisfy some compatibility conditions. For linear systems, these would be expressed as

$$\boldsymbol{\alpha}^{\mathrm{T}}\dot{\boldsymbol{x}}_{\mathrm{d}}(t) = 0, \tag{5.56}$$

where $\boldsymbol{\alpha}$ is some constant vector, from which it follows that any system trajectory would belong to the manifold

$$\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{x}_{\mathrm{d}}(t) - \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{x}_{\mathrm{d}}(t_0) = 0.$$

Consequently, it is not possible to drive the system state to any point in the state space.

Condition (ii) does not extend to nonlinear systems, because in order to define a manifold the compatibility constraints (5.56) which would now be nonlinear should also be integrable. This property does not follow from structural considerations.

5.8 Structural Analysis in Summary

Structural analysis is an important tool, which is of interest in the early stage of the control and supervision system design. It can be employed even before detailed models are available, the structural analysis only needs that the principal behaviours of a system are specified in order to perform a useful and quite comprehensive analysis. Diagnosability and isolability of a behavioural fault (violation of a constraint) in a system can be made based on such sparse information. Analytical redundancy relations for use in diagnosis can be generated either from a complete matching and subsequent backtracking through the matching to known variables, or by using the minimal structurally over-determined (MSO) sets approach followed by a similar backtracking.

Disturbances or unknown parameters are handled in a structural analysis by defining such unknown quantities as additional unknown variables. When performing a matching, one additional constraint will be needed in the just-determined subsystem to calculate each additional unknown input. This means the available ARRs will reduced in number but will be insensitive to these unknown quantities. It is a salient feature of structural analysis that it generates ARRs equally well for linear and nonlinear systems.

The fault diagnosis and fault-tolerant control results it provides are the identification of the diagnosable part of the system, and the identification of the reconfiguration possibilities of the estimation the control scheme. Since detailed behaviour models need only to be developed for those parts of the system, structural analysis is also a tool for deciding which modelling investments must be done for the design of the control and supervision system.

The structural properties hold for the class $\mathcal{S}(\mathcal{G})$ defined by the structure graph $\mathcal{G}$ and, hence, for "almost all" single systems included in this class. Only in exceptional cases, the system under consideration does not have a property that the structural analysis has found for the corresponding class. This relation has been demonstrated in this chapter by several examples.

Observability analysis is the main step to identify the diagnosable part, which is the over-constrained subsystem within the observable one. Furthermore, structural analysis not only provides the computation mechanisms for the estimation algorithms and their reconfiguration, but it can also suggest which sensors should be implemented so as to change the status of system components from undiagnosable to diagnosable.

Structural analysis cannot help in defining fault accommodation strategies, because these strategies are aimed at investigating the means of achieving the system objectives, in spite of faults, without changing its structure. On the contrary, structural analysis is of prime importance as far as reconfiguration is concerned, because the results are expressed with reference to graph properties, whose changes can be analysed when vertices and edges disappear, as the consequence of switching off some system components, after a fault has occurred.

In summary, the following algorithm describes the design procedure for diagnosis based on structural analysis.

Algorithm 5.6 *Structural analysis aiming at diagnosis*

Given: A set $\mathcal{C}$ of constraints
The sets $\mathcal{X}$ and $\mathcal{K}$ of unknown and known variables
The sets $\mathcal{U}$ and $\mathcal{Y}$ of inputs and outputs with $\mathcal{U} \cup \mathcal{Y} \subseteq \mathcal{K}$

1. Determine the *structure graph* $\mathcal{G}$

2. Find a complete matching on the unknown variables to get a proper over-determined set of constraints

3. Mark unmatched constraints c_j^u for use as analytical redundancy relations

3.a Alternatively, use MSO sets to find c_j^u

4. Express c_j^u as a function of known variables using backtracking through the matching

5. Express residuals as $r_i = c_j^u$

6. Determine the dependency mapping $\boldsymbol{r} = \boldsymbol{M}\boldsymbol{c}^u$ for the set of residuals and unmatched constraints

7. Test structural detectability and observability from the columns of $\boldsymbol{M}$

8. If isolability conditions are not satisfied, investigate whether the active isolability approach can enhance isolation of faults

9. Insert the analytical expressions of constraints in the result of Step 5 to get residuals in analytical form

Results: List of the existing analytic redundancy relations based on order in which constraints are used (symbolic form)
List of detectable faults
List of isolable faults
List of residuals in analytical form obtained through backtracking

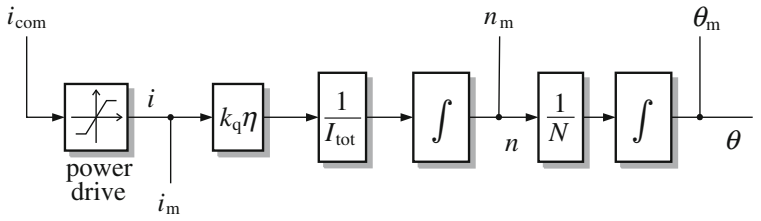


Fig. 5.36 Position actuator open loop

5.9 Exercises

Exercise 5.1 Structural analysis for industrial actuator
Make a structural model of the actuator shown in Fig. 5.36.

1. Determine the sets $\mathcal{K}$ (known variables), $\mathcal{X}$ (unknown variables) and $\mathcal{Z}$ (all variables).
2. List the set of constraints that describe the system shown in Fig. 5.36.
3. Derive the incidence matrix and draw the structure graph.
4. Ignore causality and determine a complete matching on $\mathcal{X}$ that is non-causal.
5. Use the ranking algorithm to determine a complete causal matching on $\mathcal{X}$. List the unmatched constraints.
6. Determine the parity relations found from the unmatched constraints by backtracking the structure graph to known variables along the paths of the matching,

$$c_i(\mathcal{K}_i) = 0 \wedge \mathcal{K}_i \subseteq \mathcal{K}.$$

7. Express the parity relations in analytical form using the constraints from question 2. □

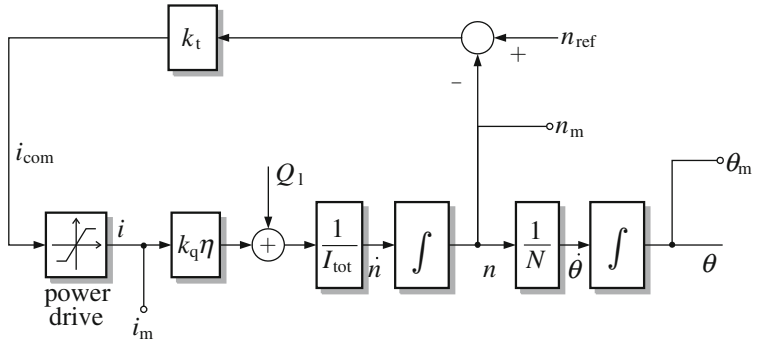


Fig. 5.37 Block diagram of DC motor with load torque and closed speed loop

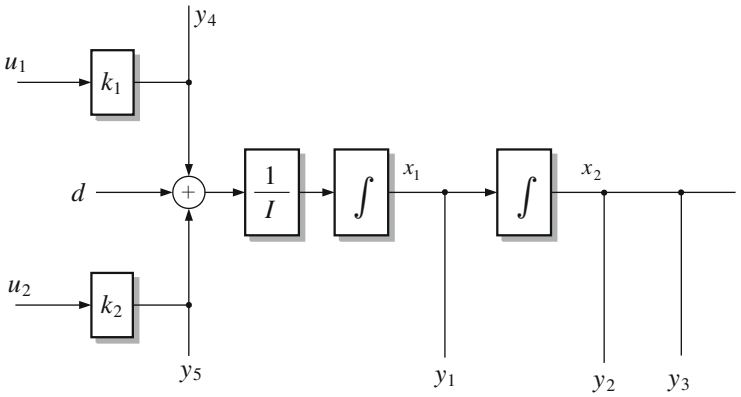


Fig. 5.38 Block diagram of single-axis satellite with input from two redundant actuators, redundant measurements of attitude (angle), measurement of angular rate and measurement of delivered actuator torques

Exercise 5.2 Structural analysis with unknown input
Consider the speed control loop of Fig. 5.37, where n_{ref} is the reference speed.

1. Using the known variables
$$\mathcal{K} = \{i_m, n_m, \theta_m, n_{\text{ref}}\}$$
and the unknown variables
$$\mathcal{X} = \left\{i, Q_l, n, \dot{n}, \theta, \dot{\theta}\right\},$$
determine the set of constraints that describe the system.
2. Build the structure graph for the system. Describe the graph as an incidence matrix and draw the graph.
3. Apply the ranking algorithm on the graph to determine at least one causal matching. List which constraints remain unmatched.
4. For each unmatched constraint, determine a parity relation $c_i(\mathcal{K}_i) = 0,\ \mathcal{K}_i \subseteq \mathcal{K}$. □

Exercise 5.3 Parity relations for single-axis satellite
This exercise considers structural analysis for a single-axis satellite described by the block diagram in Fig. 5.38. The figure illustrates a single axis of a satellite.

There are two input signals u_1 and u_2 to actuators 1 and 2, respectively, one unknown input d, and five measurements: y_1 measures the state x_1, y_2 and y_3 measure x_2; y_4 and y_5 measure torque from actuators 1 and 2, respectively.

1. Determine the sets of known variables, $\mathcal{K}$, and unknown variables $\mathcal{X}$. Verify that the intersection $\mathcal{Z} = \mathcal{K} \cup \mathcal{X}$ gives the total set of variables.
2. Determine the set of constraints that describe the system.
3. Determine the causal structure graph for the system. Represent the graph as an incidence matrix and as a drawing.

4. Use the ranking algorithm on the graph to find one or more complete matchings. List which constraints remain unmatched.
5. From the unmatched constraints, determine the parity relations in analytic form:

$$c_i(\mathcal{K}_i) = 0,\ \mathcal{K}_i \subseteq \mathcal{K}.$$

You may wish to use the MATLAB programme *SaTool* to cope with the complexity of matching or for checking your results. A GNU open source license of *SaTool* is available from the book homepage. □

Exercise 5.4 Parity relations and addition of a sensor
Let a system be composed of 3 interconnected components, c_1, c_2, c_3. Each component is described by one constraint according to the system

$$\begin{aligned} c_1 &: \ \dot{x}_1(t) - x_1(t) = 0 \\ c_2 &: \ \dot{x}_1(t) - 2\dot{x}_2(t) = 0 \\ c_3 &: \ y(t) + 3x_1(t) - x_2(t) = 0. \end{aligned}$$

The variables x_1, x_2 which characterise the operation of components c_1, c_2 are not measured, only the output y of component c_3 is known.

1. Draw the structure graph of the system.
2. Find a redundancy relation which allows to detect a fault in one of the components.
3. Would it be worth to add a fourth component, that would measure x_1 according to

$$c_4 : z(t) = x_1(t)$$

z is now an extra known variable, but of course component n°4 may also be faulty. □

Exercise 5.5 A specialised arithmetic circuit
The following specialised computation circuit is composed of 3 multipliers M_1, M_2 and M_3 and two adders A_1 and A_2 (Fig. 5.39).

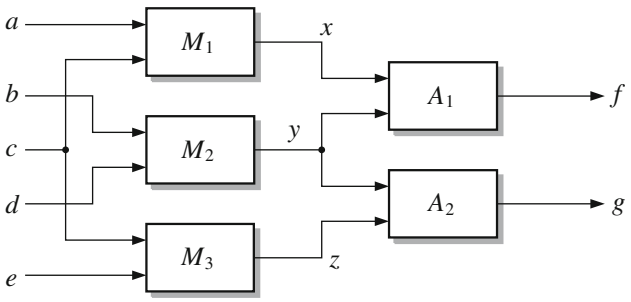


Fig. 5.39 Specialised computation circuit

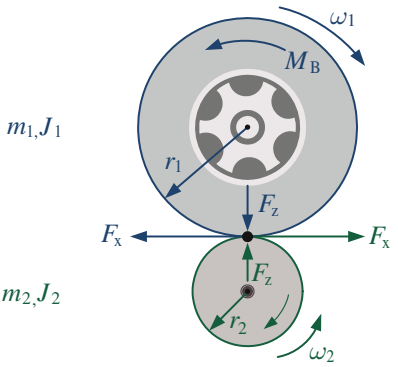


Fig. 5.40 Schematic representation of an ABS test bed

1. Write the model of each system component.
2. Give the incidence matrix of the structure graph (distinguish the known and the unknown variables).
3. Find the analytical redundancy relations by eliminating the unknown variables.
4. For each ARR, give the list of the components the faults of which it is sensitive to.
5. Is there any non-detectable or non-isolable fault?
6. What are the possible diagnostics associated with the following measurements?.

a	b	c	d	e	f	g
2	2	3	3	2	10	12

□

Exercise 5.6 Diagnosability analysis of an ABS test bed
An ABS (anti-lock braking system) test bed is schematically drawn in Fig. 5.40. In a simplified version, the test bed has two wheels, where the lower wheel is powered by a motor, whereas the upper wheel has a brake with the braking torque M_B. The wheel angular velocities are denoted by ω_1 and ω_2, the forces between the wheels by F_z, the lateral force by F_x and the masses and inertias by m_1, m_2, J_1 and J_2. The wheel vertical force F_z is determined by the geometry of the test bed including the air pressure in the tyres (which may be too low by fault).

The model of the test bed is given below:

$$
\begin{aligned}
c_1: &\quad J_1\dot{\omega}_1(t) = -M_B(t) + M_L(t)\\
c_2: &\quad M_L(t) = F_x(t) r_1\\
c_3: &\quad F_x(t) = \mu(t) F_z\\
c_4: &\quad \mu(t) = f(\lambda(t)) \quad \text{(Slip diagram)}\\
c_5: &\quad \lambda(t) = 1 - \frac{\omega_2(t) r_2}{\omega_1(t) r_1}\\
c_6: &\quad J_2\dot{\omega}_2(t) = -M_L(t)\\
c_7: &\quad \dot{\omega}_1(t) = \frac{\mathrm{d}\omega_1(t)}{\mathrm{d}t}\\
c_8: &\quad \dot{\omega}_2(t) = \frac{\mathrm{d}\omega_2(t)}{\mathrm{d}t}.
\end{aligned}
$$

Measurable signals are $M_B(t), \omega_1(t), \omega_2(t)$ whereas the signals $\dot{\omega}_1(t), \dot{\omega}_2(t)$ may be measured if this is necessary for fault diagnosis.

1. Draw the structure graph.
2. Analyse the test bed and determine analytical redundancy relations for fault diagnosis.
3. Which signals have to be measured to make the test bed detectable or isolable? □

5.10 Bibliographical Notes

Offering a way to advise how to solve large sets of equations, structural concepts and bipartite graphs were introduced and seminal theoretical results for bipartite graphs were obtained in [17, 86, 87]. The structural approach was first brought into the field of fault in [76].

Decomposition of large systems. Structural concepts have been used since the 1960 and 1970s for the decomposition of large systems of equations in view of their hierarchical resolution [142, 341]. An important issue in that field is also the solvability of large scale differential and algebraic equation systems, for which [193, 371] addressed and employed structural analysis.

Algorithms. Algorithms to compute maximum matchings were studied along with the penetration of electronic computers into engineering research motivated by important applications in operational analysis and in chemical engineering. An algorithm of complexity $O(\mathcal{N}^3)$ to find maximum matchings was proposed in [89], while [150] found an algorithm of complexity $O(\mathcal{N}^{2.5})$ for bipartite graphs. Maximum matchings can also be found from the solutions to the assignment problem [189], or from the maximum flow problem [110, 111]. For details on the algorithms and more bibliographical notes, refer to [17, 62, 131, 206]. Theorem 5.2 was proved in [17].

Looking into maximal isolability and minimum computational complexity, it turned out that finding all possible analytical redundancy relations through finding all possible complete matchings was impractical if not impossible for industrial size systems. Inspired by experience from automotive diagnosis, references [184, 187] proposed to find MSO sets as a more direct way to determined all possible ARRs for a given system and an alternative decomposition of the structure graph and an extremely efficient algorithm (cf. Algorithm 5.4).

Observability, controlability. The technique has also been used for analysis of system structural properties like observability and controllability, where most works use a digraph representation and address linear systems [130, 195, 196, 232]. They have also been extended to the design of multivariable control systems, including considerations like disturbance rejection [286, 301].

Fault diagnosis. In the field of fault diagnosis, structural concepts have been used since the beginning of the 1990s, for the analysis of system monitorability [76] and

for the design of structured residuals [126], which provide straightforward decision procedures for fault isolation [69]. An overview can be found in [325].

Realisability and optimization. Issues with realisation of residual generators in large systems caused the development of selection procedures [19], continued by implementations with mixed causality in [337, 352] considered further into issues of causal computations. Furthermore, [354] suggested algorithms for realizability constrained selection of residual generators.

Applications. Significant applications in marine systems were described in [23, 28, 159]. A significant effort related to diagnosis in car engines and for other automotive applications were reported in [350, 352]. Application to large 3-phase systems was discussed in [180]. Diagnosis to determine downhole drilling incidents in [387] and combined diagnosis, active fault isolation and fault-tolerant control was demonstrated for thruster assisted position mooring for offshore production vessels in [238].

Finally, structural concepts have been applied to the problem of sensor selection [57, 230], for component-oriented analysis [370] and for service diagnosis [372, 373].

Relations to AI. The Artificial Intelligence approach to causality in device behaviour [158], which is used in the theory of model-based diagnosis, is also very close to the concept of matching in bipartite graphs. Since the obtained models are mainly under a graphic form, the theory of bond graphs has brought about many specific tools for structural analysis.

Multiple faults and active isolation. Structural analysis was also found useful to cope with the complexity of analysis in cases of multiple faults [23]. An extension of the structural analysis to advise on possibilities of active isolation was suggested in [33] with an application reported in [238].

Algorithms and software tools. The SaTool software environment (GNU public license) that has been used for several examples in this book was introduced in [31]. A large framework for diagnosis design in the automotive industry was presented in [107]. Efficient algorithms for finding structurally minimal over-determined sets were suggested in [187], and [6] compared different algorithms.

Part II
Continuous-Variable Systems

Chapter 6
Fault Diagnosis of Deterministic Systems

Abstract This chapter provides solutions to the fault detection, isolation and estimation problems for systems described by deterministic continuous-variable models. The chapter considers faults that can be modelled as additive signals acting on the process. The methods presented lead to a diagnostic system which is separated in two parts: a residual generation module and a residual evaluation module.

6.1 Introduction

Continuous-variable models (or analytical models) consist of sets of differential or difference equations. They can be deduced by application of the laws of physics, chemistry, etc. to the supervised and/or controlled process. The external variables entering these equations are called inputs. One distinguishes control inputs, which are known and can be manipulated, from disturbances which cannot be manipulated. Disturbances that are not measured are called unknown inputs. Besides, imperfections in the model and measurement noise may be represented by stochastic processes (or sequences) appearing as additional inputs. When such random input is used, one speaks about stochastic models, as opposed to deterministic models. This chapter and the next one will show the design of fault detection, isolation and/or estimation systems for processes described by deterministic continuous-variable models with unknown input. Such systems are made of two parts as already indicated in Chap. 1: a residual generation module and a residual evaluation module (or decision system) (Fig. 6.1).

The residuals are signals that, in the absence of faults, deviate from zero only due to modelling uncertainties, with nominal value being zero, or close to zero under actual working conditions. If a fault should occur, the residuals deviate from zero with a magnitude such that the new condition can be distinguished from the fault free working mode. The role of the decision system is to determine whether the residuals differ significantly from zero and, from the pattern of zero and non-zero residuals, to decide which is the most likely fault being present, if any, and in turn, isolate which component(s) could be the origin of a fault. When the diagnostic system is used in a

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8_6

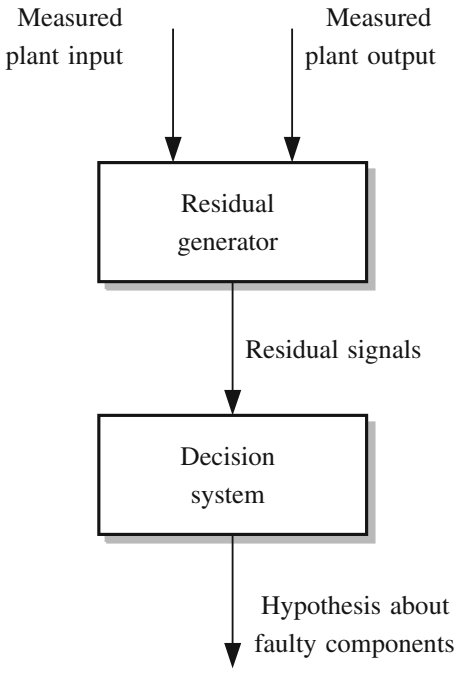


Fig. 6.1 Structure of a fault diagnosis system

fault-tolerant controller, as described in Chaps. 1 and 8, details in the diagnostic task will depend on the type of faulty device and on the way the faulty condition could be treated.

Sensor faults can often be handled through estimating the faulty output signal using an estimator based on other available measurements less the one isolated as faulty. Observability of the reduced system is naturally required in this case. For this type of sensor fault, the diagnostic system needs only to perform fault detection and isolation to determine which measured signals should be disregarded. For an actuator fault which does not cause a complete loss of command, a remedial action could be to modify the control signal to the set of actuators by an increment computed in such a way that the fault is compensated. In this case, an estimate of the fault signal is needed.

The fundamental notion on which residual generation for continuous-variable systems rests is analytical redundancy. Analytical redundancy relations are equations that are deduced from an analytical model, which solely use measured variables as input. Analytical redundancy relations must be consistent in the absence of a fault, and can thus be used for residual generation. A simple example is given to introduce this notion, before considering more formal developments in subsequent sections.

Example 6.1 Residuals for the ship autopilot
Consider, the following part of the ship autopilot example (see Sect. 2.2). The turn rate ω_3 and the heading angle ψ are related through

$$\dot{\psi}(t) = \omega_3(t). \tag{6.1}$$

Let us neglect the effect of waves and assume that the measurements can only be affected by a bias. Hence, sensor faults are represented by additive signals and the measurement equations can be written:

$$\psi_m(t) = \psi(t) + f_\psi(t) \tag{6.2}$$
$$\omega_{3m}(t) = \omega_3(t) + f_\omega(t) \tag{6.3}$$

where the index m denotes measured quantities, and $f_\psi(t)$, $f_\omega(t)$ are the potential biases. Since most supervision systems are implemented as a software, only sampled data are available. They are linked through the following discrete model deduced from (6.1):

$$\psi(k+1) = \psi(k) + \omega_3(k)T_s, \tag{6.4}$$

where T_s stands for sampling period. By considering the equation error, r, resulting from (6.4) when the variables are substituted by their measured value, the following expression is obtained:

$$r(k) = \psi_m(k) - \psi_m(k-1) - \omega_{3m}(k-1)T_s. \tag{6.5}$$

This quantity has the properties expected for a residual. Indeed, introducing (6.2), (6.3) into (6.5) yields

$$r(k) = f_\psi(k) - f_\psi(k-1) - f_\omega(k-1)T_s.$$

This shows that, in the absence of a fault (namely when $f_\psi(k) = f_\psi(k-1) = f_\omega(k-1) = 0$), $r(k)$ is zero. Upon occurrence of a bias in the measurement of ω_3 say at time k_0, $r(k)$ takes a constant non-zero value for all $k \geq k_0$. Finally, the appearance of a bias on the measurement of ψ at time instant k_0 shows up as a spike at time k_0, but has no permanent effect on r. Both faults thus affect r and this signal is zero in the absence of fault. Hence it can be named a residual signal. For decision making, it suffices to compare the residual to a specified threshold. The latter should be chosen in such a way that biases that appear to be significant for the considered application are detected.

When measurement noise is significant, comparison to a simple threshold might not be practicable, because the change in the mean of the residual due to the fault can be hidden by the effect of the noise on the residual. This noise needs to be taken into account as described in the following two discretised "noisy" versions of (6.2), (6.3):

$$\psi_m(k) = \psi(k) + f_\psi(k) + v_\psi(k) \tag{6.6}$$
$$\omega_{3m}(k) = \omega_3(k) + f_\omega(k) + v_\omega(k), \tag{6.7}$$

where $v_\psi(i), v_\omega(i), i = 1, 2, \ldots$ are mutually uncorrelated white noise sequences with variance $E(v_\psi^2(k)) = Q_\psi$ and $E(v_\omega^2(k)) = Q_{\omega_3}$ respectively.

Substituting (6.6) and (6.7) into (6.5) yields:

$$r(k) = f_\psi(k) - f_\psi(k-1) - f_\omega(k-1)T_s + v_\psi(k) - v_\psi(k-1) - v_\omega(k-1)T_s.$$

$(r(1), \ldots, r(k))$ is now a random sequence which must be evaluated by suitable algorithms. Only its mean value is equal to zero in the absence of fault. □

The difference of treatment between deterministic and stochastic models is reflected in the organisation of the chapter: Sects. 6.2–6.4 deal with the first class of models and Sects. 6.5 and 6.6 with the second one. Analytical redundancy relations (ARR) based on a deterministic model were already addressed in the previous chapter where structural models were used for their determination. The link with this chapter is the object of Sect. 6.2 were the principle of the determination of ARR from a deterministic nonlinear state-space model is presented. The particular case of a deterministic linear state-space model is considered in Sect. 6.3, and a complete algorithm is provided to design a specific type of analytical redundancy relations. A more general presentation of parity relations for fault detection, isolation and estimation is then presented in Sect. 6.4 from a linear input-output model of the process.

The methods of Sects. 6.2 and 6.3 assure perfect insensitivity (or decoupling) of the residuals from an unknown input. When this is not possible, e.g. due to model uncertainty, or small amounts of noise on sensors or on the process, approximate disturbance decoupling of the residual with respect to the unknown input can be obtained by an optimisation approach. This is the subject of Sect. 6.5. Finally, Sect. 6.6 treats residual evaluation for the deterministic case.

6.2 Analytical Redundancy in Nonlinear Deterministic Systems

6.2.1 Logical Background

Analytical redundancy can be seen as a tool for obtaining conditions, based on available measurements, that are necessarily fulfilled when the supervised system works in a specific operating mode. In order to illustrate the principle of analytical redundancy, consider deterministic systems described in normal operation by state and measurement equations

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta, t) \tag{6.8}$$

$$\boldsymbol{y}(t) = \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta, t), \tag{6.9}$$

where $\boldsymbol{x} \in |\mathcal{R}^n$ is the state vector, which is not available, $\boldsymbol{u} \in |\mathcal{R}^m$ is the control input vector, $\boldsymbol{d} \in |\mathcal{R}^{n_\mathrm{d}}$ is an uncontrolled deterministic vector (disturbance). θ is a parameter vector which is considered to be known, and $\boldsymbol{y} \in |\mathcal{R}^p$ is the measurement vector. Let $\mathcal{H}_0$ be the situation corresponding to normal operation, and $\mathcal{H}_1 = \neg\mathcal{H}_0$ some faulty situation. The following logical statements are true

$$\mathcal{H}_0 \iff [\dot{\boldsymbol{x}}(t) = \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta, t)] \wedge \left[\boldsymbol{y}(t) = \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta, t)\right]$$

$$\mathcal{H}_1 \iff [\dot{\boldsymbol{x}}(t) \neq \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta(t))] \vee \left[\boldsymbol{y}(t) \neq \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta(t))\right].$$

The violation of equality constraints that results from faults may be described in two ways:

- In the first option, faults are assumed to result from parametric variations, which is represented as

$$\theta_f(t) \neq \theta \iff \theta_f(t) = \theta + \boldsymbol{f}(t), \boldsymbol{f}(t) \neq 0,$$

  where $\theta_f(t)$ stands for the parameter vector associated with the faulty system.
- In the second option, no hypothesis is made about the origin of the discrepancy, which is just represented as an additive vector

$$\begin{array}{l}[\dot{\boldsymbol{x}}(t) \neq \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta(t))] \vee \left[\boldsymbol{y}(t) \neq \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta(t))\right] \\ \iff \\ \exists \left(\boldsymbol{f}_{\mathrm{x}}, \boldsymbol{f}_{y}\right) \neq (0,0): \\ \quad \left[\dot{\boldsymbol{x}}(t) = \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta(t)) + \boldsymbol{f}_{\mathrm{x}}(t)\right] \\ \quad \vee \left[\boldsymbol{y}(t) = \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta(t)) + \boldsymbol{f}_{y}(t)\right].\end{array}$$

In both cases, the normal and the faulty system are represented using some "fault vector" $\boldsymbol{f}(t)$ where normal operation is associated with $\boldsymbol{f}(t) = 0$. Most often, the preliminary analysis of the system has identified a set of faults that are likely to occur, and that the FDI system to be designed should detect, isolate and estimate. When such knowledge is available, it results in the logical statement

$$i \in I : \mathcal{H}_i \iff \boldsymbol{f}(t) = \boldsymbol{f}_i(\eta_i, t) \neq 0,$$

where $\mathcal{H}_i$ denotes the ith fault situation, $I = \{1, 2, \ldots, n_f\}$ where n_f is the number of possible fault modes, and the knowledge available about each fault is modelled by the possible time evolution of the vector $\boldsymbol{f}$ which depends on some unknown parameters η_i (fault estimation therefore directly refers to the estimation of these parameters).

6.2.2 Analytical Redundancy Relations with No Unknown Inputs

Introducing the fault vector $\boldsymbol{f}(t)$ in the state and measurement equations, and setting $\boldsymbol{d}(t) = 0$, for all t one gets [1]

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{f}(t)), \quad \boldsymbol{y}(t) = \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{f}(t)), \tag{6.10}$$

where, since θ is known, the dependency of the state and measurement equations on the parameter is no longer made explicit, and time-invariant systems are considered

[1] The same symbols $\boldsymbol{g}$ and $\boldsymbol{h}$ as in (6.8), (6.9) are used by an abuse of notation.

in order to shorten the notations. It turns out that from (6.10), it is possible to construct *residuals*, i.e. quantities which can be computed in real time from the available data, and whose behaviour is different under the different situations $\mathcal{H}_0$ and $\mathcal{H}_1$. Such residuals are obtained from a two step construction:

Step 1: Derivation of the outputs. Assuming that all functions are differentiable with respect to their arguments, it is possible to construct the derivative $\dot{\boldsymbol{y}}(t)$ of the output signal $\boldsymbol{y}(t)$:

$$\dot{\boldsymbol{y}}(t) = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}(\cdot)\dot{\boldsymbol{x}}(t) + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{u}}(\cdot)\dot{\boldsymbol{u}}(t) + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{f}}(\cdot)\dot{\boldsymbol{f}}(t)$$

Replacing $\dot{\boldsymbol{x}}(t)$ by its value, one gets

$$\begin{aligned}\dot{\boldsymbol{y}}(t) &= \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}(\cdot)\boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{f}(t)) + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{u}}(\cdot)\dot{\boldsymbol{u}}(t) + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{f}}(\cdot)\dot{\boldsymbol{f}}(t) \\ &:= \boldsymbol{h}_1(\boldsymbol{x}(t), \bar{\boldsymbol{u}}^{(1)}(t), \bar{\boldsymbol{f}}^{(1)}(t)),\end{aligned}$$

where $\bar{\boldsymbol{u}}^{(1)}(t)$ is a short notation for $\left(\boldsymbol{u}^{\mathrm{T}}(t), \dot{\boldsymbol{u}}^{\mathrm{T}}(t)\right)^{\mathrm{T}}$. Iterating this process until some order of derivation q (to be determined later), and assuming the existence of all required derivatives, one obtains

$$\bar{\boldsymbol{y}}^{(q)}(t) = H^q\left(\boldsymbol{x}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)\right) \tag{6.11}$$

which is a set of $(q+1)\,p$ equations—or constraints—(the dimension of $\bar{\boldsymbol{y}}^{(q)}(t)$), where the different variables have the following dimensions: $\boldsymbol{x} \in \mathbb{R}^n$, $\bar{\boldsymbol{u}}^{(q)}(t) \in \mathbb{R}^{(q+1)\times m}$, $\bar{\boldsymbol{f}}^{(q)}(t) \in \mathbb{R}^{(q+1)\times n_f}$. The known variables are $\bar{\boldsymbol{y}}^{(q)}$ and $\bar{\boldsymbol{u}}^{(q)}$ while the unknown variables are $\boldsymbol{x}$. $\bar{\boldsymbol{f}}^{(q)}(t)$ has a particular status, since it is known (equal to zero) when $\mathcal{H}_0$ is true, while it is unknown when $\mathcal{H}_1$ is true.

Example 6.2 Redundancy in a nonlinear system
Applying the above procedure with $s = 2$ to the system

$$\begin{pmatrix}\dot{x}_1(t) \\ \dot{x}_2(t)\end{pmatrix} = \begin{pmatrix}-x_1(t) + x_2^2(t) + u(t) + f_1(t) \\ -2x_2(t) + f_2(t)\end{pmatrix}$$
$$y(t) = x_1(t) + f_3(t)$$

gives

$$\begin{aligned}\dot{y}(t) &= -x_1(t) + x_2^2(t) + u(t) + f_1(t) + \dot{f}_3(t) \\ \ddot{y}(t) &= x_1(t) - 5x_2^2(t) - u(t) - f_1(t) + 2x_2(t)f_2(t) + \dot{u}(t) + \dot{f}_1(t) + \ddot{f}_3(t).\end{aligned}$$

(6.11) is thus a system of three equations

$$\bar{y}^{(2)}(t) = \begin{pmatrix} y(t) \\ \dot{y}(t) \\ \ddot{y}(t) \end{pmatrix} \tag{6.12}$$
$$= \begin{pmatrix} x_1(t) + f_3(t) \\ -x_1(t) + x_2^2(t) + u(t) + f_1(t) + \dot{f}_3(t) \\ x_1(t) - 5x_2^2(t) - u(t) - f_1(t) + 2x_2(t)f_2(t) + \dot{u}(t) + \dot{f}_1(t) + \ddot{f}_3(t) \end{pmatrix}. \square$$

Step 2: Elimination of the state. Assume that $(q+1)\,p > n$ and the Jacobian $\frac{\partial H^q(\cdot)}{\partial \boldsymbol{x}}$ is of rank n. Note that the first condition gives a lower bound on the order of derivation that is necessary in establishing (6.11). It follows that (6.11) can be decomposed into

$$\begin{pmatrix} \bar{\boldsymbol{y}}_m^{(q)}(t) \\ \bar{\boldsymbol{y}}_{nm}^{(q)}(t) \end{pmatrix} - \begin{pmatrix} H_m^q\left(\boldsymbol{x}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)\right) \\ H_{nm}^q\left(\boldsymbol{x}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)\right) \end{pmatrix} = \boldsymbol{0} \tag{6.13}$$

where the first subsystem is of dimension n and allows to compute $x(t)$ (at least locally) as a function of the other variables

$$\boldsymbol{x}(t) = \phi(\bar{\boldsymbol{y}}_m^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t))$$

(this results from the implicit function theorem). Replacing $\boldsymbol{x}(t)$ by its value in the second subsystem, which is of dimension $(q+1)\,p - n$, one obtains a system that is equivalent to (6.11)

$$\boldsymbol{x}(t) = \phi(\bar{\boldsymbol{y}}_m^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)) \tag{6.14}$$
$$\boldsymbol{0} = \boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)), \tag{6.15}$$

where

$$\boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t))$$
$$= \bar{\boldsymbol{y}}_{nm}^{(q)}(t) - H_{nm}^q(\phi(\bar{\boldsymbol{y}}_m^q(t)), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)).$$

The set of constraints (6.15) is seen to contain only inputs, outputs and fault signals (along with their derivatives). It is called an analytical redundancy relations (ARR) associated with the pair $(\boldsymbol{g}, \boldsymbol{h})$ and $\boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}, \bar{\boldsymbol{u}}^{(q)}, \bar{\boldsymbol{f}}^{(q)})$ is called the residual vector.

Remark 6.1 (*Link to structural approach*) A structural condition for ARR to exist is that (6.11) is overconstrained with respect to the unknowns $\boldsymbol{x}(t)$, i.e. there is a matching which is complete with respect to $\boldsymbol{x}(t)$. Decomposing the set of constraints (6.11) into matched (index m) and non-matched ones (index nm) yields (6.13), where the matched subsystem has n constraints while the non-matched subsystem has $(q+1)\,p - n$ constraints. From the interpretation of matchings in the previous chapter, $\boldsymbol{x}(t)$ is computed in the matched subsystem, as a function of the other

variables $\phi\left(\bar{\boldsymbol{y}}_m^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)\right)$ and replacing $\boldsymbol{x}(t)$ by its value in the non-matched subsystem gives the redundancy relations. □

Example 6.2 (cont.) Redundancy in a nonlinear system
Step 2 is now applied to (6.12). The variable t is omitted below. The state $(x_1, x_2)^{\mathrm{T}}$ can be computed from the first two equations of (6.12) leading to the equivalent system

$$\begin{aligned} x_1 &= y - f_3 \\ x_2 &= \pm\sqrt{\dot{y} + y - f_3 - u - f_1 - \dot{f}_3} \\ 0 &= \ddot{y} - y + f_3 + 5\left(\dot{y} + y - f_3 - u - f_1 - \dot{f}_3\right) + u + \cdots \\ &\quad \cdots + f_1 + 2\left(\sqrt{\dot{y} + y - f_3 - u - f_1 - \dot{f}_3}\right) f_2 - \dot{u} - \dot{f}_1 - \ddot{f}_3, \end{aligned} \tag{6.16}$$

where the third equation is seen to depend only on the available inputs and outputs and on the faults. □

6.2.3 Unknown Inputs, Exact Decoupling

When unknown inputs are present, a state-space model of the system takes the form[2]

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \boldsymbol{f}(t)) \\ \boldsymbol{y}(t) &= \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \boldsymbol{f}(t)). \end{aligned} \tag{6.17}$$

Applying the same technique as above leads to

$$\bar{\boldsymbol{y}}^{(q)}(t) = H^q\left(\boldsymbol{x}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{d}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)\right). \tag{6.18}$$

Under the condition that $(q+1)\,p > n + (q+1)\,n_{\mathrm{d}}$ and the Jacobian

$$\left[\begin{array}{cc} \frac{\partial H^q(\cdot)}{\partial \boldsymbol{x}} & \frac{\partial H^q(\cdot)}{\partial \bar{\boldsymbol{d}}^{(q)}} \end{array}\right]$$

is of rank $n + (q+1)\,n_{\mathrm{d}}$ both the state and the unknown inputs can be eliminated, leading to the equivalent system

$$\left(\begin{array}{c} \boldsymbol{x}(t) \\ \bar{\boldsymbol{d}}^{(q)}(t) \end{array}\right) = \left(\begin{array}{c} \phi_x(\bar{\boldsymbol{y}}_m^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)) \\ \phi_{\mathrm{d}}(\bar{\boldsymbol{y}}_m^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)) \end{array}\right) \tag{6.19}$$

$$0 = \boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)), \tag{6.20}$$

where (6.20) are the analytical redundancy relations, which are independent of the unknown inputs, hence the name "exact decoupling" which is given to this approach.

[2] Again the same functions g and h as above are used by an abuse of notation.

Note that exact decoupling is possible only if the structural graph of system (6.18) is overconstrained with respect to both the unknowns $\boldsymbol{x}$ and $\bar{\boldsymbol{d}}^{(q)}$.

6.2.4 How to Find Analytical Redundancy Relations

There are several procedures by which ARR can be found. They all rest on the elimination of $\boldsymbol{x}(t)$ (and $\bar{\boldsymbol{d}}^{(q)}(t)$ when unknown inputs are present), either by starting with (6.8) and (6.9) or by establishing first (6.11).

Elimination procedures fit the nature of the functions $\boldsymbol{g}$ and $\boldsymbol{h}$. When all functions are linear, projection approaches are well suited: this is the parity space approach which will be described in Sect. 6.3. Most often, nonlinear models involve polynomial functions (because polynomials can approximate any smooth nonlinear function). There are, basically, three elimination techniques for polynomial functions. All three require the components of the state to be eliminated according to some selected order. *Elimination theory* rests on Euclidean division and derivation. *Gröbner bases* uses Euclidean division and the computation of so-called S-polynomials. *Characteristic sets* (also called Ritt's algorithm) rest on Euclidean division and derivation. The state is directly eliminated from the system (6.8), (6.9), and ARR with minimum derivative order can be obtained.

6.2.5 ARR-based Diagnosis

Fault detection. In the absence of unknown inputs, or when exact decoupling is possible, the following logical statements hold

$$\begin{array}{lll} (6.10) \iff (6.14), (6.15) & \Rightarrow & \boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)) = \boldsymbol{0} \\ (6.17) \iff (6.19) & \Rightarrow & \boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)) = \boldsymbol{0} \end{array} \tag{6.21}$$

From (6.21) it follows that in both cases necessary conditions for normal system operation are given by

$$\mathcal{H}_0 \Rightarrow \boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), 0) = \boldsymbol{0}$$

Therefore, fault detection immediately follows from

$$\boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), 0) \neq 0 \Rightarrow \mathcal{H}_1.$$

Remark 6.2 (*Non detectable faults*) Note that $\boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), 0) = \boldsymbol{0}$ does not imply $\mathcal{H}_0$ since the condition expressed by the analytical redundancy relation is only necessary. In fact, $\boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), 0) = \boldsymbol{0}$ is to be read: $\mathcal{H}_0$ is not falsified by

the observations, or in other terms "it is not impossible that the system is healthy". In fact, special fault values that are not detectable through analytical redundancy could exist. They correspond to non-zero values of $\boldsymbol{f}(t)$ that yield $\boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)) = \boldsymbol{0}$. □

Example 6.2 (cont.) Redundancy in a nonlinear system
The redundancy relation in (6.16) writes

$$\begin{aligned}&\ddot{y} + 5\dot{y} + 4y - 4u - \dot{u}\\ &\quad = f_1 - 2\left(\sqrt{\dot{y} + y - f_3 - u - f_1 - \dot{f}_3}\right) f_2 + 4f_3 + \dot{f}_1 + 5\dot{f}_3 + \ddot{f}_3.\end{aligned}$$

Therefore, the residual is

$$r\left(\bar{y}^{(2)}, \bar{u}^{(2)}, 0\right) = \ddot{y} + 5\dot{y} + 4y - 4u - \dot{u}$$

and the fault detection rule is

$$\ddot{y} + 5\dot{y} + 4y - 4u - \dot{u} \neq 0 \Rightarrow \mathcal{H}_1. \square$$

Fault isolation. Fault isolation is approached in a similar way, by the design of structured residuals. Assume it is possible to separate the set of faults I into two subsets I_1 and I_2 such that $I = I_1 \cup I_2$. Set

$$\boldsymbol{f}(t) = (\boldsymbol{f}_{I_1}^{\mathrm{T}}(t) \;\; \boldsymbol{f}_{I_2}^{\mathrm{T}}(t))^{\mathrm{T}},$$

where only $\boldsymbol{f}_{I_1}(t)$ ($\boldsymbol{f}_{I_2}(t)$) is non-zero upon occurrence of a fault in I_1 (I_2). If the set of residuals can also be separated in two subsets

$$\boldsymbol{r}(\bar{y}^{(s)}(t), \bar{u}^{(s)}(t), \bar{f}^{(s)}(t)) = \begin{pmatrix} \boldsymbol{r}_1(\bar{y}^{(s)}(t), \bar{u}^{(s)}(t), \bar{f}^{(s)}(t)) \\ \boldsymbol{r}_2(\bar{y}^{(s)}(t), \bar{u}^{(s)}(t), \bar{f}^{(s)}(t)) \end{pmatrix}, \tag{6.22}$$

so that (a) $\boldsymbol{r}_1$ is insensitive to faults in I_2 but sensitive to faults in I_1 while (b) $\boldsymbol{r}_2$ is insensitive to faults in I_1 but sensitive to faults in I_2, then, as shown below, it is possible to distinguish between the occurrence of a fault from the class I_1 or I_2. The logical expressions corresponding to these assumptions are

$$(a)\quad \left.\begin{array}{l} \forall i \in I_1 : \boldsymbol{f}_{I_1}(t) = \boldsymbol{f}_i(\eta_i, t) = 0 \\ \exists i \in I_2 : \boldsymbol{f}_{I_2}(t) = \boldsymbol{f}_i(\eta_i, t) \neq 0 \end{array}\right\} \Rightarrow \begin{array}{l} \boldsymbol{r}_1(\bar{y}^{(s)}(t), \bar{u}^{(s)}(t), \bar{f}^{(s)}(t)) = 0 \\ \boldsymbol{r}_2(\bar{y}^{(s)}(t), \bar{u}^{(s)}(t), \bar{f}^{(s)}(t)) \neq 0 \end{array}$$

$$(b)\quad \left.\begin{array}{l} \exists i \in I_1 : \boldsymbol{f}_{I_1}(t) = \boldsymbol{f}_i(\eta_i, t) \neq 0 \\ \forall i \in I_2 : \boldsymbol{f}_{I_2}(t) = \boldsymbol{f}_i(\eta_i, t) = 0 \end{array}\right\} \Rightarrow \begin{array}{l} \boldsymbol{r}_1(\bar{y}^{(s)}(t), \bar{u}^{(s)}(t), \bar{f}^{(s)}(t)) \neq 0 \\ \boldsymbol{r}_2(\bar{y}^{(s)}(t), \bar{u}^{(s)}(t), \bar{f}^{(s)}(t)) = 0. \end{array}$$

There are four possible situations (logical 0 means $\boldsymbol{r} = 0$ while logical 1 means $\boldsymbol{r} \neq 0$) and the following conclusions are true.

r_1	r_2	Conclusion
0	0	$\mathcal{H}_0$ is not falsified (no fault is detected)
0	1	$\mathcal{H}_0$ is falsified by a fault $i \in I_2$
1	0	$\mathcal{H}_0$ is falsified by a fault $i \in I_1$
1	1	$\mathcal{H}_0$ is falsified by a fault $i \in I_1$ and a fault $j \in I_2$

Therefore, under (6.22), it is possible to isolate a fault within the subset I_1 or within the subset I_2. By designing several partitions of the set of faults into two classes it is obviously possible to isolate faults within smaller subsets that result from the intersections of all these partitions.

Remark 6.3 (*Non-isolable faults*) Only a limited number of partitions into two classes enjoying property (6.22) can be obtained for a given system. Therefore, it may happen that whatever the partition such that (6.22) holds, two given faults, say i and j are always in the same class. These faults always have the same effect on the analytical redundancy relations, and therefore, they are not isolable from each other, which means that every FDI conclusion will contain "the fault is either i or j (or both)". □

Example 6.3 Two-tank system
In Example 5.40, the set of constraints associated with the two-tank system components wrote

Pump:	$q_P = u.f(h_1)$
Tank 1:	$\dot{h}_1 = \frac{1}{A}(q_P - q_L - q_{12})$
Tank 2:	$\dot{h}_2 = \frac{1}{A}(q_{12} - q_2)$
Pipe between tanks ($h_1 > h_2$):	$q_{12} = k_1\sqrt{h_1 - h_2}$
Output pipe:	$q_2 = k_2\sqrt{h_2}$
Outflow measurement:	$q_m = k_m.q_2.$

The state-space equations are

$$\begin{pmatrix} \dot{h}_1 \\ \dot{h}_2 \end{pmatrix} = \begin{pmatrix} -\frac{k_1}{A}\sqrt{h_1 - h_2} + \frac{f(h_1)}{A}.u - \frac{1}{A}.q_L \\ \frac{k_1}{A}\sqrt{h_1 - h_2} - \frac{k_2}{A}\sqrt{h_2} \end{pmatrix} \tag{6.23}$$

and the measurement equation is

$$q_m = k_m k_2 \sqrt{h_2}. \tag{6.24}$$

Derivating once the output gives

$$\begin{aligned} \dot{q}_m &= k_m k_2 (h_2)^{-1/2} \dot{h}_2 \\ \dot{q}_m &= k_m k_2 (h_2)^{-1/2} \left(\frac{k_1}{A}\sqrt{h_1 - h_2} - \frac{k_2}{A}\sqrt{h_2} \right). \end{aligned} \tag{6.25}$$

From (6.24) and (6.25) the two states h_1 and h_2 can be computed

$$\begin{aligned} h_2 &= \left(\frac{q_m}{k_m k_2}\right)^2 \\ h_1 &= q_m^2 \left(1 + (1 + \dot{q}_m)^2\right). \end{aligned} \tag{6.26}$$

Derivating once again gives

$$\ddot{q}_m = \frac{(h_1 - h_2)^{-1/2} \sqrt{h_2} \left(\dot{h}_1 - \dot{h}_2\right) - (h_2)^{-1/2} \dot{h}_2 (h_1 - h_2)^{1/2}}{h_2},$$

where replacing $h_1, h_2, \dot{h}_1, \dot{h}_2$ by their values taken from (6.26), (6.23) and (6.24)–(6.25) gives the redundancy relation

$$\begin{aligned} &r(q_m, \dot{q}_m, \ddot{q}_m, u, q_L) \\ &\quad = \sqrt{h_2} (h_1 - h_2)^{1/2} \ddot{q}_m - \dot{h}_1 + \dot{h}_2 + (h_2)^{-1} \dot{h}_2 (h_1 - h_2) = 0 \end{aligned} \tag{6.27}$$

and the leakage detection rule

$$r(q_m, \dot{q}_m, \ddot{q}_m, u, 0) \neq 0 \Rightarrow q_L \neq 0. \square$$

6.3 Analytical Redundancy Relations for Linear Deterministic Systems - Time Domain

Let us consider the following continuous-time state-space model

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{E}_x \boldsymbol{d}(t) + \boldsymbol{F}_{\mathrm{x}} \boldsymbol{f}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) + \boldsymbol{E}_y \boldsymbol{d}(t) + \boldsymbol{F}_y \boldsymbol{f}(t), \end{aligned} \tag{6.28}$$

where $\boldsymbol{x} \in |\mathcal{R}^n$ denotes the state vector, $\boldsymbol{u} \in |\mathcal{R}^m$, is the vector of measured input signals, $\boldsymbol{y} \in |\mathcal{R}^p$ is the vector of measured plant output signals, $\boldsymbol{d} \in |\mathcal{R}^{n_{\mathrm{d}}}$ and $\boldsymbol{f} \in |\mathcal{R}^{n_f}$ are vectors of unknown input signals. $\boldsymbol{f}$ represents the faults one wishes to detect, while $\boldsymbol{d}$ are unknown disturbances that should not be detected.

The aim is to solve the following problem.

Problem 6.1 (*Design of linear analytical redundancy relations*) Given a model of the supervised process of the form (6.28), determine, if possible, a set of linear relations between the measured inputs and outputs and their derivatives up to a certain order, say q, such that,

- in the absence of fault,

$$\sum_{i=1}^{q} \boldsymbol{W}_{y,i}\, \boldsymbol{y}^{(i)}(t) + \sum_{i=1}^{q} \boldsymbol{W}_{u,i}\, \boldsymbol{u}^{(i)}(t) = 0,$$

where $z^{(i)}(t)$ denotes the ith derivative of $z(t)$ and $\boldsymbol{W}_{y,i}$, $\boldsymbol{W}_{u,i}$ are $n_r \times p$ and $n_r \times m$ matrices of real elements, n_r being the number of relations (to be determined),

- in the presence of a fault,

$$\sum_{i=1}^{q} \boldsymbol{W}_{y,i}\, \boldsymbol{y}^{(i)}(t) + \sum_{i=1}^{q} \boldsymbol{W}_{u,i}\, \boldsymbol{u}^{(i)}(t) \neq 0.$$

Such relations are a particular kind of analytical redundancy relations called parity relations.

In order to solve this problem, let us consider the successive time derivatives of $\boldsymbol{y}$ up to order q:

$$\begin{aligned} \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) + \boldsymbol{E}_y \boldsymbol{d}(t) + \boldsymbol{F}_y \boldsymbol{f}(t) \\ \dot{\boldsymbol{y}}(t) &= \boldsymbol{C}\dot{\boldsymbol{x}}(t) + \boldsymbol{D}\dot{\boldsymbol{u}}(t) + \boldsymbol{E}_y \dot{\boldsymbol{d}}(t) + \boldsymbol{F}_y \dot{\boldsymbol{f}}(t) \\ &= \boldsymbol{C}\boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{C}\boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{D}\dot{\boldsymbol{u}}(t) + \boldsymbol{C}\boldsymbol{E}_{\mathrm{x}}\, \boldsymbol{d}(t) \\ &\quad + \boldsymbol{E}_y\, \dot{\boldsymbol{d}}(t) + \boldsymbol{C}\boldsymbol{F}_{\mathrm{x}}\, \boldsymbol{f}(t) + \boldsymbol{F}_y\, \dot{\boldsymbol{f}}(t), \end{aligned} \tag{6.29}$$

where the last equality is deduced by substitution of (6.28) for $\dot{\boldsymbol{x}}(t)$. By iterating this process, the following expression for the qth derivative of $\boldsymbol{y}$ is obtained:

$$\begin{aligned} \boldsymbol{y}^{(q)}(t) = {} & \boldsymbol{C}\boldsymbol{A}^q \boldsymbol{x}(t) + \boldsymbol{C}\boldsymbol{A}^{(q-1)} \boldsymbol{B}\boldsymbol{u}(t) + \cdots + \boldsymbol{C}\boldsymbol{B}\boldsymbol{u}^{(q-1)}(t) + \boldsymbol{D}\boldsymbol{u}^{(q)}(t) + \\ & + \boldsymbol{C}\boldsymbol{A}^{(q-1)} \boldsymbol{E}_{\mathrm{x}} \boldsymbol{d}(t) + \cdots + \boldsymbol{C}\boldsymbol{E}_{\mathrm{x}}\, \boldsymbol{d}^{(q-1)}(t) + \boldsymbol{E}_y\, \boldsymbol{d}^{(q)}(t) + \\ & + \boldsymbol{C}\boldsymbol{A}^{(q-1)} \boldsymbol{F}_{\mathrm{x}} \boldsymbol{f}(t) + \cdots + \boldsymbol{C}\boldsymbol{F}_{\mathrm{x}}\, \boldsymbol{f}^{(q-1)}(t) + \boldsymbol{F}_y\, \boldsymbol{f}^{(q)}(t). \end{aligned} \tag{6.30}$$

The above set of equations can be concatenated into the expression

$$\bar{\boldsymbol{y}}^{(q)}(t) = \mathcal{O}\boldsymbol{x}(t) + \boldsymbol{T}_{u,q} \bar{\boldsymbol{u}}^{(q)}(t) + \boldsymbol{T}_{d,q} \bar{\boldsymbol{d}}^{(q)}(t) + \boldsymbol{T}_{f,q} \bar{\boldsymbol{f}}^{(q)}, \tag{6.31}$$

where $\bar{\boldsymbol{y}}^{(q)}(t) = \left(\boldsymbol{y}(t)^{\mathrm{T}} \dot{\boldsymbol{y}}(t)^{\mathrm{T}} \ldots \boldsymbol{y}^{(q)}(t)^{\mathrm{T}}\right)^{\mathrm{T}}$, and $\bar{\boldsymbol{u}}^{(q)}(t)$, $\bar{\boldsymbol{d}}^{(q)}(t)$, $\bar{\boldsymbol{f}}^{(q)}(t)$ have a similar form with $\boldsymbol{u}(t)$, $\boldsymbol{d}(t)$ and $\boldsymbol{f}(t)$ substituted for $\boldsymbol{y}(t)$,

$$\mathcal{O} = \begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{C}\boldsymbol{A} \\ \vdots \\ \boldsymbol{C}\boldsymbol{A}^q \end{pmatrix}, \qquad \boldsymbol{T}_{u,q} = \begin{pmatrix} \boldsymbol{D} & 0 & 0 & \ldots & 0 \\ \boldsymbol{C}\boldsymbol{B} & \boldsymbol{D} & 0 & \ldots & 0 \\ \vdots & \ddots & \ddots & & \\ \boldsymbol{C}\boldsymbol{A}^{q-1}\boldsymbol{B} & \ldots & \ldots & \boldsymbol{C}\boldsymbol{B} & \boldsymbol{D} \end{pmatrix}$$

and a similar definition holds for the block Toeplitz matrices $\boldsymbol{T}_{d,q}$, $\boldsymbol{T}_{f,q}$ with respectively $\boldsymbol{E}_y$ and $\boldsymbol{E}_x$ or $\boldsymbol{F}_y$ and $\boldsymbol{F}_x$ substituted for $\boldsymbol{D}$ and $\boldsymbol{B}$.

If there exists a value of q such that

$$\text{rank}\left(\mathcal{O}\ \ \boldsymbol{T}_{d,q}\right) < (q+1)p,$$

the left nullspace of $\left(\mathcal{O}\ \ \boldsymbol{T}_{d,q}\right)$ is not empty. The dimension of this subspace, say n_r, is given as $n_r = (q+1)p - \text{rank}\left(\mathcal{O}\ \ \boldsymbol{T}_{d,q}\right)$. Let $\boldsymbol{W}$ be a $n_r \times (q+1)p$ matrix of which each row is a basis vector for this subspace. Multiplying (6.31) on the left by $\boldsymbol{W}$ results in the following equality

$$\boldsymbol{W}\bar{\boldsymbol{y}}^{(q)}(t) - \boldsymbol{W}\boldsymbol{T}_{u,q}\bar{\boldsymbol{u}}^{(q)}(t) = \boldsymbol{W}\boldsymbol{T}_{f,q}\bar{\boldsymbol{f}}^{(q)}(t), \tag{6.32}$$

since $\boldsymbol{W}$ has been specifically computed to eliminate the terms in $\boldsymbol{x}(t)$ and $\bar{\boldsymbol{d}}^{(q)}(t)$. Equation (6.32) describes n_r analytical redundancy relations. Indeed, in the absence of fault, the right-hand side is equal to zero, and it is normally different from zero in the presence of a fault.

In order to implement such relations, and thus to compute the quantity

$$\boldsymbol{r}(t) = \boldsymbol{W}\bar{\boldsymbol{y}}^{(q)}(t) - \boldsymbol{W}\boldsymbol{T}_{u,q}\bar{\boldsymbol{u}}^{(q)}(t), \tag{6.33}$$

it is necessary to evaluate the derivatives that appear in the above relation. Such signals are highly sensitive to noise, so that filtered estimates of the derivatives have to be used. One approach is to resort to a state variable filter, which amounts to implementing the scheme of Fig. 6.2. Such a filter is used for each component of $\boldsymbol{y}(t)$ and $\boldsymbol{u}(t)$. Letting $z(t)$ denote the input of such a filter, the ith integrator output provides the ith filtered derivative of z, $z_f^{(i)}$. This filter corresponds to the analog simulation of the observability canonical state-space representation for the relation

$$z_f(s) = \frac{1}{s^q + a_1 s^{(q-1)} + \cdots + a_q} z(s).$$

By taking this filter into account, (6.33) can be rewritten in the frequency domain as

$$\boldsymbol{r}_f(s) = (\boldsymbol{W}_y(s)\boldsymbol{y}(s) + \boldsymbol{W}_u(s)\boldsymbol{u}(s))/p_f(s), \tag{6.34}$$

where

$$\boldsymbol{W}_y(s) = \sum_{i=0}^{q} \boldsymbol{W}_i s^i$$

with $\boldsymbol{W}_i$, the matrix made of columns $i\ p+1$ to $(i+1)p$ of $\boldsymbol{W}$, $\boldsymbol{W}_u(s)$ is defined similarly with $\boldsymbol{W}\boldsymbol{T}_{u,q}$ substituted for $\boldsymbol{W}$ and

$$p_f(s) = s^q + a_1 s^{(q-1)} + \cdots + a_q.$$

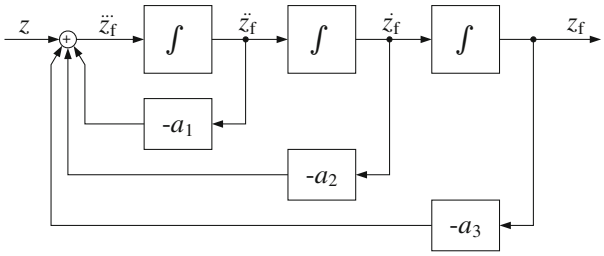


Fig. 6.2 Block diagram of a third-order state variable filter

Vector $\boldsymbol{r}$ is called a parity vector. It has generally different directions and magnitudes in the presence of the different fault modes. The n_r dimensional space of all such vectors is called the parity space, and any linear combination of the rows of (6.33) is called a parity relation.

The procedure for designing and implementing parity relations is now summarised.

Algorithm 6.1 *Parity relations for deterministic linear systems*

Given: A linear state-space model of the form (6.28) and a suitable order of derivation q

Compute off-line:
1. Matrices $\mathcal{O}$, $\boldsymbol{T}_{d,q}$, $\boldsymbol{T}_{u,q}$
2. A basis $\boldsymbol{W}$ for the left null space of $\begin{pmatrix}\mathcal{O} & \boldsymbol{T}_{d,q}\end{pmatrix}$
3. State space filters for the estimation of the derivatives of $\boldsymbol{y}$ and $\boldsymbol{u}$ up to order q.

At each time instant:
1. Acquire the new data $\boldsymbol{y}(t)$, $\boldsymbol{u}(t)$.
2. Compute $\boldsymbol{r}_f(t)$ from (6.34).

Result: A residual vector $\boldsymbol{r}_f(t)$ for an increasing time horizon.

An alternative approach to determine analytical redundancy relations can be deduced from the input-output model of the supervised process, namely in the frequency domain. It directly results in relations involving the filtered derivatives of the measured signals. By extension this method is called the (generalised) parity space approach. It is the object of the next section. Fault isolation can be handled in the linear case in a similar way as for the nonlinear case. The detailed treatment of this issue is deferred to Sect. 6.4.3.

Example 6.4 Parity relations for the ship
A linearised model of the ship example can be written as

$$\begin{pmatrix} \dot{\omega}_3 \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} b\eta_1 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \omega_3 \\ \psi \end{pmatrix} + \begin{pmatrix} b \\ 0 \end{pmatrix} \delta + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \omega_w \tag{6.35}$$

$$\begin{pmatrix} \omega_{3m} \\ \psi_{\mathrm{m}} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \omega_3 \\ \psi \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} f_\omega \\ f_\psi \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \omega_w, \tag{6.36}$$

when linearisation around $\omega_3 = 0$ is considered. Here δ, the rudder angle, is a known input, while ω_w, the wave disturbance, is an unknown input.

Straightforward computations yield the following expression for (6.31) with $q = 1$:

$$\begin{aligned} \begin{pmatrix} \omega_{3m} \\ \psi_{\mathrm{m}} \\ \dot{\omega}_{3m} \\ \dot{\psi}_{\mathrm{m}} \end{pmatrix} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ b\eta_1 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \omega_3 \\ \psi \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ b & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \delta \\ \dot{\delta} \end{pmatrix} \\ &\quad + \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \omega_w \\ \dot{\omega}_w \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f_\omega \\ f_\psi \\ \dot{f}_\omega \\ \dot{f}_\psi \end{pmatrix} \end{aligned} \tag{6.37}$$

The block matrix $\begin{pmatrix} \mathcal{O} & \boldsymbol{T}_{d,1} \end{pmatrix}$ takes the form:

$$\begin{pmatrix} \mathcal{O} & \boldsymbol{T}_{d,1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ b\eta_1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}.$$

A basis vector for the one-dimensional left nullspace of this matrix can be written

$$\boldsymbol{W} = \begin{pmatrix} 1 & 0 & 0 & -1 \end{pmatrix}.$$

Expression (6.32) then yields

$$\omega_{3m} - \dot{\psi}_{\mathrm{m}} = f_\omega - \dot{f}_\psi.$$

Hence a residual can be computed according as:

$$r_f(s) = (\omega_{3m}(s) - s\psi_{\mathrm{m}}(s))/(s + a), \tag{6.38}$$

where a is a design parameter to be adjusted according to the noise level. This expression is a particular case of the more general form (6.52) below for a residual for the ship example. Further discussion of the proposed residual is provided in Sect. 6.3. □

6.4 Analytical Redundancy Relations for Linear Deterministic Systems - Frequency Domain

In this section, the problems of fault detection, fault isolation and fault estimation are solved using the parity space approach to residual generation, from an input-output model of the supervised system. An alternative method would be to design observer-based residual generators, which yields similar filters, as indicated in the bibliographical notes. The observer-based approach will be used in the section on diagnosis systems design from a stochastic model, so that the reader will be acquainted with both methods.

6.4.1 Fault Detection

Consider again a system described by a linear continuous-time state-space model of the form

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{E}_x\,\boldsymbol{d}(t) + \boldsymbol{F}_{\mathrm{x}}\,\boldsymbol{f}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) + \boldsymbol{E}_y\boldsymbol{d}(t) + \boldsymbol{F}_y\boldsymbol{f}(t), \end{aligned} \tag{6.39}$$

where $\boldsymbol{x} \in |\mathcal{R}^n$ denotes the state vector, $\boldsymbol{u} \in |\mathcal{R}^m$, is the vector of measured input signals, $\boldsymbol{y} \in |\mathcal{R}^p$ is the vector of measured plant output signals, $\boldsymbol{d} \in |\mathcal{R}^{n_{\mathrm{d}}}$ and $\boldsymbol{f} \in |\mathcal{R}^{n_f}$ are vectors of unknown input signals. $\boldsymbol{f}$ represents the faults one wishes to detect, while $\boldsymbol{d}$ are unknown disturbances that should not be detected.

Such a model can also be written in terms of transfer functions:

$$\boldsymbol{y}(s) = \boldsymbol{H}_{\mathrm{yu}}(s)\boldsymbol{u}(s) + \boldsymbol{H}_{\mathrm{yd}}(s)\boldsymbol{d}(s) + \boldsymbol{H}_{\mathrm{yx}}(s)\boldsymbol{x}(0) + \boldsymbol{H}_{\mathrm{yf}}(s)\boldsymbol{f}(s), \tag{6.40}$$

where

$$\begin{aligned} \boldsymbol{H}_{\mathrm{yu}}(s) &= \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} + \boldsymbol{D} \\ \boldsymbol{H}_{\mathrm{yx}}(s) &= \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1} \\ \boldsymbol{H}_{\mathrm{yd}}(s) &= \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{E}_x + \boldsymbol{E}_y \\ \boldsymbol{H}_{\mathrm{yf}}(s) &= \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{F}_x + \boldsymbol{F}_y. \end{aligned}$$

As indicated in Fig. 6.1, a residual generator is a filter with input $\boldsymbol{u}$ and $\boldsymbol{y}$. As supervision of linear time-invariant systems is addressed here, the class of considered filters will be restricted to linear time-invariant systems of the following form

$$\begin{aligned} \dot{\boldsymbol{z}}(t) &= \boldsymbol{A}_z\boldsymbol{z}(t) + \boldsymbol{B}_{\mathrm{zu}}\,\boldsymbol{u}(t) + \boldsymbol{B}_{\mathrm{zy}}\,\boldsymbol{y}(t), \quad \boldsymbol{z}(0) = \boldsymbol{z}_0 \\ \boldsymbol{r}(t) &= \boldsymbol{C}_{\mathrm{rz}}\,\boldsymbol{z}(t) + \boldsymbol{D}_{\mathrm{ru}}\,\boldsymbol{u}(t) + \boldsymbol{D}_{\mathrm{ry}}\,\boldsymbol{y}(t) \end{aligned} \tag{6.41}$$

or, in transfer function form, assuming zero initial conditions:

$$\boldsymbol{r}(s) = \boldsymbol{V}_{\mathrm{ru}}(s)\boldsymbol{u}(s) + \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{y}(s) = \begin{pmatrix}\boldsymbol{V}_{\mathrm{ru}}(s) & \boldsymbol{V}_{\mathrm{ry}}(s)\end{pmatrix}\begin{pmatrix}\boldsymbol{u}(s)\\ \boldsymbol{y}(s)\end{pmatrix}. \tag{6.42}$$

The problem of residual generator design for fault detection based on a deterministic model can be stated as follows:

Problem 6.2 (*Residual generator design for fault detection based on a deterministic model*) Given, a model of the supervised process of the form (6.39) or (6.40) determine a stable linear time-invariant system (6.41) or (6.42) such that:

- In the absence of fault ($\boldsymbol{f}(t) = 0$ for all t), the output signal $\boldsymbol{r}(t), t > 0$ asymptotically decays to zero for any input $\boldsymbol{u}(t), \boldsymbol{d}(t), t > 0$ and any initial conditions $\boldsymbol{x}(0), \boldsymbol{z}(0)$.
- $\boldsymbol{r}(t)$ is affected by $\boldsymbol{f}(t)$.

The first condition assures that, after a transient due to the effect of initial conditions, the residual is almost equal to zero. The second condition is a fault detectability[3] condition. The output $\boldsymbol{r}(t)$ is affected by $\boldsymbol{f}(t)$ when the transfer matrix between $\boldsymbol{f}(s)$ and $\boldsymbol{r}(s)$ obtained by combining (6.40) and (6.42) is non-zero. A time domain definition of this notion is somewhat more cumbersome, hence, we defer it to Sect. 7.3.2.

Quite often, each component of the vector $\boldsymbol{f}(t)$ corresponds to a different fault. The detectability condition is then defined component-wise. One distinguishes the following notions:

Definition 6.1 (*Weak detectability*) The ith fault ($f_i(t) \neq 0$ for all $t \geq t_0$) is weakly detectable if there exists a stable residual generator such that $\boldsymbol{r}(t)$ is affected by $f_i(t)$.

In the literature, weak detectability is also referred to as detectability.

Definition 6.2 (*Strong detectability*) A fault f_i is strongly detectable if there exists a stable residual generator such that $\boldsymbol{r}(t)$ reaches a non-zero steady-state value for a fault signal that has a bounded final value different from zero.

6.4.2 Solution by the Parity Space Approach

In order to determine the conditions to be fulfilled by $\boldsymbol{V}_{\mathrm{ru}}(s)$ and $\boldsymbol{V}_{\mathrm{ry}}(s)$ for (6.42) to be a residual generator, (6.40) is substituted for $\boldsymbol{y}(s)$ in (6.42):

$$\begin{aligned}\boldsymbol{r}(s) &= \boldsymbol{V}_{\mathrm{ru}}(s)\boldsymbol{u}(s) + \boldsymbol{V}_{\mathrm{ry}}(s)\left(\boldsymbol{H}_{\mathrm{yu}}(s)\boldsymbol{u}(s) + \boldsymbol{H}_{\mathrm{yx}}(s)\boldsymbol{x}(0)\right.\\ &\quad \left. + \boldsymbol{H}_{\mathrm{yd}}(s)\boldsymbol{d}(s) + \boldsymbol{H}_{\mathrm{yf}}(s)\boldsymbol{f}(s)\right)\\ &= \begin{pmatrix}\boldsymbol{V}_{\mathrm{ru}}(s) + \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yu}}(s) & \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yd}}(s)\end{pmatrix}\begin{pmatrix}\boldsymbol{u}(s)\\ \boldsymbol{d}(s)\end{pmatrix}\\ &\quad + \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yx}}(s)\boldsymbol{x}(0) + \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yf}}(s)\boldsymbol{f}(s)\end{aligned} \tag{6.43}$$

[3] This notion should not be confused with the detectability of a linear system or a pair (C, A); indeed, the latter notion depends on the map from state to measured output, while the fault detectability is an input(i.e. fault)/output(i.e. residual) property.

Figure 6.3 illustrates this residual generator.

Fulfillment of the first condition in Problem 6.2 requires:

$$\begin{pmatrix} \boldsymbol{V}_{\mathrm{ru}}(s) + \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yu}}(s) & \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yd}}(s) \end{pmatrix} = \boldsymbol{O} \tag{6.44}$$

together with the asymptotic stability of $\boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yx}}(s)$. Since, in healthy working mode, the plant is normally stabilised by an appropriate controller, the latter condition amounts to requiring the stability of the filter. This can be guaranteed by an appropriate choice of the denominator of $\boldsymbol{V}_{\mathrm{ru}}(s)$ and $\boldsymbol{V}_{\mathrm{ry}}(s)$. Therefore, we concentrate now on the way to achieve (6.44). The question of fault detectability will be addressed once the class of all filters that fulfil (6.44) is characterised.

Note that (6.44) can be rewritten:

$$\begin{pmatrix} \boldsymbol{V}_{\mathrm{ry}}(s) & \boldsymbol{V}_{\mathrm{ru}}(s) \end{pmatrix} \begin{pmatrix} \boldsymbol{H}_{\mathrm{yu}}(s) & \boldsymbol{H}_{\mathrm{yd}}(s) \\ \boldsymbol{I} & \boldsymbol{O} \end{pmatrix} = 0. \tag{6.45}$$

For any filter, the least common multiple of the denominators of the entries of $\boldsymbol{V}_{\mathrm{ry}}(s)$ and $\boldsymbol{V}_{\mathrm{ru}}(s)$, $p(s)$ can be determined. Using $p(s)$, the left most matrix in (6.45) can be written:

$$\begin{pmatrix} \boldsymbol{V}_{\mathrm{ry}}(s) & \boldsymbol{V}_{\mathrm{ru}}(s) \end{pmatrix} = \frac{\begin{pmatrix} \bar{\boldsymbol{V}}_{\mathrm{ry}}(s) & \bar{\boldsymbol{V}}_{\mathrm{ru}}(s) \end{pmatrix}}{p(s)}, \tag{6.46}$$

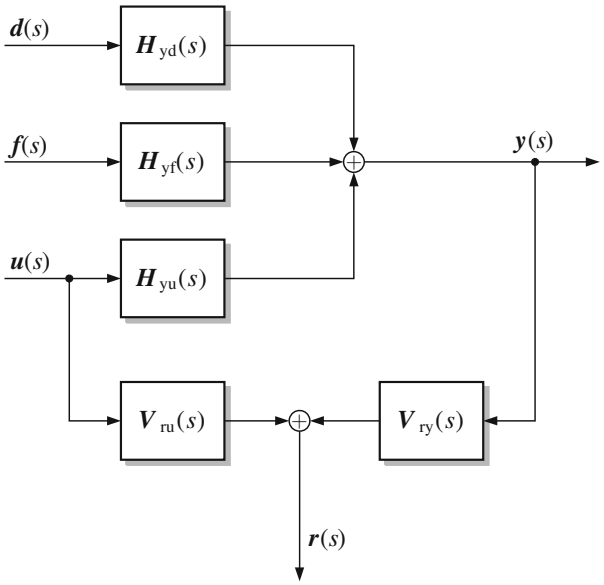


Fig. 6.3 Structure of residual generator in the parity space formulation

where $\bar{\boldsymbol{V}}_{\mathrm{ry}}(s)$ and $\bar{\boldsymbol{V}}_{\mathrm{ru}}(s)$ are suitable polynomial matrices. Hence, the whole class of filters that meet (6.45) can be obtained by characterising the set of polynomial matrices $\begin{pmatrix}\bar{\boldsymbol{V}}_{\mathrm{ry}}(s) & \bar{\boldsymbol{V}}_{\mathrm{ru}}(s)\end{pmatrix}$ that fulfil:

$$\begin{pmatrix}\bar{\boldsymbol{V}}_{\mathrm{ry}}(s) & \bar{\boldsymbol{V}}_{\mathrm{ru}}(s)\end{pmatrix}\begin{pmatrix}\boldsymbol{H}_{\mathrm{yu}}(s) & \boldsymbol{H}_{\mathrm{yd}}(s)\\ \boldsymbol{I} & \boldsymbol{O}\end{pmatrix} = 0. \tag{6.47}$$

This is the set of polynomial matrices that lie in the left nullspace of

$$\boldsymbol{H}(s) = \begin{pmatrix}\boldsymbol{H}_{\mathrm{yu}}(s) & \boldsymbol{H}_{\mathrm{yd}}(s)\\ \boldsymbol{I} & \boldsymbol{O}\end{pmatrix}. \tag{6.48}$$

This space is denoted $\mathcal{N}_L(\boldsymbol{H}(s))$. Its dimension is equal to the difference between the number of rows of $\boldsymbol{H}(s)$ and its rank, namely

$$\dim(\mathcal{N}_L(\boldsymbol{H}(s))) = m + p - \text{ rank } \boldsymbol{H}(s) = m + p - (m + n_{\mathrm{d}}) = p - n_{\mathrm{d}},$$

where m is the number of inputs, p the number of outputs, and n_{d} the number of unknown inputs (disturbances). It has been assumed that $\boldsymbol{H}_{\mathrm{yu}}(s)$ and $\boldsymbol{H}_{\mathrm{yd}}(s)$ have full column rank.[4] Note that the number of plant output signals must be larger than the number of disturbances for the left nullspace to be non-zero.

One way to characterise the set of polynomial matrices $\begin{pmatrix}\bar{\boldsymbol{V}}_{\mathrm{ry}}(s) & \bar{\boldsymbol{V}}_{\mathrm{ru}}(s)\end{pmatrix}$ that meet (6.47) is to determine an irreducible polynomial basis, for the rational vector space $\mathcal{N}_L(\boldsymbol{H}(s))$. Further, let $\boldsymbol{F}(s)$ be a matrix of which the rows make such an irreducible polynomial basis, then any suitable matrix $\begin{pmatrix}\bar{\boldsymbol{V}}_{\mathrm{ry}}(s) & \bar{\boldsymbol{V}}_{\mathrm{ru}}(s)\end{pmatrix}$ can be obtained by combinations of the rows of $\boldsymbol{F}(s)$, namely

$$\begin{pmatrix}\bar{\boldsymbol{V}}_{\mathrm{ry}}(s) & \bar{\boldsymbol{V}}_{\mathrm{ru}}(s)\end{pmatrix} = \boldsymbol{Q}(s)\boldsymbol{F}(s), \tag{6.49}$$

where $\boldsymbol{Q}(s)$ is an arbitrary polynomial matrix with appropriate number of columns.

A general parametrisation of the family of residual generators is obtained from (6.49). Substitution of (6.49) into (6.46) yields

$$\begin{pmatrix}\boldsymbol{V}_{\mathrm{ry}}(s) & \boldsymbol{V}_{\mathrm{ru}}(s)\end{pmatrix} = \frac{\boldsymbol{Q}(s)\boldsymbol{F}(s)}{p(s)}.$$

Introducing this expression into (6.42) finally results in

$$r(s) = \frac{\boldsymbol{Q}(s)\boldsymbol{F}(s)}{p(s)}\begin{pmatrix}\boldsymbol{y}(s)\\ \boldsymbol{u}(s)\end{pmatrix}. \tag{6.50}$$

[4] The notion of rank considered here is the normal rank computed as $\max_s$ rank $H(s)$ where the maximum is taken over all complex values of s.

The choice of the matrix $\boldsymbol{Q}(s)$ and the polynomial $p(s)$ depends on the specification of the diagnosis problem. Typically, the residual generator should ensure filtering of high frequency disturbances which always exist, even though they were not considered in the model, and adequate properties at low frequencies. Sometimes, precise information on the frequency range of the fault is available, and $\boldsymbol{Q}(s)/p(s)$ can be designed to perform appropriate filtering.

Remark 6.4 (*Link with parity relations deduced from the state-space model*) Equation (6.34) clearly has the same form as (6.42) and, by construction, it fulfils the first condition of Problem 6.2 provided $p_f(s)$ has all its roots in the open left-half plane. Hence, there exist a matrix $\boldsymbol{Q}(s)$ and a polynomial $p(s)$ for which (6.34) and (6.50) are identical. □

Modelling uncertainty. Although modelling uncertainties have not been introduced here, they can be accounted for a posteriori when $\boldsymbol{F}(s)$ has several rows. $\boldsymbol{Q}(s)$ is then used to select appropriate rows in $\boldsymbol{F}(s)$. To explain the idea, let $\boldsymbol{F}_i(s)$, $i = 1, \ldots, n_r$ denote the ith row of $\boldsymbol{F}(s)$ and consider the scalar residuals

$$r_i(s) = \frac{\boldsymbol{F}_i(s)}{p(s)} \begin{pmatrix} \boldsymbol{y}(s) \\ \boldsymbol{u}(s) \end{pmatrix} \qquad i = 1, \ldots, n_r.$$

By performing a simulation of all these filters with actual plant measurements as input, one may compare how significantly the actual residuals $r_i(t)$, $i = 1, \ldots, n_r$ deviate from zero in the absence of fault, once the transient due to initial conditions has vanished. This reflects the effect of modelling errors on the residuals. Besides, by using faulty data obtained with a simulation or corresponding to actual plant measurements it is also possible to compare the actual sensitivities to faults. A kind of "signal to noise ratio" could be defined for each residual as

$$SNR_i = \frac{\int_{t_0}^{t_0+T} r_i^{F\,2}(t)\mathrm{d}t}{\int_{t_1}^{t_1+T} r_i^{FF\,2}(t)\mathrm{d}t}, \tag{6.51}$$

where $r_i^F(t)$ denotes the residual obtained with the measurement associated to the faulty mode, and r_i^{FF} corresponds to the fault free situation. T is a user defined horizon, t_0 and t_1 are time instants associated to faulty and fault free data sequences. Matrix $\boldsymbol{Q}(s)$ should then be chosen to select the components of $\boldsymbol{r}(s)$ for which the "signal-to-noise ratio" is significantly larger than 1.

Computational aspects. The problem of finding an irreducible polynomial basis for $\mathcal{N}_L(\boldsymbol{H}(s))$ can be transformed into the determination of a similar basis for a polynomial matrix instead of the rational matrix $\boldsymbol{H}(s)$. It suffices to notice that

$$\boldsymbol{H}(s) = \bar{\boldsymbol{H}}(s)/h(s),$$

where $h(s)$ is the least common multiple of all denominators. An irreducible polynomial basis for $\bar{\boldsymbol{H}}(s)$ is also an irreducible polynomial basis for $\boldsymbol{H}(s)$, and vice-versa. Numerically stable algorithms for the computation of an irreducible polynomial basis are available in the literature, and they have been programmed in the polynomial toolbox of MATLAB.

The symbolic tools Maple and Mathematica can calculate a basis for the left nullspace of $\boldsymbol{H}(s)$. The Maple command *nullspace basis* applied to the matrix $\boldsymbol{H}^{\mathrm{T}}(s)$ will provide the row basis given in analytical form. Calculation of a basis is not unique, so the result can be expanded or reduced by a polynomial fraction as desired. The result is not necessarily irreducible, either, but the reduction to an irreducible basis is usually straightforward once a factorisation is made of the entries in the nullspace basis.[5]

Example 6.4 (cont.) Parity relations for the ship
A model of the form (6.40) can be easily deduced from the linear state-space model for the ship example. The following transfer matrices are obtained when sensor faults are considered, and when state and sensor noise are neglected:

$$\boldsymbol{H}_{\mathrm{yu}}(s) = \begin{pmatrix} \frac{b}{s-b\eta_1} \\ \frac{b}{(s-b\eta_1)s} \end{pmatrix}, \quad \boldsymbol{H}_{\mathrm{yd}}(s) = \begin{pmatrix} 1 \\ \frac{1}{s} \end{pmatrix}, \quad \boldsymbol{H}_{\mathrm{yx}}(s) = \begin{pmatrix} \frac{1}{s-b\eta_1} & 0 \\ \frac{1}{s(s-b\eta_1)} & \frac{1}{s} \end{pmatrix}$$

and $\boldsymbol{H}_{\mathrm{yf}} = \boldsymbol{I}_2$. In the above expressions,

$$\begin{aligned} \boldsymbol{x}(t) &= (\omega_3(t) \quad \psi(t))^{\mathrm{T}} \\ \boldsymbol{y}(t) &= (\omega_{3m}(t) \quad \psi_{\mathrm{m}}(t))^{\mathrm{T}} \\ d(t) &= \omega_w(t) \\ u(t) &= \delta(t) \\ \boldsymbol{f}(t) &= \begin{pmatrix} f_\omega(t) & f_\psi(t) \end{pmatrix}^{\mathrm{T}} \end{aligned}$$

hold. It is assumed that η_1 is negative, so that the ship is stable. $\boldsymbol{H}_{\mathrm{yx}}(s)$ is not asymptotically stable however, due to the integrator linking speed and position. We shall see below what slight modification must be introduced in the theory to handle the pole at the origin.

The matrix $\boldsymbol{H}(s)$ takes the form:

$$\boldsymbol{H}(s) = \begin{pmatrix} \frac{b}{s-b\eta_1} & 1 \\ \frac{b}{s(s-b\eta_1)} & \frac{1}{s} \\ 1 & 0 \end{pmatrix} = \frac{1}{s(s-b\eta_1)} \begin{pmatrix} bs & s(s-b\eta_1) \\ b & (s-b\eta_1) \\ s(s-b\eta_1) & 0 \end{pmatrix}.$$

[5]The Maple symbolic mathematics engine is a stand-alone product. It is also a part of the MATLAB Symbolic Toolbox. MATLAB®, Maple® and Mathematica® are registered trademarks of their respective owners.

The last matrix corresponds to $\bar{\boldsymbol{H}}(s)$. An irreducible basis for its left nullspace can be calculated, or found by inspection, to be

$$F(s) = (1 \quad -s \quad 0).$$

Thus, any vector of rational functions of the form

$$\left(\frac{q(s)}{p(s)} \quad \frac{-sq(s)}{p(s)} \quad 0 \right),$$

where $p(s)$ is an arbitrary polynomial with roots in the left-half plane and $q(s)$ is an arbitrary polynomial with degree less than $p(s)$, fulfils condition (6.45). Candidate residual generators have the form:

$$r(s) = \frac{q(s)}{p(s)}\omega_{3m}(s) - \frac{sq(s)}{p(s)}\psi_{\mathrm{m}}(s). \tag{6.52}$$

Note that, by setting $q(s) = 1$, one recovers (6.38) with $p(s) = s + a$.

Substituting the model equations for $\omega_3(s)$ and $\psi(s)$ yields

$$\begin{aligned} r(s) = & \frac{q(s)}{p(s)} \left(\frac{b}{s - b\eta_1}\delta(s) + \omega_w(s) + \frac{1}{s - b\eta_1}\omega_3(0) + f_\omega(s) \right) \\ & - \frac{sq(s)}{p(s)} \left(\frac{b}{s(s - b\eta_1)}\delta(s) + \frac{\omega_w(s)}{s} + \frac{1}{s(s - b\eta_1)}\omega_3(0) + \frac{1}{s}\psi(0) + f_\psi(s) \right) \\ = & -\frac{q(s)}{p(s)}\psi(0) + \frac{q(s)}{p(s)}f_\omega(s) - \frac{sq(s)}{p(s)}f_\psi(s). \end{aligned}$$

In order to assure that the residual asymptotically vanishes, two solutions are possible:

- Introduction of a derivative action in $q(s)$, so that $q(s) = s\bar{q}(s)$ and the term involving $\psi(0)$ in the above equation is null at steady state.
- Modification of (6.52) by adding a correction term associated with the initial position (supposed to be measured correctly). This yields

$$r(s) = \frac{q(s)}{p(s)}\psi(0) + \frac{q(s)}{p(s)}\omega_{3m}(s) - \frac{sq(s)}{p(s)}\psi_{\mathrm{m}}(s)$$

or, after substitution of $\omega_{3m}(s)$ and $\psi_{\mathrm{m}}(s)$ in terms of the model equations:

$$r(s) = \frac{q(s)}{p(s)}f_\omega(s) - \frac{sq(s)}{p(s)}f_\psi(s). \tag{6.53}$$

The first solution also introduces a derivative action in the transfer functions between $f_\omega(s)$ and $r(s)$, and between $f_\psi(s)$ and $r(s)$. Hence step like faults do not have any steady-state effect on the residual. On the other hand, in (6.53), $q(s)$ can be chosen so that a step-like fault f_ω has a steady-state effect on r, but a step-like fault in f_ψ can only influence temporarily r due to the zero at the origin in $\frac{sq(s)}{p(s)}$. Application of the theory below will indicate that, indeed, fault f_ω is strongly detectable, but f_ψ is only weakly detectable. □

Example 6.5 Parity relations - ship with three output measurements
Some useful observations can be made later from the above example but using an additional instrument to measure the ship heading. This third instrument is taken to be independent of the other two. This is a realistic case since redundant heading instruments are required for most merchant ships.

With two independent heading angle measurements

$$y_2(s) = \psi_{\mathrm{m}}^{(1)}(s)$$

and

$$y_3(s) = \psi_{\mathrm{m}}^{(2)}(s),$$

the matrix $\boldsymbol{H}(s)$ takes the form:

$$\boldsymbol{H}(s) = \frac{1}{s(s-b\eta_1)} \begin{pmatrix} bs & s(s-b\eta_1) \\ b & (s-b\eta_1) \\ b & (s-b\eta_1) \\ s(s-b\eta_1) & 0 \end{pmatrix}.$$

The left nullspace basis for $\boldsymbol{H}(s)$ is computed to be

$$\begin{pmatrix} \frac{-1}{s} & 1 & 0 & 0 \\ \frac{-1}{s} & 0 & 1 & 0 \end{pmatrix}.$$

This means a family of candidate residual generators exist, which have the form

$$\boldsymbol{r}(s) = \frac{q(s)}{p(s)} \begin{pmatrix} \frac{-1}{s} & 1 & 0 & 0 \\ \frac{-1}{s} & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \omega_{3m}(s) \\ \psi_m^{(1)}(s) \\ \psi_m^{(2)}(\delta) \\ \delta(s) \end{pmatrix}.$$

The relation between components of the residual vector $\boldsymbol{r}(s)$ to faults $\boldsymbol{f}(s)$ and wave disturbance ω_w is

$$\begin{aligned} r_1(s) &= \frac{q(s)}{p(s)} \left(-\frac{1}{s} f_\omega(s) + f_\psi^{(1)}(s) \right) \\ r_2(s) &= \frac{q(s)}{p(s)} \left(-\frac{1}{s} f_\omega(s) + f_\psi^{(2)}(s) \right). \end{aligned}$$

It is evident that all elements of the residual are decoupled from the wave disturbance, which was the intention.

Forming a third residual using the plain difference between heading angle measurements

$$r_3(s) = \psi_{\mathrm{m}}^{(1)}(s) - \psi_{\mathrm{m}}^{(2)}(s),$$

which would be a straightforward choice as an output parity equation, is indeed possible, but since this would be a linear relation between the two residuals already defined, this would not add to the information contained in the residual vector. □

Fault detectability. To deduce theoretical results on fault detectability, the expression of the residual in the presence of faults must be determined. Substituting (6.45) into (6.43) yields

$$\begin{aligned} \boldsymbol{r}(s) &= \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yx}}(s)\boldsymbol{x}(0) + \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yf}}(s)\boldsymbol{f}(s) \\ &= \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yx}}(s)\boldsymbol{x}(0) + \sum_{i=1}^{n_f} \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}^i_{\mathrm{yf}}(s) f_i(s), \end{aligned} \tag{6.54}$$

where $\boldsymbol{H}^i_{\mathrm{yf}}(s)$ denotes the ith column of $\boldsymbol{H}_{\mathrm{yf}}(s)$. It can be shown that a necessary and sufficient condition for detectability of the ith fault is:

$$\boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}^i_{\mathrm{yf}}(s) \neq 0, \tag{6.55}$$

where $\boldsymbol{V}_{\mathrm{ry}}(s)$ also fulfils

$$\boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yd}}(s) = 0. \tag{6.56}$$

The latter condition comes from the second entry in (6.44). For (6.55) and (6.56) to be simultaneously verified, one should not be able to express $\boldsymbol{H}^i_{\mathrm{yf}}(s)$ as a linear combination of the columns of $\boldsymbol{H}_{\mathrm{yd}}(s)$. In other words, there cannot exists any non-zero polynomial set $\alpha_0(s), \alpha_1(s), \ldots, \alpha_{n_{\mathrm{d}}}(s)$ such that:

$$\alpha_0(s)\boldsymbol{H}^i_{\mathrm{yf}}(s) + \alpha_1(s)\boldsymbol{H}^1_{\mathrm{yd}}(s) + \cdots + \alpha_{n_{\mathrm{d}}}(s)\boldsymbol{H}^{n_{\mathrm{d}}}_{\mathrm{yd}}(s) = \boldsymbol{O}$$

This condition is fulfilled when

$$\operatorname{rank} \begin{pmatrix} \boldsymbol{H}_{\mathrm{yd}}(s) & \boldsymbol{H}^i_{\mathrm{yf}}(s) \end{pmatrix} > \operatorname{rank} \boldsymbol{H}_{\mathrm{yd}}(s), \tag{6.57}$$

where

$$\operatorname{rank} \boldsymbol{A}(s) = \max_s \operatorname{rank} \boldsymbol{A}(s)$$

denotes the normal rank of the rational matrix $\boldsymbol{A}(s)$. In the latter expression, the rank-operation in the right-hand side acts on a matrix of complex numbers obtained for a specific value of s. It can thus be evaluated in the standard way. Equation (6.57) is a necessary and sufficient condition for the ith fault to be weakly detectable.

To determine a test for strong fault detectability, substitute the model (6.40) for $\boldsymbol{y}(s)$ in the parametrisation of the class of residual generators (6.50)

$$\begin{aligned} \boldsymbol{r}(s) &= \frac{\boldsymbol{Q}(s)\boldsymbol{F}(s)}{p(s)} \begin{pmatrix} \boldsymbol{H}_{\mathrm{yu}}(s) & \boldsymbol{H}_{\mathrm{yd}}(s) & \boldsymbol{H}_{\mathrm{yf}}(s) \\ \boldsymbol{I} & \boldsymbol{O} & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}(s) \\ \boldsymbol{d}(s) \\ \boldsymbol{f}(s) \end{pmatrix} \\ &= \frac{\boldsymbol{Q}(s)\boldsymbol{F}(s)}{p(s)} \begin{pmatrix} \boldsymbol{H}_{\mathrm{yf}}(s) \\ \boldsymbol{O} \end{pmatrix} \boldsymbol{f}(s), \end{aligned} \tag{6.58}$$

where the second equality accounts for the fact that $\boldsymbol{F}(s)$ is a basis for the left nullspace of $\boldsymbol{H}(s)$. The transient term due to $\boldsymbol{x}(0)$ was not considered as its effect vanishes when t tends to infinity. Strong detectability of the fault f_i is thus achieved if there exists some polynomial $p(s)$ and polynomial matrix $\boldsymbol{Q}(s)$ such that

$$\frac{\boldsymbol{Q}(s)\boldsymbol{F}(s)}{p(s)} \begin{pmatrix} \boldsymbol{H}^i_{\mathrm{yf}}(s) \\ \boldsymbol{O} \end{pmatrix} \Bigg|_{s=0} \neq 0. \tag{6.59}$$

As $p(0)$ is necessarily chosen non-zero to assure asymptotic stability of the filter, and $\boldsymbol{Q}(s)$ can be chosen arbitrarily, a necessary and sufficient condition for strong fault detectability is

$$\boldsymbol{F}(s) \begin{pmatrix} \boldsymbol{H}^i_{\mathrm{yf}}(s) \\ \boldsymbol{O} \end{pmatrix} \Bigg|_{s=0} \neq 0. \tag{6.60}$$

Note that this expression may be different from $\boldsymbol{F}(0) \begin{pmatrix} \boldsymbol{H}^i_{\mathrm{yf}}(0) \\ \boldsymbol{O} \end{pmatrix}^{\mathrm{T}}$ and, hence, substitution by $s = 0$ must be performed after computation of the matrix product.

Example 6.6 Detectability - ship with two output measurements
To check that fault f_ω is detectable, (6.57) is applied as follows

$$\operatorname{rank} \begin{pmatrix} 1 & 1 \\ \frac{1}{s} & 0 \end{pmatrix} > \operatorname{rank} \begin{pmatrix} 1 \\ \frac{1}{s} \end{pmatrix}.$$

Similarly, the inequality

$$\operatorname{rank} \begin{pmatrix} 1 & 0 \\ \frac{1}{s} & 1 \end{pmatrix} > \operatorname{rank} \begin{pmatrix} 1 \\ \frac{1}{s} \end{pmatrix}$$

ensures that f_ψ is detectable. Condition (6.60) is now used to check strong fault detectability. For fault f_ω it yields

$$\begin{pmatrix} 1 & -s & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \Bigg|_{s=0} = 1.$$

Thus fault f_ω is strongly detectable. For fault f_ψ one gets

$$\begin{pmatrix} 1 & -s & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \Bigg|_{s=0} = 0,$$

which indicates that f_ψ is not strongly detectable, as was expected. □

The procedure for residual generator design can be summarised as follows.

Algorithm 6.2 *Residual generator design with the parity space method*

Given: A model of the supervised system in the form (6.40).

Computation:

1. Compute matrix $\boldsymbol{H}(s)$ as defined by (6.48).
2. Determine an irreducible polynomial basis for $\mathcal{N}_L(\boldsymbol{H}(s))$, and let $\boldsymbol{F}(s)$ be the matrix whose rows make such a basis. If $\boldsymbol{F}(s) = \boldsymbol{O}$, the problem has no solution.
3. Design the filter $\frac{\boldsymbol{Q}(s)}{p(s)}$ as a low-pass or a band-pass filter which possibly selects appropriate rows in $\boldsymbol{F}(s)$ according to SNR_i, $(i = 1, \ldots, \beta)$ in Eq. (6.51).
4. Check for weak or strong fault detectability as needed.

Result: A residual generator in the form (6.50).

6.4.3 Fault Isolation

For fault-tolerant control, faults should not only be detected, but also be isolated, namely the faulty components should be determined. The problem of residual generator design for fault detection and isolation based on a deterministic model can be stated as follows.

Consider, a system described by a continuous-time linear state-space model of the form

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \sum_{j=1}^{n_f} \boldsymbol{F}_{\mathrm{x}}^{j} \boldsymbol{f}_j(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) + \sum_{j=1}^{n_f} \boldsymbol{F}_{\mathrm{y}}^{j} \boldsymbol{f}_j(t), \end{aligned} \tag{6.61}$$

where $\boldsymbol{f}_j \in |\mathcal{R}^{n_{f_j}}$, $j = 1, \ldots, n_f$ represent the faults that must be detected and isolated. In terms of transfer functions, (6.61) can be written as

$$\boldsymbol{y}(s) = \boldsymbol{H}_{\mathrm{yu}}(s)\boldsymbol{u}(s) + \boldsymbol{H}_{\mathrm{yx}}(s)\boldsymbol{x}(0) + \sum_{j=1}^{n_f} \boldsymbol{H}_{yf_j}(s)\boldsymbol{f}_j(s), \tag{6.62}$$

where

Table 6.1 Effects of the faults on the residuals

↗	$\boldsymbol{f}_1$	$\boldsymbol{f}_2$	$\boldsymbol{f}_3$
$\boldsymbol{r}_1$	×	0	0
$\boldsymbol{r}_2$	0	×	0
$\boldsymbol{r}_3$	0	0	×

$$\boldsymbol{H}_{\mathrm{yu}}(s) = \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} + \boldsymbol{D},\ \boldsymbol{H}_{\mathrm{yx}}(s) = \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}$$

and

$$\boldsymbol{H}_{yf_j}(s) = \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{F}_{\mathrm{x}}^{j} + \boldsymbol{F}_{\mathrm{y}}^{j}.$$

Problem 6.3 (*Residual generator design for fault detection and isolation based on a deterministic model*) Given a model of the supervised process of the form (6.61) or (6.62), determine a set of n_f stable linear time-invariant filters described by

$$\begin{aligned} \dot{\boldsymbol{z}}_\ell(t) &= \boldsymbol{A}_{\mathrm{z},\ell}\boldsymbol{z}_\ell(t) + \boldsymbol{B}_{\mathrm{zu},\ell}\boldsymbol{u}(t) + \boldsymbol{B}_{\mathrm{zy},\ell}\boldsymbol{y}(t), \quad \boldsymbol{z}_\ell(0) = \boldsymbol{z}_{0,\ell} \\ \boldsymbol{r}_\ell(t) &= \boldsymbol{C}_{\mathrm{rz},\ell}\boldsymbol{z}_\ell(t) + \boldsymbol{D}_{\mathrm{ru},\ell}\boldsymbol{u}(t) + \boldsymbol{D}_{\mathrm{ry},\ell}\boldsymbol{y}(t), \quad \ell = 1, \ldots, n_f \end{aligned} \tag{6.63}$$

or, in transfer function form, assuming zero initial conditions,

$$\boldsymbol{r}_\ell(s) = \boldsymbol{V}_{\mathrm{ru},\ell}(s)\boldsymbol{u}(s) + \boldsymbol{V}_{\mathrm{ry},\ell}(s)\boldsymbol{y}(s), \qquad \ell = 1, \ldots, n_f, \tag{6.64}$$

such that the following conditions are met.

- $\boldsymbol{r}_\ell(t)$ asymptotically decays to zero for any $\boldsymbol{u}(t)$ and any $\boldsymbol{f}_j(t),\ j = 1, \ldots, n_f,\ j \neq \ell,\ t > 0$.
- $r_\ell(t)$ is affected by $f_\ell(t)$.

In this problem statement, the ℓth residual can only be affected by the ℓth fault, and not by the others. The table below represents this situation when $n_f = 3$. A symbol × in Table 6.1 indicates that the fault in the corresponding column affects the residual of the corresponding row.

The faults that do not affect the ℓth residual can be seen as unknown inputs to which this residual should not be sensitive. Hence, to design a residual generator that output r_ℓ, it suffices to use the solution of the problem of residual generation for fault detection in which vector $\boldsymbol{d}$ is replaced by $(\boldsymbol{f}_1^{\mathrm{T}} \ldots \boldsymbol{f}_{\ell-1}^{\mathrm{T}}\ \boldsymbol{f}_{\ell+1}^{\mathrm{T}} \ldots \boldsymbol{f}_{n_f}^{\mathrm{T}})^{\mathrm{T}}$. n_f such problems should be solved for $\ell = 1, \ldots, n_f$ in order to obtain the n_f filters that make a solution to the fault isolation problem.

From the conditions for fault detectability, the following conditions can be deduced for the above scheme to work:

$$\begin{aligned} &\text{rank}\ \left(\boldsymbol{H}_{\mathrm{y},f_\ell}(s)\ \ \boldsymbol{H}_{\mathrm{y},f_j}(s)\right) > \text{rank}\ \boldsymbol{H}_{\mathrm{y},f_j}(s) \\ &\text{for all } \ell,\ j = 1, \ldots, n_f,\ \ell \neq j. \end{aligned} \tag{6.65}$$

Table 6.2 Effects of the faults on the residuals—non-diagonal structure

↗	$\boldsymbol{f}_1$	$\boldsymbol{f}_2$	$\boldsymbol{f}_3$
$\boldsymbol{r}_1$	×	×	0
$\boldsymbol{r}_2$	×	0	×

A necessary condition for (6.65) to hold is

$$\sum_{\substack{j\ =\ 1, n_f \\ j\ \neq\ \ell}} n_{f_j} < p, \tag{6.66}$$

where p is the number of measured output signals (dimension of $\boldsymbol{y}$).

When condition, (6.65) is not met, the diagonal structure of Table 6.1 cannot be obtained, and one should attempt to group the fault vectors in different classes and to generate residuals that are affected by a specific fault class and not by the others. The table below illustrates one way to perform such a grouping, in a situation where $n_f = 3$ and two residual generators are designed.

In the situation of Table 6.2, all three faults can be distinguished as the combination of r_1 and r_2 reacts differently to each fault. However, simultaneous faults cannot be isolated because they affect both residuals in all cases.

Example 6.7 Isolability - ship with three output measurements
For the ship with one rate measurement and two heading measurements (Example 6.5), a residual generator is achieved, which was decoupled from the disturbance,

$$\begin{pmatrix} r_1(s) \\ r_2(s) \end{pmatrix} = \begin{pmatrix} -\frac{1}{s} & 1 & 0 \\ -\frac{1}{s} & 0 & 1 \end{pmatrix} \begin{pmatrix} f_{\omega 3}(s) \\ f_{\psi}^{(1)}(s) \\ f_{\psi}^{(2)}(s) \end{pmatrix}. \tag{6.67}$$

This residual generator has the properties shown in Table 6.2. □

Sensor fault isolation in a fault-tolerant control setting. If it has been detected that one out of a set of faults is present, but it has not been possible to isolate which fault is actually present, and this was due to the design of the residual generator specification, alternatives are available on the fault-tolerant setting because the supervisory system has control of the input signals to the plant. Similar to system identification, where a dedicated test signal is applied to obtain the optimal information about a particular parameter, a dedicated test signal can be applied on the control input to help confirm particular hypotheses. This procedure can help to reduce the time to diagnose and, hence, the time to reconfigure a controller.

Example 6.8 Dedicated test signal for isolation - ship steering
If two identical rate sensors are available in the ship steering example, and the residual generator was designed to be insensitive to the wave disturbance, it is not possible to isolate faults

f_ω^1 and f_ω^2. In a fault-tolerant control setting, we employ active test signal generation to isolate the fault once it has been detected that one of the rate sensor units is defect. Let us define a dedicated test signal

$$\delta(t) = \tilde{\delta}(t),\ t \subset [0,\ T],$$

which is applied immediately after the hypothesis of

$$\left\{ \hat{f}_\omega^1(t) \vee \hat{f}_\omega^2(t) \right\} \neq 0$$

is confirmed. Observe a-priori the response in the non-faulty condition

$$\omega_3^{rec}(t) = g_{\omega_3}\left(\tilde{\delta}(t),\ U(t)\right),\ t \subset [0, T]$$

note that the function g_{ω_3} is not calculated, the angular rate is merely recorded and stored. Calculate the correlations

$$\begin{aligned} cor_{31}(t) &= \frac{1}{t} \int_0^t \omega_3^{rec}(\tau)\omega_{3m}^1(\tau)\mathrm{d}\tau \\ cor_{21}(t) &= \frac{1}{t} \int_0^t \omega_{3m}^2(\tau)\omega_{3m}^1(\tau)\mathrm{d}\tau \\ cor_{32}(t) &= \frac{1}{t} \int_0^t \omega_3^{rec}(\tau)\omega_{3m}^2(\tau)\mathrm{d}\tau. \end{aligned}$$

These correlation signals with appropriate normalisation make it straightforward to determine which hypothesis is the most likely. □

6.4.4 Fault Estimation

The isolation schemes signify which fault is present but do not assess the magnitude of the fault. Fault estimates are needed in certain fault accommodation approaches as was indicated in Sect. 6.1. This notion is defined as follows.

Definition 6.3 (*Fault estimation*) Fault estimation is the ability to estimate the magnitude of a fault $\boldsymbol{f}_i(t)$ and its time history.

Combining (6.42) and (6.58), the link between the fault vector $\boldsymbol{f}(s)$ and the residual $\boldsymbol{r}(s)$ is seen to be

$$\boldsymbol{r}(s) = \boldsymbol{V}_{\mathrm{ru}}(s)\boldsymbol{u}(s) + \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{y}(s) = \frac{\boldsymbol{Q}(s)\boldsymbol{F}(s)}{p(s)} \begin{pmatrix} \boldsymbol{H}_{\mathrm{yf}}(s) \\ \boldsymbol{O} \end{pmatrix} \boldsymbol{f}(s), \tag{6.68}$$

where it is assumed that initial conditions have vanished. Letting

$$\boldsymbol{F}(s) = (\boldsymbol{F}_1(s) \quad \boldsymbol{F}_2(s)),$$

where $\boldsymbol{F}_1(s)$ has p columns and $\boldsymbol{F}_2(s)$ has m columns, Eq. (6.68) can be written

$$\boldsymbol{r}(s) = \boldsymbol{V}_{\mathrm{ru}}(s)\boldsymbol{u}(s) + \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{y}(s) = \frac{\boldsymbol{Q}(s)\boldsymbol{F}_1(s)}{p(s)}\boldsymbol{H}_{\mathrm{yf}}(s)\boldsymbol{f}(s). \tag{6.69}$$

On the other hand, Eq. (6.54) yields the following relation when the transient due to the initial conditions is neglected

$$\boldsymbol{r}(s) = \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yf}}(s)\boldsymbol{f}(s),$$

hence

$$\boldsymbol{V}_{\mathrm{ry}}(s) := \frac{\boldsymbol{Q}(s)\boldsymbol{F}_1(s)}{p(s)}. \tag{6.70}$$

As a compact notation, introduce $\boldsymbol{H}_{rf}(s)$ by

$$\boldsymbol{H}_{rf}(s) := \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yf}}(s) = \frac{\boldsymbol{Q}(s)\boldsymbol{F}_1(s)}{p(s)}\boldsymbol{H}_{\mathrm{yf}}(s).$$

If it is possible to determine a suitable left inverse to $\boldsymbol{H}_{rf}(s)$, say $\boldsymbol{G}(s)$, an estimate of $\boldsymbol{f}(s)$ would be

$$\hat{\boldsymbol{f}}(s) = \boldsymbol{G}(s)\boldsymbol{r}(s) = \boldsymbol{G}(s)(\boldsymbol{V}_{\mathrm{ru}}(s)\boldsymbol{u}(s) + \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{y}(s)). \tag{6.71}$$

Left inverse transformation. If the polynomial matrix $\boldsymbol{H}_{rf}(s)$ is square, then the estimate $\hat{\boldsymbol{f}}(s) = \boldsymbol{H}_{rf}{}^{-1}(s)$ where the ijth element of $\boldsymbol{H}_{rf}{}^{-1}$, call it h_{ij} is the usual inverse

$$h_{ij}(s) = \frac{1}{\det(\boldsymbol{H}_{rf}(s))}(-1)^{i+j}(\boldsymbol{M}_{ji}(s)),$$

where $\boldsymbol{M}_{ji}(s)$ is the determinant of the matrix formed by $\boldsymbol{H}_{rf}(s)$ after deleting row j and column i.

If $\boldsymbol{H}_{rf}(s)$ is non-square, with l rows and n_f columns, then, there exists a left pseudo-inverse $\boldsymbol{G}(s)$ of $\boldsymbol{H}_{rf}(s)$ if and only if

$$\text{rank } (\boldsymbol{H}_{rf}(s)) = n_f,$$

where the normal rank is considered. $\boldsymbol{G}(s)$ is given as

$$\boldsymbol{G}(s) = (\boldsymbol{H}_{rf}^{\mathrm{T}}(s)\boldsymbol{H}_{rf}(s))^{-1}\boldsymbol{H}_{rf}^{\mathrm{T}}(s). \tag{6.72}$$

The pseudo-inverse has the property

$$\boldsymbol{G}(s)\boldsymbol{H}_{rf}(s) = \boldsymbol{I}_{n_f}$$

with $\boldsymbol{I}_{n_f}$ being the unity matrix of dimension n_f.

Remark 6.5 (*Causality of solution*) To be able to implement the filter Eq. (6.71), $\boldsymbol{G}(s)\,\boldsymbol{V}_{\mathrm{ru}}(s)$ and $\boldsymbol{G}(s)\,\boldsymbol{V}_{\mathrm{ry}}(s)$ must be proper and stable transfer functions. This may not be true when $\boldsymbol{G}(s)$ is computed as above. A modified procedure can be found in the literature (see the bibliographical notes for this chapter). □

Fault estimation after isolation. A necessary condition to be able to compute the above rational estimate, based on a pseudo inverse transformation, is that the rank of the $\boldsymbol{H}_{rf}$ matrix is equal to the number of faults to be estimated. As the number of faults is often larger than the number of independent residuals, it is necessary to take advantage of the results of the fault isolation to limit estimation of faults to those that the isolation algorithm found to be present in the system.

Assume the subset of the fault vector $\boldsymbol{f}_i$, $i \in [j, \ldots, k]$ has been determined necessary to estimate by the isolation algorithm. The above general expressions then hold for the entries of the transfer function matrices that relate to f_i, $i \in [j, \ldots, k]$.

Assume a single fault has been determined present. Then, a single column in $\boldsymbol{H}_{rf}(s)$ needs to be considered. The result for this simplest case can be formulated as follows.

Given, the stable residual generator

$$\boldsymbol{r}(s) = \boldsymbol{V}_{\mathrm{ru}}(s)\boldsymbol{u}(s) + \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{y}(s))$$

and a transfer function model relating this residual to faults

$$\boldsymbol{r}(s) = \boldsymbol{H}_{rf}(s)\,\boldsymbol{f}(s).$$

Assume that the isolation procedure indicates that fault number i is present, and let the ith column of $\boldsymbol{H}_{rf}(s)$ be

$$\boldsymbol{h}_i(s) = \frac{\bar{\boldsymbol{h}}_i(s)}{\boldsymbol{\eta}(s)},$$

where $\bar{\boldsymbol{h}}_i(s)$ is a polynomial vector with entries $\bar{\boldsymbol{h}}_{ji}(s)$ and $\boldsymbol{\eta}(s)$ is the least common denomination of the entries of $\boldsymbol{h}_i(s)$.

Theorem 6.1 (Single fault estimation) *On the condition that* $\boldsymbol{\eta}(s)$ *and* $\tilde{\boldsymbol{h}}_i^{\mathrm{T}}(s)\tilde{\boldsymbol{h}}_i(s) = \sum_{j=1}^{l} \bar{\boldsymbol{h}}_{ji}^2(s)$ *are stable polynomials, an estimate of* $\boldsymbol{f}_i$, $\hat{\boldsymbol{f}}_i$ *is given by:*

$$\hat{\boldsymbol{f}}_i(s) = \left(\boldsymbol{h}_i^{\mathrm{T}}(s)\,\boldsymbol{h}_i(s)\right)^{-1}\,\boldsymbol{h}_i(s)^{\mathrm{T}}\boldsymbol{r}(s). \tag{6.73}$$

This estimator is causal when $\deg \boldsymbol{\eta}(s) = \max\ \deg\ \bar{\boldsymbol{h}}_{ji}(s)$. This is easily proved by direct computation of the pseudo inverse in Eq. (6.73):

$$\left(\boldsymbol{h}_i(s)^{\mathrm{T}}\,\boldsymbol{h}_i(s)\right)^{-1}\,\boldsymbol{h}_i(s)^{\mathrm{T}} = \frac{(s)}{\sum_{i=1}^{l}\bar{\boldsymbol{h}}_{ji}^2(s)}\,\left(\bar{\boldsymbol{h}}_{1i}(s),\,\ldots,\,\bar{\boldsymbol{h}}_{li}(s)\right). \tag{6.74}$$

If the above condition on the degree is not met, a low-pass approximation for the fault estimate can be obtained by multiplying the denominator of Eq. (6.74) by $(s+)^{\beta}$, where $\alpha \in |\mathcal{R}^{+}$and β is chosen so that all entries in Eq. (6.74) are causal.

Example 6.9 Fault estimation - ship with three output measurements
Fault estimation following isolation for the ship with three output measurements results from the residual generator obtained in Example 6.7

$$\boldsymbol{H}_{rf}(s) = \begin{pmatrix} -\frac{1}{s} & 1 & 0 \\ -\frac{1}{s} & 0 & 1 \end{pmatrix} = \frac{1}{s}\begin{pmatrix} -1 & s & 0 \\ -1 & 0 & s \end{pmatrix}.$$

Fault 1 isolated: The estimate of fault number 1 is

$$\hat{f}_1 = \frac{s}{2}\,(\boldsymbol{r}_1(s) + \boldsymbol{r}_2(s)).$$

Since this filter is not causal, a low-pass filtered approximation for the rate gyro fault is needed, where $\alpha \in |\mathcal{R}^{+}$

$$\hat{f}_1 = \frac{s}{2(s+\alpha)}\,(\boldsymbol{r}_1(s) + \boldsymbol{r}_2(s)). \tag{6.75}$$

Fault 2 isolated: The estimate of fault number 2 is

$$\hat{\boldsymbol{f}}_2 = \boldsymbol{r}_1(s).$$

Fault 3 isolated: The estimate of fault number 3 is

$$\hat{\boldsymbol{f}}_3 = \boldsymbol{r}_2(s).$$

It should be noted that an erroneous isolation will give gross errors in the fault estimate.

In an implementation, the above fault estimators would run in parallel. Once a particular fault is isolated, the estimate can be rapidly provided. □

Alternative methods to fault estimation. In cases, where the above algebraic approach to fault estimation fails, asymptotic estimation of faults may be achievable using an observer on an augmented system, where the state is augmented by the fault(s) to be estimated (modelling faults to be constant):

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\boldsymbol{x}\\ \boldsymbol{f}\end{pmatrix} = \begin{pmatrix}\boldsymbol{A} & \boldsymbol{F}_x\\ \boldsymbol{O} & \boldsymbol{O}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}\\ \boldsymbol{f}\end{pmatrix} + \begin{pmatrix}\boldsymbol{B}\\ 0\end{pmatrix}\boldsymbol{u}(t)$$
$$\boldsymbol{y}(t) = \begin{pmatrix}\boldsymbol{C} & \boldsymbol{F}_y\end{pmatrix}\begin{pmatrix}\boldsymbol{x}\\ \boldsymbol{f}\end{pmatrix}.$$

A necessary condition for an asymptotically stable estimator to exist is that the pair

$$\left(\begin{pmatrix} \boldsymbol{A} & \boldsymbol{F}_x \\ \boldsymbol{O} & \boldsymbol{O} \end{pmatrix}, \begin{pmatrix} \boldsymbol{C} & \boldsymbol{F}_y \end{pmatrix} \right)$$

is observable. Observer-based methods are covered extensively in the literature (see the bibliographical notes for references).

In summary, the procedure for estimating the magnitude of a fault is as follows:

Algorithm 6.3 *Fault estimation*

Given: A model of the supervised process of the form (6.40) and a residual generator of the form (6.42)

Compute:

1. The transfer matrix $\boldsymbol{H}_{rf}(s)$ relating residuals to faults
2. A left inverse to $\boldsymbol{H}_{rf}(s)$
3. An estimator of the form (6.71), possibly after appropriate filtering of the left inverse in order to obtain a causal and stable estimator for all faults.

Result: A causal and stable fault estimator based on the measurements of $\boldsymbol{u}(s)$ and $\boldsymbol{y}(s)$.

6.5 Optimisation-Based Approach to Diagnosis

6.5.1 Problem Statement

The above methods were based on algebraic or polynomial manipulations, and relied on the ability to achieve exact decoupling from disturbances and from input to the residual. When this is not possible, the influence $\boldsymbol{d}(t)$ and $\boldsymbol{u}(t)$ have on the residual competes with that generated by faults $\boldsymbol{f}(t)$. If the effects of input and disturbance on the residual are non-zero, we do not obtain

$$\begin{pmatrix} \boldsymbol{V}_{\mathrm{ru}}(s) + \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yu}}(s) & \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yd}}(s) \end{pmatrix} \begin{pmatrix} \boldsymbol{u}(s) \\ \boldsymbol{d}(s) \end{pmatrix} = 0 \tag{6.76}$$

for all $\boldsymbol{u}(s)$ and $\boldsymbol{d}(s)$ and

$$\begin{aligned} \boldsymbol{r}(s) = &\begin{pmatrix} \boldsymbol{V}_{\mathrm{ru}}(s) + \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yu}}(s) & \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yd}}(s) \end{pmatrix} \begin{pmatrix} \boldsymbol{u}(s) \\ \boldsymbol{d}(s) \end{pmatrix} \\ &+ \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yf}}(s)\boldsymbol{f}(s) + \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yx}}(s)\boldsymbol{x}(0) \end{aligned} \tag{6.77}$$

is strictly speaking not a residual generator according to the definition.

The purpose of this section is to find ways to relax the requirement on exact decoupling for the residual generator. Instead, some optimal approximation should be obtained in the sense that the design shall satisfy certain criteria.

The design objectives should be to

1. Provide a sufficient suppression of disturbances $\boldsymbol{d}$ seen from the residual,
2. Maximise the sensitivity $\boldsymbol{r}$ of the residual with respect to all or a to a selected set of faults in $\boldsymbol{f}$.
3. Make the residual signal sufficiently insensitive to variations in the input signal $\boldsymbol{u}$.
4. Provide the designer with tools to enter a specification of the desired performance.

Formulating the design objectives as performance indices will enable a rigorous treatment. From the condition Eq. (6.76), perfect decoupling of disturbance requires

$$\boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yd}}(s) = \boldsymbol{O}.$$

Insensitivity to input requires the model to fulfil

$$\boldsymbol{V}_{\mathrm{ru}}(s) + \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yu}}(s) = \boldsymbol{O}$$

and both are subject to the constraint that sensitivity to faults is not vanishing

$$\boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yf}}(s) \neq \boldsymbol{O}$$

Norms and gains. In order to treat the relaxed condition, it is not required that the right-hand sides are exactly zero, but we wish to obtain minimal values subject to constraints like stable systems and causal realisation of filters. In order to formulate adequate optimisation problems, recall the definitions of the vector norm and the matrix norm induced by a vector norm: Let $\boldsymbol{x} \in |\mathcal{R}^n$. Then the vector p-norm of $\boldsymbol{x}$ is

$$|\boldsymbol{x}|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

In particular, when $p = 2, \infty$,

$$|\boldsymbol{x}|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$$

and

$$|\boldsymbol{x}|_\infty = \max_{1\le i\le n} |x_i| \,.$$

Further, let $\boldsymbol{A} = \left(a_{ij}\right) \in |\mathcal{R}^{m\times n}$ and $\boldsymbol{x} \in |\mathcal{R}^n$. The matrix norm induced by a vector p-norm is defined as

$$|\boldsymbol{A}|_p = \sup_{x\neq 0} \frac{|\boldsymbol{A}\boldsymbol{x}|_p}{|\boldsymbol{x}|_p}$$

In particular, when $p = 2, \infty$,

$$|\boldsymbol{A}|_2 = \sqrt{\lambda_{\max}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})} = \bar{\sigma}(\boldsymbol{A})$$

and

$$|\boldsymbol{A}|_\infty = \max_{1\le i\le m} \sum_{j=1}^{n} \left|a_{ij}\right| \quad \text{(maximum absolute row sum)},$$

where $\lambda_{\max}$ is the largest eigenvalue, and $\bar{\sigma}$ is the largest singular value.

It is noted that an induced norm can be viewed as a mapping from a vector space $\mathcal{C}^n$ equipped with a norm $|\cdot|_p$ to a vector space $\mathcal{C}^m$ with a norm $|\cdot|_p$. The induced norms have the interpretation of input/output amplification gains.

Let $\boldsymbol{H}(\mathrm{j}\omega) \in \mathcal{C}^{m\times n}$ be a stable transfer function, i.e. with all poles strictly in the left-half plane. Then the 2-norm is

$$|\boldsymbol{H}|_2 = \text{trace}\left(\frac{1}{2\pi}\int_{-\infty}^{\infty} \boldsymbol{H}(\mathrm{j}\omega)\,\boldsymbol{H}^{\mathrm{T}}(-\mathrm{j}\omega)\,\mathrm{d}\omega\right)^{\frac{1}{2}}$$

and the ∞-norm

$$|\boldsymbol{H}|_\infty = \max_\omega \bar{\sigma}\,(\boldsymbol{H}(\mathrm{j}\omega)).$$

An important result is that

$$|\boldsymbol{H}\boldsymbol{f}|_2^2 \le |\boldsymbol{H}|_\infty^2\,|\boldsymbol{f}|_2^2 = \max_\omega \bar{\sigma}(\boldsymbol{H}(\mathrm{j}\omega))^2\,|\boldsymbol{f}|_2^2$$

since

$$\begin{aligned}|\boldsymbol{H}\boldsymbol{f}|_2^2 &= \frac{1}{2\pi}\int_{-\infty}^{\infty} \boldsymbol{f}^{\mathrm{T}}(-\mathrm{j}\omega)\boldsymbol{H}^{\mathrm{T}}(-\mathrm{j}\omega)\boldsymbol{H}(\mathrm{j}\omega)\boldsymbol{f}(\mathrm{j}\omega)\,\mathrm{d}\omega \\ &\leq |\boldsymbol{H}|_\infty^2 \frac{1}{2\pi}\int_{-\infty}^{\infty} \boldsymbol{f}^{\mathrm{T}}(-\mathrm{j}\omega)\boldsymbol{f}(\mathrm{j}\omega)\,\mathrm{d}\omega \\ &= |\boldsymbol{H}|_\infty^2 |\boldsymbol{f}|_2^2,\end{aligned}$$

which shows $|\boldsymbol{H}|_\infty^2$ is the upper bound for the signal power transmitted from input to output of the transfer function $\boldsymbol{H}(s)$.

Formulation as an optimisation problem. The first property of a relaxed residual generator should be minimisation of the effect of disturbances in the residual.

A direct minimisation of the effect the disturbance has on the residual is expressed in the induced norm

$$\min_{\boldsymbol{V}_{\mathrm{ry}}} \boldsymbol{J}_{id} = \min_{\boldsymbol{V}_{\mathrm{ry}}} \frac{\left|\boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yd}}(s)\boldsymbol{d}(s)\right|_2^2}{|\boldsymbol{d}(s)|_2^2} = \min_{\boldsymbol{V}_{\mathrm{ry}}} \left|\boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yd}}(s)\right|_\infty^2$$

subject to

$$\boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yf}}(s) \neq \boldsymbol{O}.$$

The constraint prevents the trivial solution $\boldsymbol{V}_{\mathrm{ry}}(s) = 0$.

The signal power comprised in the residual caused by the disturbance over the power generated by faults should be minimised, hence a feasible index could be

$$\max_{\boldsymbol{V}_{\mathrm{ry}}} \boldsymbol{J}_2 = \max_{\boldsymbol{V}_{\mathrm{ry}}} \left(\frac{\left|\boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yf}}(s)\boldsymbol{f}(s)\right|_2^2}{\left|\boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yd}}(s)\boldsymbol{d}(s)\right|_2^2} \right)_{|\boldsymbol{d}|\neq 0}.$$

The interpretation of this index is to maximise the signal over noise ratio in the residual, using the total power, i.e. over all frequencies. This index cannot be easily optimised. If we, however, make a slight modification to the optimisation criterion, standard tools are available.

As a general tool for optimisation, the standard setup formulation is widely used in robust and optimal control theory and is widely supported by computer aided design tools. Hence, it is advantageous to describe the optimisation problem in the standard setup formulation.

Application of the standard methods require a specific formulation of the problem, which is first illustrated using manipulation on the block diagram in Fig. 6.4 for the case, where the objective is to find a polynomial matrix $\boldsymbol{F}(s)$ such that a signal $\boldsymbol{e}(s)$ is insensitive to a disturbance $\boldsymbol{d}(s)$

$$\boldsymbol{e}(s) = \left(\boldsymbol{H}_{\mathrm{zd}}(s) - \boldsymbol{F}(s)\boldsymbol{H}_{\mathrm{yd}}(s)\right)\, d(s).$$

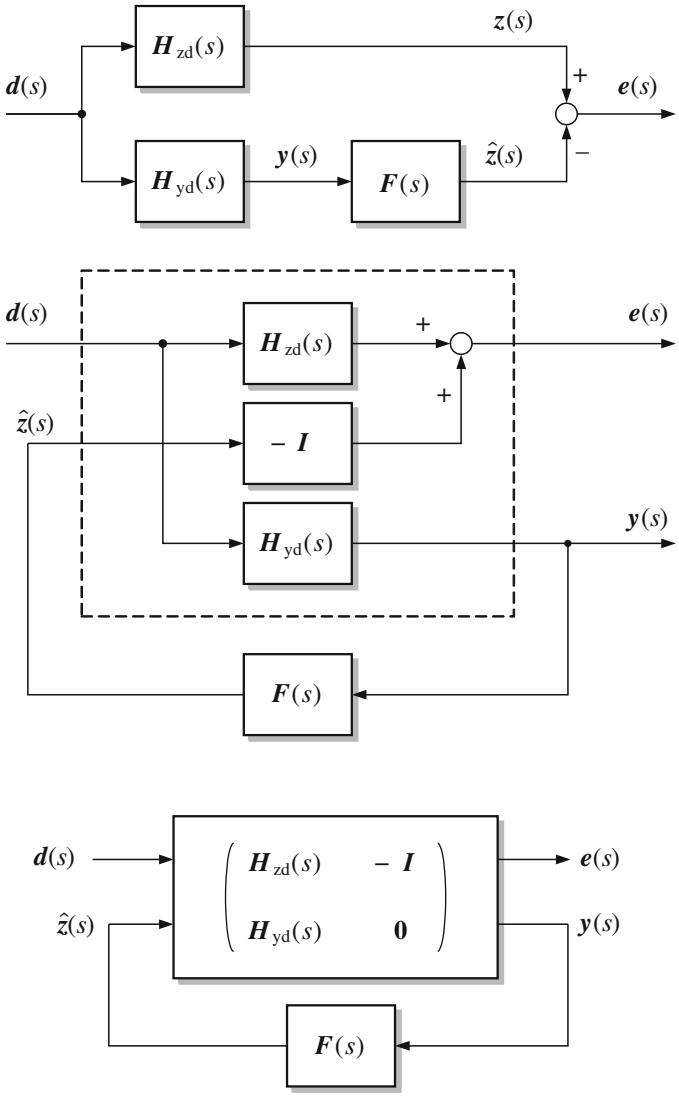


Fig. 6.4 Manipulation of the block diagram to arrive at a standard problem formulation. The *upper* two diagrams are equivalent, the *lower* is the representation used to determine $\boldsymbol{F}(s)$ by standard methods

6.5.2 Solution Using the Standard Setup Formulation

We introduce first the basic notion of the standard estimation setup and the standard estimation problem, which have a direct bearing on design of residual generators.

Definition 6.4 (*Standard estimation setup*) Let a system be given by input vector (known and unknown input) $\boldsymbol{d} \in |\mathcal{R}^{n_\mathrm{d}}$, state vector $\boldsymbol{x} \in |\mathcal{R}^{n}$, an auxiliary output

$\boldsymbol{z} \in |\mathcal{R}^l$ and measured output vector $\boldsymbol{y} \in |\mathcal{R}^p$ with state-space equation

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{E}_{\mathrm{x}}\boldsymbol{d}(t) \\ \boldsymbol{z}(t) &= \boldsymbol{C}_z\boldsymbol{x}(t) + \boldsymbol{E}_z\boldsymbol{d}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}_y\boldsymbol{x}(t) + \boldsymbol{E}_y\boldsymbol{d}(t) \end{aligned} \tag{6.78}$$

and, ignoring initial conditions, represented in the Laplace domain by

$$\begin{aligned} \boldsymbol{z}(s) &= \boldsymbol{H}_{\mathrm{zd}}(s)\boldsymbol{d}(s) \\ \boldsymbol{y}(s) &= \boldsymbol{H}_{\mathrm{yd}}(s)\boldsymbol{d}(s), \end{aligned} \tag{6.79}$$

where

$$\boldsymbol{H}_{\mathrm{zd}}(s) = \left(\begin{array}{cc} \boldsymbol{A} & \boldsymbol{E}_{\mathrm{x}} \\ \boldsymbol{C}_z & \boldsymbol{E}_{\mathrm{z}} \end{array}\right) = \boldsymbol{C}_{\mathrm{z}}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{E}_{\mathrm{x}} + \boldsymbol{E}_{\mathrm{z}} \tag{6.80}$$

$$\boldsymbol{H}_{\mathrm{yd}}(s) = \left(\begin{array}{cc} \boldsymbol{A} & \boldsymbol{E}_{\mathrm{x}} \\ \boldsymbol{C}_y & \boldsymbol{E}_y \end{array}\right) = \boldsymbol{C}_{\mathrm{y}}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{E}_{\mathrm{x}} + \boldsymbol{E}_{\mathrm{y}}. \tag{6.81}$$

Problem 6.4 (*Standard estimation problem*) Let $\hat{\boldsymbol{z}}(s)$ be an estimate of $\boldsymbol{z}(s)$. Denote the difference by $\boldsymbol{e}_{\mathrm{z}}(s) = \boldsymbol{z}(s) - \hat{\boldsymbol{z}}(s)$. For the system defined by the standard problem setup (6.79), determine a stable transfer function matrix $\boldsymbol{F}(s)$ to provide an estimate of the auxiliary output given the measured output,

$$\hat{\boldsymbol{z}}(s) = \boldsymbol{F}(s)\boldsymbol{y}(s) \tag{6.82}$$

subject to a suitable norm of the estimation error, $|\boldsymbol{e}_{\mathrm{z}}(s)|$ being less than a chosen gain factor

$$\sup_{\boldsymbol{F}(s)} |\boldsymbol{e}_{\mathrm{z}}(s)| < \gamma \Leftrightarrow \sup_{\boldsymbol{F}(s)} \left|\boldsymbol{H}_{\mathrm{zd}}(s) - \boldsymbol{F}(s)\boldsymbol{H}_{\mathrm{yd}}(s)\right| < \gamma, \tag{6.83}$$

where the norm can be of types $\mathcal{H}_2$ or $\mathcal{H}_\infty$ for instance.

Equation (6.83) follows from expanding the estimation error:

$$\begin{aligned} \boldsymbol{e}_{\mathrm{z}}(s) &= \boldsymbol{z}(s) - \hat{\boldsymbol{z}}(s) \\ &= \boldsymbol{z}(s) - \boldsymbol{F}(s)\boldsymbol{y}(s) \\ &= (\boldsymbol{H}_{\mathrm{zd}}(s) - \boldsymbol{F}(s)\boldsymbol{H}_{\mathrm{yd}}(s))\boldsymbol{d}(s). \end{aligned} \tag{6.84}$$

Remark 6.6 Different filtering and estimation problems can be easily formulated within this general estimation framework. The state can be estimated using $\boldsymbol{C}_{\mathrm{z}} = \boldsymbol{I}_{n,n}$ and $\boldsymbol{E}_{\mathrm{z}} = \boldsymbol{O}$. The input can be estimated using $\boldsymbol{C}_{\mathrm{z}} = \boldsymbol{O}$ and $E_{\mathrm{z}} = I_{l,l}$. The estimation setup will be used later for residual generation. □

Standard $\mathcal{H}_2$ and $\mathcal{H}_\infty$ methods that find the minimum of function according to the selected norm can also be applied to find a suitable $\boldsymbol{F}(s)$ transfer function matrix

for the estimation problem. Use of widely available software for this purpose (for example the MATLAB μ toolbox) requires formulation in what is referred to as the *standard system setup and standard problem* in robust control.

Definition 6.5 (*Standard system setup*) Let a system be described in the Laplace domain by the transfer function matrix $P(s)$, and four vectors, input $\boldsymbol{u}(s) \in \mathcal{C}^m$, auxiliary input $\boldsymbol{d}(s) \in \mathcal{C}^{n_\mathrm{d}}$, auxiliary output $\boldsymbol{e}(s) \in \mathcal{C}^{m_e}$ and measured output $\boldsymbol{y}(s) \in \mathcal{C}^p$. Input and output are related through $\boldsymbol{P}(s) \in \mathcal{C}^{(p+m_e)\times(m+n_\mathrm{d})}$ as

$$\begin{pmatrix} \boldsymbol{e}(s) \\ \boldsymbol{y}(s) \end{pmatrix} = \boldsymbol{P}(s) \begin{pmatrix} \boldsymbol{d}(s) \\ \boldsymbol{u}(s) \end{pmatrix} = \begin{pmatrix} \boldsymbol{P}_{ed}(s) & \boldsymbol{P}_{eu}(s) \\ \boldsymbol{P}_{\mathrm{yd}}(s) & \boldsymbol{P}_{\mathrm{yu}}(s) \end{pmatrix} \begin{pmatrix} \boldsymbol{d}(s) \\ \boldsymbol{u}(s) \end{pmatrix}$$

Let the transfer function matrix $\boldsymbol{F}(s) \in \mathcal{C}^{m\times p}$ be a feedback controller for the system, between $\boldsymbol{y}$ and $\boldsymbol{u}$,

$$\boldsymbol{u}(s) = \boldsymbol{F}(s)\boldsymbol{y}(s)$$

Using this setup and utilising solutions for two fundamental optimisation problems in the design of residual generators, the $\mathcal{H}_\infty$ sub-optimal control problem and the $\mathcal{H}_2$ sub-optimal control problem.

Problem 6.5 ($\mathcal{H}_\infty$ *sub-optimal control*) Given a system in form of the standard system setup of Definition 6.5. Design a stabilising controller $\boldsymbol{F}(s)$ such that the norm of the closed-loop transfer function $\boldsymbol{T}_{ed}(s)$ from auxiliary input $\boldsymbol{d}(s)$ to auxiliary output $\boldsymbol{e}(s)$ is lower than a specified bound γ:

$$\sup_{\boldsymbol{F}} |T_{ed}|_\infty < \gamma \Leftrightarrow \sup_{\boldsymbol{F}(\mathrm{j}\omega)} \bar{\sigma}(T_{ed}(\mathrm{j}\omega)) < \gamma,$$

where $\bar{\sigma}$ denotes the largest singular value.

The $\mathcal{H}_\infty$ norm gives the maximum sinusoidal gain of the system (energy gain or induced L_2 system gain).

Problem 6.6 ($\mathcal{H}_2$ *sub-optimal control problem*) Given a system in form of the standard system setup in Definition 6.5. Design a stabilising controller $\boldsymbol{F}(s)$ such that the $\mathcal{H}_2$ norm of the closed-loop transfer function $\boldsymbol{T}_{ed}(s)$ from auxiliary input $\boldsymbol{d}(s)$ to auxiliary output $\boldsymbol{e}(s)$ is minimised.

The standard estimation problem of Fig. 6.4 can be formulated in the standard setup. The generalised system $\boldsymbol{P}(s)$ then takes the form

$$\boldsymbol{P}(s) = \begin{pmatrix} \boldsymbol{H}_{\mathrm{zd}}(s) & -\boldsymbol{I} \\ \boldsymbol{H}_{\mathrm{yd}}(s) & \boldsymbol{O} \end{pmatrix}.$$

Note that there is no feedback through the system since $\boldsymbol{P}_{\mathrm{yu}}(s) = \boldsymbol{O}$.

6.5.3 Residual Generation

The above result can be applied in connection with detection, isolation and estimation of faults. We aim at making $\hat{z}(s)$ a residual signal. We investigate two problems. The first is to suppress disturbances as well as possible. The second is to make a balanced optimisation, where the fault signature is preserved in the residual while disturbances are suppressed to the extent possible. Both results follow from appropriate formulation of the standard problem. The strategy is to select an auxiliary output $z(s)$ and give it the properties that the residual should have. This means the formulation of $z(s)$ is directly a specification of the residual. In designing the estimate $\hat{z}(s)$ to track $z(s)$ as closely as possible, according to a given criterion, a sub-optimal estimator is obtained for the ideal residual. The accuracy with which the specification is met is seen in the choice of the optimisation coefficient γ.

The basic residual generator will have the form

$$\boldsymbol{r}(s) = \boldsymbol{F}(s)(\boldsymbol{y}(s) - \boldsymbol{H}_{\mathrm{yu}}(s)\boldsymbol{u}(s)). \tag{6.85}$$

The design problem is to determine the operator $\boldsymbol{F}(s)$.

Remark 6.7 (*Relation to parity space formulation*) The residual generator Eq. (6.43) had as prerequisite, following from Problem 6.2, that

$$\boldsymbol{V}_{\mathrm{ru}} + \boldsymbol{V}_{\mathrm{ry}}\boldsymbol{H}_{\mathrm{yu}} = \boldsymbol{O}$$

hence

$$\begin{aligned}\boldsymbol{r}(s) &= \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{y}(s) + \boldsymbol{V}_{\mathrm{ru}}\boldsymbol{u}(s)\\ &= \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{y}(s) - \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yu}}\boldsymbol{u}(s)\\ &= \boldsymbol{V}_{\mathrm{ry}}(s)(\boldsymbol{y}(s) - \boldsymbol{H}_{\mathrm{yu}}(s)\boldsymbol{u}(s)).\end{aligned}$$

Comparison with Eq. (6.85) shows that finding the solution $\boldsymbol{F}(s)$ in the standard setup is equivalent to determining the operator $\boldsymbol{V}_{\mathrm{ry}}(s)$. □

In the design, two requirements have to be combined.

Residual generation with specification on fault sensitivity and disturbance suppression. The goal is now to have the residual replicating a fault through a specified dynamical relation while the disturbance should be suppressed as far as possible. Therefore, we include the fault vector $\boldsymbol{f}(s)$ in the system description and define the auxiliary output $z(s)$ to be dependent only of the fault vector:

$$\begin{aligned}\boldsymbol{y}(s) &= \boldsymbol{H}_{\mathrm{yd}}(s)\boldsymbol{d}(s) + \boldsymbol{H}_{\mathrm{yf}}(s)\boldsymbol{f}(s)\\ z(s) &= \boldsymbol{H}_{\mathrm{zf}}(s)\boldsymbol{f}(s)\\ \hat{z}(s) &= \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{y}(s)\\ \boldsymbol{e}_{\mathrm{z}}(s) &= z(s) - \hat{z}(s)\end{aligned}$$

The selection of the auxiliary output reflects directly the properties that the residual should have. $\boldsymbol{H}_{\mathrm{zd}} = \boldsymbol{O}$ is chosen because we wish to interpret $\boldsymbol{d}(s)$ as a disturbance and decouple it from the residual. The specification of $\boldsymbol{H}_{\mathrm{zf}}(s)$ is a design choice. There may not exist a solution $\boldsymbol{V}_{\mathrm{ry}}(s)$ for all arbitrary specifications, so $\boldsymbol{H}_{\mathrm{zf}}(s)$ is the key design parameter.

The performance that should be achieved is that the residual follows $\boldsymbol{z}(s)$ as close as possible, hence, the relation

$$\left|\boldsymbol{H}_{\mathrm{zf}}(s) - \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yf}}(s)\right|_{\infty} < \gamma_s$$

should hold, where γ_s characterises the desired fault sensitivity (or tracking) performance. Simultaneously, the effect of the disturbance should be below a certain level, hence

$$\left|\boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yd}}(s)\right|_{\infty} < \gamma_r,$$

where γ_r is a measure of robustness with respect to input effects. Combining the two, the physically motivated optimisation problem yields:

$$\left|\left(-\boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yd}}(s)\right) \quad \left(\boldsymbol{H}_{\mathrm{zf}}(s) - \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yf}}(s)\right)\right|_{\infty} < \gamma. \tag{6.86}$$

Since

$$\begin{aligned}
&\boldsymbol{z}(s) - \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{y}(s) \\
&= \left(\boldsymbol{H}_{\mathrm{zf}}(s) - \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yf}}(s)\right)\boldsymbol{f}(s) - \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yd}}(s)\boldsymbol{d}(s) \\
&= \left(\left(-\boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yd}}(s)\right) \quad \left(\boldsymbol{H}_{\mathrm{zf}}(s) - \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yf}}(s)\right)\right)\begin{pmatrix}\boldsymbol{d}(s)\\ \boldsymbol{f}(s)\end{pmatrix}
\end{aligned}$$

Equation (6.86) is equivalent to

$$\sup_{\begin{pmatrix}\boldsymbol{d}\\ \boldsymbol{p}\end{pmatrix}\neq\boldsymbol{0}} \frac{\left|\boldsymbol{z}(\mathrm{j}\omega) - \boldsymbol{V}_{\mathrm{ry}}(\mathrm{j}\omega)\boldsymbol{y}(\mathrm{j}\omega)\right|_2}{\left|\begin{pmatrix}\boldsymbol{d}(\mathrm{j}\omega)\\ \boldsymbol{f}(\mathrm{j}\omega)\end{pmatrix}\right|_2} < \gamma \Leftrightarrow \sup_{\begin{pmatrix}\boldsymbol{d}\\ \boldsymbol{p}\end{pmatrix}\neq\boldsymbol{0}} \frac{|\boldsymbol{e}_{\mathrm{z}}(\mathrm{j}\omega)|_2}{\left|\begin{pmatrix}\boldsymbol{d}(\mathrm{j}\omega)\\ \boldsymbol{f}(\mathrm{j}\omega)\end{pmatrix}\right|_2} < \gamma.$$

The residual generation design problem is illustrated in Fig. 6.5. The upper diagram in the Figure depicts the residual generator with both specifications $\boldsymbol{H}_{\mathrm{zd}}$ and $\boldsymbol{H}_{\mathrm{zf}}$ given. In formulating the requirement that disturbance feed-through to the residual should be minimal, $\boldsymbol{H}_{\mathrm{zd}} = \boldsymbol{O}$ is specified in the setup shown in the lower part of Fig. 6.5.

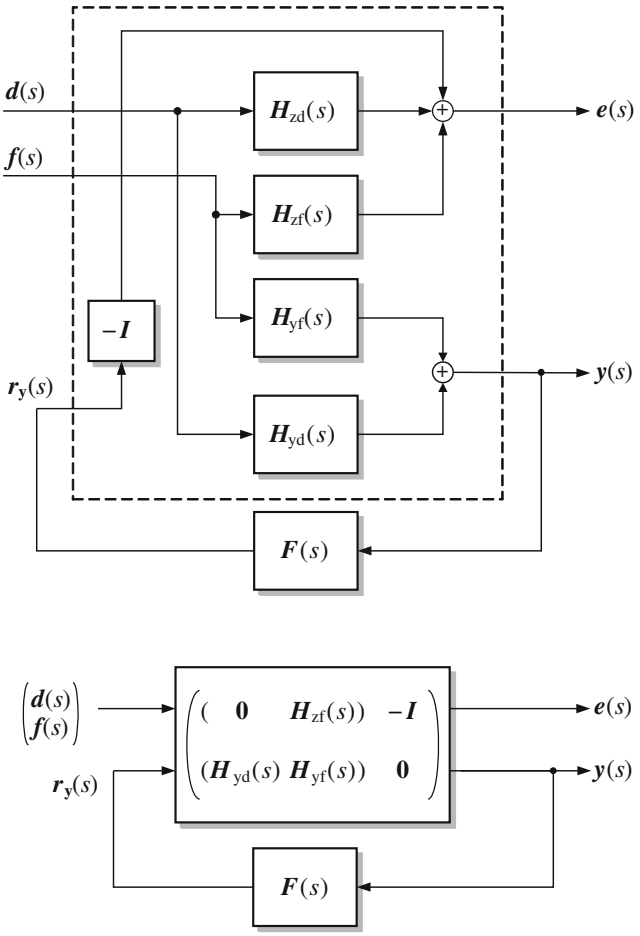


Fig. 6.5 Residual generator depicted in a standard setup formulation with specifications $\boldsymbol{H}_{\mathrm{zf}}$ and $\boldsymbol{H}_{\mathrm{zd}}$ in the *upper* part of the figure. $\boldsymbol{H}_{\mathrm{zd}}$ is specified as zero in the design problem shown in the *lower* part of the figure

In the standard setup, the residual generator design has the following form:

Problem 6.7 (*Residual generation with specification on fault sensitivity and disturbance suppression*) Given, an LTI system with input $\mathbf{u}(s)$, unknown input (disturbances) $\mathbf{d}(s)$ and faults $\mathbf{f}(s)$ and let input-output relations of the system be described by $(\boldsymbol{H}_{\mathrm{yu}}, \boldsymbol{H}_{\mathrm{yd}}, \boldsymbol{H}_{\mathrm{yf}})$. Introduce an auxiliary variable $\mathbf{z}(s)$ and specify a transfer function matrix $\boldsymbol{H}_{\mathrm{zf}}$ and a real number γ_s. Let $\mathbf{z}(s) = \boldsymbol{H}_{\mathrm{zf}}\mathbf{f}(s)$. Determine $\boldsymbol{V}_{\mathrm{ry}}$ such that the maximal deviation between $\hat{\mathbf{z}}(s) = \boldsymbol{V}_{\mathrm{ry}}\mathbf{y}(s)$ and $z(s)$ is bounded by γ_s:

$$|\mathbf{z}(s) - \mathbf{z}(\hat{s})| < \gamma_s \tag{6.87}$$

The solution to this problem of residual generation design has the following form:

1. Define the standard problem setup:

$$\begin{aligned}
\text{aux.input}: \quad & \boldsymbol{d}(s) \leftarrow \begin{pmatrix} \boldsymbol{d}(s) \\ \boldsymbol{f}(s) \end{pmatrix} \\
\text{input}: \quad & \boldsymbol{u}(s) \leftarrow \boldsymbol{r}(s) \\
\text{aux.output}: \quad & \boldsymbol{e}(s) \leftarrow \boldsymbol{e}_{\mathrm{z}}(s) = \boldsymbol{z}(s) - \boldsymbol{r}(s) \\
\text{output}: \quad & \boldsymbol{y}(s) \leftarrow \boldsymbol{y}(s) \\
& \boldsymbol{P}(s) \leftarrow \begin{pmatrix} \begin{pmatrix} \boldsymbol{O} & \boldsymbol{H}_{\mathrm{zf}}(s) \end{pmatrix} & -\boldsymbol{I} \\ \begin{pmatrix} \boldsymbol{H}_{\mathrm{yd}}(s) & \boldsymbol{H}_{\mathrm{yf}}(s) \end{pmatrix} & \boldsymbol{O} \end{pmatrix} \\
& \boldsymbol{F}(s) \leftarrow \boldsymbol{V}_{\mathrm{ry}}(s)
\end{aligned}$$

2. Use software that solves the standard problem to determine a solution in form of a stable transfer function $\boldsymbol{V}_{\mathrm{ry}}(s)$ that satisfies the inequality

$$\sup_{\begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{f} \end{pmatrix} \neq \boldsymbol{0}} \frac{|\boldsymbol{e}_{\mathrm{z}}|_2}{\left| \begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{f} \end{pmatrix} \right|_2} < \gamma$$

which is equivalent to finding a solution to

$$\left| \begin{pmatrix} -\boldsymbol{V}_{\mathrm{ry}}\boldsymbol{H}_{\mathrm{yd}} & \boldsymbol{H}_{\mathrm{zf}} - \boldsymbol{V}_{\mathrm{ry}}\boldsymbol{H}_{\mathrm{yf}} \end{pmatrix} \right|_\infty < \gamma.$$

If a result exists, which is not guaranteed, the result is strong in the sense it provides the residual generator with optimal weighting between suppression of disturbances and specified sensitivity to faults.

In practice it is worthwhile to start a design with investigating the extent to which disturbances can be suppressed using the disturbance suppression problem. When insight in the problem has been gained, continue with supplying a specification to the problem and iterate until a suitable compromise has been found between disturbance suppression and fault tracking.

Fault detection. When the purpose is to design a pure fault detection filter, a sensible way to specify $\boldsymbol{H}_{\mathrm{zf}}(s)$ is to require that it is a row vector with non-zero causal and stable entries. When a residual vector is sought the specification becomes:

$$\forall \omega;\, j\omega \neq z_k : \begin{cases} \text{rank } (\boldsymbol{H}_{\mathrm{zf}}(\mathrm{j}\omega)) \geq 1 \\ \forall i : h_i(\mathrm{j}\omega) \neq 0, \end{cases}$$

where $h_i(\mathrm{j}\omega)$ stands for the ith column of $\boldsymbol{H}_{\mathrm{zf}}(\mathrm{j}\omega)$ and z_k are the zeros of $\boldsymbol{H}_{\mathrm{zf}}(s)$.

Fault isolation. If the number of faults to be isolated is $\boldsymbol{n}_f$ and simultaneous faults can occur, $\boldsymbol{H}_{\mathrm{zf}}(s)$ has to fulfil the requirement:

$$\text{rank } \boldsymbol{H}_{\mathrm{zf}}(s) = \boldsymbol{n}_f,$$

where, as usual, the normal rank is considered.

When simultaneous faults are not considered, a vector $\boldsymbol{z}$ of size l is sufficient to isolate $2^l - 1$ faults by considering suitable coding sets. This translates into the following specification for matrix $\boldsymbol{H}_{\mathrm{zf}}(s)$. To isolate n_f faults, choose a matrix $\boldsymbol{H}_{\mathrm{zf}}(s)$ such that:

$$\begin{aligned} &\text{rank } \boldsymbol{H}_{\mathrm{zf}}(s) \geq \log_2(n_f + 1) \\ &\text{rank } \big(\boldsymbol{h}_i(s)\, \boldsymbol{h}_j(s)\big) = 2 \quad \text{with} \quad i = 1, \ldots, \boldsymbol{n}_f \quad i \neq j, \ j = 1, \ldots, n_f. \end{aligned}$$

Fault estimation. Fault estimation can be obtained by specifying $\boldsymbol{H}_{\mathrm{zf}}(s) = \boldsymbol{I}$. In this case,

$$\boldsymbol{z}(s) = \boldsymbol{I}\, f(s)$$

and

$$\boldsymbol{e}(s) = \boldsymbol{z}(s) - \hat{\boldsymbol{z}}(s).$$

In the ideal situation, where no disturbance exists, this specification aims at assuring that $\hat{\boldsymbol{z}}$ tracks the fault f by guaranteeing that

$$\sup \frac{|\hat{\boldsymbol{z}} - \boldsymbol{z}|_2}{|f|_2} < \gamma.$$

When disturbances do exist, a trade-off is made between fault tracking and insensitivity of $\hat{\boldsymbol{z}}$ to the disturbance. The block diagram to specify fault estimation from the solution to a standard problem is shown in Fig. 6.6.

Design considerations. In connection with using $\mathcal{H}_2$ or $\mathcal{H}_\infty$ optimisation to design the residual generator, a weight function can further be included in the setup to some advantage of the designer. The weight function $\boldsymbol{W}(s)$ can be applied to specify the frequency range(s), where detection, isolation or estimation should be obtained most effectively. The way to include a weight matrix in the design is to modify the $\boldsymbol{P}(s)$ system matrix to

$$\boldsymbol{P}(s) = \begin{pmatrix} \begin{pmatrix} \boldsymbol{O} & \boldsymbol{W}\boldsymbol{H}_{\mathrm{zf}} \end{pmatrix} & -\boldsymbol{W} \\ \begin{pmatrix} \boldsymbol{H}_{\mathrm{yd}} & \boldsymbol{H}_{\mathrm{yf}} \end{pmatrix} & \boldsymbol{O} \end{pmatrix}.$$

The weighting matrix specifies which frequency ranges a designer emphasizes to meet the bound γ and where it can be relaxed.

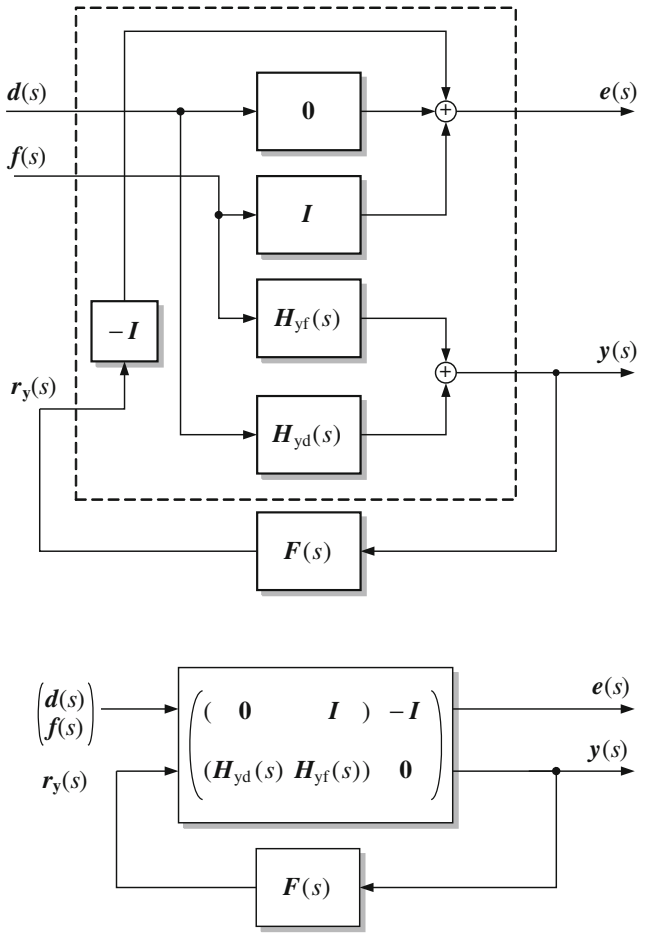


Fig. 6.6 If a solution $\boldsymbol{F}(s)$ exists, fault estimation is obtained by solving the standard problem using the specification $\boldsymbol{H}_{\mathrm{zd}} = \boldsymbol{O}$, $\boldsymbol{H}_{\mathrm{zf}} = \boldsymbol{I}$

The main issue in using the standard setup to obtain sub-optimal residual generators of the different classes described above is the selection of a proper specification $\boldsymbol{H}_{\mathrm{zf}}(s)$. Whereas the optimisation itself is left to the software tools available, good results are only obtained if a good specification is provided. An iterative design method has proved useful in practice.

Algorithm 6.4 *Residual generator design*

1. **Formulate problem:** Formulate the relevant version of the standard setup for the problem.
2. **Design specification:** Specify an initial qualified guess on the specification $\boldsymbol{H}_{\mathrm{zf}}(s)$. The specification needs to be bounded from below, otherwise, the optimal solution will be $\boldsymbol{F} = \boldsymbol{O}$ and $\boldsymbol{H}_{\mathrm{zf}} = \boldsymbol{O}$.
3. **Solve problem:** Find the function $\boldsymbol{F}(s)$ in the residual generator

$$\boldsymbol{r}(s) = \boldsymbol{F}(s)(\boldsymbol{y}(s) - \boldsymbol{H}_{\mathrm{yu}}(s)\boldsymbol{u}(s)),$$

   where $\boldsymbol{F}(s)$ is the best obtainable solution to the problem given the specification $\boldsymbol{H}_{\mathrm{zf}}(s)$.
4. **Iterate until converged:** Continue until the value of γ obtained has converged.
5. **Iterate in specification:** Based on this residual generator, specify a new $\boldsymbol{H}_{\mathrm{zf}}(s)$ and repeat the design.

The procedure usually requires very few iterations.

6.6 Residual Evaluation

Given, a residual generator for the deterministic case, i.e. there are only insignificant random disturbances or measurement noise, the purpose of this section is to find a method for residual evaluation that will determine whether a fault is present.

6.6.1 Residual - General Case

Let a set of residuals obtained from structural analysis have the form

$$\boldsymbol{r} = (r_1, r_2, \ldots, r_n)^{\mathrm{T}}.$$

Consider one of these residuals

$$r_j(t) = p_j(k_i, c_i, t) \qquad k_i \in K^{(j)}, c_i \in C^{(j)}, \quad j = 1, \ldots, n, \tag{6.88}$$

where p_j is of the form in which the constraints in $C^{(j)}$ were formulated: linear, nonlinear, tabular, quantised, logical or hybrid. As it is useful to categorise known variables into the natural categories input u, measured y, and parameters θ, the parity

relation is written as

$$r_j(t) = p_j(u_i, y_i, \theta_i, c_i, t) \quad u_i, y_i, \theta_i \in K^{(j)}, c_i \in C^{(j)}, \quad j = 1, \ldots, n. \quad (6.89)$$

The parity relations implemented for residual generation would not be the true system constraints c_i nor the true parameters θ_i but would be estimates of those, $\hat{c}_i, \hat{\theta}_i$, respectively.

In order to shape the signatures of faults in the residuals or suppress noise, filtering of the raw parity relation Eq. (6.89) will usually take place, and also the filtered version is a residual,

$$r_j(t) = \int_0^t w_j(t-\tau) p_j(u_i, y_i, \hat{\theta}_i, \hat{c}_i, \tau) \mathrm{d}\tau, \quad j = 1, \ldots, n, \quad (6.90)$$

where $w_j(t-\tau)$ is the impulse response of the filter applied to parity relation j.

Further, the vector of residuals could be constructed as a linear combination of the elements from the above residuals, Eq. (6.90),

$$\boldsymbol{r}(t) = \boldsymbol{W} \begin{pmatrix} r_1(t) \\ \vdots \\ r_n(t) \end{pmatrix}, \quad (6.91)$$

where $\boldsymbol{W} \in |\mathcal{R}^{n \times n}$, $\det(\boldsymbol{W}) \neq 0$.

Uncertainty. In real life, $\hat{c}_i \neq c_i$, and $\hat{\theta}_i \neq \theta_i$, hence $r_j(t)$ could be non-zero even though there was no violation of a constraint in relation j, $\forall c_i \in C^{(j)} : c_i = 0$. In particular, actuator demand and disturbances could drive the residual away from zero when parameters and constraints are not exactly equal to those of the real object. In order to make residual evaluation under such uncertainty, it is necessary to accept that a residual can have some deviation from zero even in the no-fault case. However, the effect on $r_j(t)$ has to be bounded, hence $p_j(u_i, y_i, \hat{\theta}_i, \hat{c}_i, t)$ is bounded,

$$\left\| p_j(u_i, y_i, \hat{\theta}_i, \hat{c}_i, t) \right\| \leq \alpha_j(u_i, y_i, t) \wedge 0 < \alpha(u_i, y_i, t) < \infty. \quad (6.92)$$

LTI case. If the object for diagnosis is linear and time-invariant (LTI), the residual generator could be LTI with a frequency representation

$$\boldsymbol{r}(s) = \boldsymbol{H}_{\mathrm{ru}}(s)\boldsymbol{u}(s) + \boldsymbol{H}_{rd}(s)\boldsymbol{d}(s) + \boldsymbol{H}_{rf}(s)\boldsymbol{f}(s) \quad (6.93)$$

being an explicit function of input, disturbances and faults.

In an ideal case, residual generation is perfect and we have $\boldsymbol{H}_{\mathrm{ru}}(s) = \boldsymbol{O}$ and $\boldsymbol{H}_{rd}(s) = \boldsymbol{O}$. Residual evaluation then reduces to investigating the properties of

$$\boldsymbol{r}(s) = \boldsymbol{H}_{rf}(s)\boldsymbol{f}(s). \tag{6.94}$$

In the general case, still with an LTI system, model uncertainty and unmodelled dynamics will give rise to $\boldsymbol{H}_{\mathrm{ru}}(s) \neq \boldsymbol{O}$ and $\boldsymbol{H}_{rd}(s) \neq \boldsymbol{O}$. Residual evaluation need then be made such that false alarms are avoided from control input $\boldsymbol{u}(t)$ and disturbances $\boldsymbol{d}(t)$ within the normal range.

6.6.2 Evaluation Against a Threshold

Validating that no fault is present is equivalent with checking that the residual vector is zero. Validating the presence of a fault means checking whether the residual is or has been different from zero. The two hypotheses and the associated condition on the residual vector are

$$\begin{array}{lll} \mathcal{H}_0(0,t): & \text{no fault is present} & \|\boldsymbol{r}\| = 0 \\ \mathcal{H}_1(f_j,t_j): & \text{fault } f_j \text{ was present since time } t_j & \|\boldsymbol{r}(t)\| \neq 0, t \geq t_j, \end{array} \tag{6.95}$$

where $\|\boldsymbol{r}\|$ is an appropriate norm of the residual.

Test function. For generality, introduce a test function $\varphi(r(t))$, which provides a measure (norm) of the residual's deviation from zero. Some common test functions are the following

- Absolute value
$$\varphi(r_j(t)) = \left|r_j(t)\right|. \tag{6.96}$$

- An approximation to the two-norm of the residual vector
$$\varphi(r_j(t)) = \left(\frac{1}{T}\int_{t-T}^{t} \left|r_j(\tau)\right|^2 \mathrm{d}\tau\right)^{\frac{1}{2}}. \tag{6.97}$$

- Square root of filtered absolute value, squared,
$$\varphi(r_j(t)) = \left(\int_0^t w_{\varphi}^{(j)}(t-\tau)\left|r_j(\tau)\right| \mathrm{d}\tau\right)^{\frac{1}{2}}. \tag{6.98}$$

- Filtered mean square value of signal
$$\varphi(r_j(t)) = \int_0^t w_{\varphi}^{(j)}(t-\tau)\left(r_j(\tau) - \frac{1}{T}\int_{\tau-T}^{\tau} r_j(\tau_2)\mathrm{d}\tau_2\right)^2 \mathrm{d}\tau, \tag{6.99}$$

where $w_{\varphi}^{(j)}(t)$ is the impulse response of a filter used particularly for evaluation of residual j. In this context, the test function given in Eq. (6.96) is considered further.

Threshold function. The next step in residual evaluation is to determine a threshold function $\Phi(t)$ for evaluation of the test function $\varphi(t)$. $\Phi(t)$ should have the properties

no fault:	$\forall t \geq 0,\ f(t) = 0:$	$\varphi(r(t)) \leq \Phi(t)$
weakly detectable fault:	$\exists t \geq t_0 : f(t) \neq 0:$	$\varphi(r(t)) > \Phi(t)$
strongly detectable fault:	$\forall t \geq t_1 \geq t_0 : f(t) \neq 0,\ t \geq t_0 :$	$\varphi(r(t)) > \Phi(t)$

LTI case. In the ideal LTI case, Eq. (6.94), $\Phi(t)$ could be chosen constant and as close to zero as allowed by practical values of bias and noise in the residual.

In the non-ideal case, Eq. (6.93) applies and input and disturbances have some feed-through to the residual. With the test function $\varphi(t) = \left\| r_j(t) \right\|_2$, the threshold need be determined such that

$$\Phi_j(t) \geq \sup_{f=0, \|u,d\| < \varepsilon} (\varphi(r_j(t)))$$

is achieved in the time domain. The fact that total power calculated in the time domain and in the frequency domain are equal is used to determine the threshold function,

$$||r(\mathrm{j}\omega)||^2 = \frac{1}{2\pi} \int_0^{\infty} r(\mathrm{j}\omega) r(-\mathrm{j}\omega)\, \mathrm{d}\omega = \lim_{T \to \infty} \int_0^{T} |r(t)|^2 \mathrm{d}t = ||r(t)||^2.$$

From Eq. (6.93), the residual is given in the frequency domain. Component j of the residual is

$$r_j(s) = (\boldsymbol{H}_{\mathrm{ru}}(s)\boldsymbol{u}(s))^{(j)} + (\boldsymbol{H}_{rd}(s)\boldsymbol{d}(s))^{(j)} + \left(\boldsymbol{H}_{rf}(s)\boldsymbol{f}(s)\right)^{(j)}.$$

With k control inputs

$$\begin{aligned} \left\| r_j(\mathrm{j}\omega) \right\|_2 &\leq \| \boldsymbol{H}_{\mathrm{ru}}(\mathrm{j}\omega)\boldsymbol{u}(\mathrm{j}\omega) \|_2^{(j)} + \| \boldsymbol{H}_{rd}(\mathrm{j}\omega)\boldsymbol{d}(\mathrm{j}\omega) \|_2^{(j)} \\ &\leq \sum_{i=1}^{k} \| \boldsymbol{H}_{\mathrm{ru}}(\mathrm{j}\omega) \|_{\infty}^{(ji)} \| u_i(\mathrm{j}\omega) \|_2 + \| \boldsymbol{H}_{rd}(\mathrm{j}\omega)\boldsymbol{d}(\mathrm{j}\omega) \|_{\infty}^{(j)} \end{aligned} \tag{6.100}$$

for all admissible $\boldsymbol{u}$ and $\boldsymbol{d}$. The first term in the right hand side is a gain times input power. The second is the maximal contribution to the residual from disturbances.

Let the effect of disturbances on the residual be bounded by

$$\| \boldsymbol{H}_{rd}(\mathrm{j}\omega)\boldsymbol{d}(\mathrm{j}\omega) \|_{\infty}^{(j)} < \beta_{\mathrm{d}}^{(j)}, \tag{6.101}$$

then $\Phi(t)$ should be chosen as the time-varying function

$$\Phi_j(t) = \sum_{i=1}^{k} \beta_i \, \|u_i(t)\|_2 + \beta_{\mathrm{d}}^{(j)} \tag{6.102}$$
$$\beta_i = \|\boldsymbol{H}_{\mathrm{ru}}(\mathrm{j}\omega)\|_{\infty}^{(ji)} .$$

This threshold is a function of maximal gains from control inputs to residual and of the maximum gain from disturbances to residual. It is often referred to as a time-varying threshold in the literature. The term adaptive threshold has also been used.

If the time-varying threshold Eq. (6.102) is too conservative, a dynamical bound could be specified as

$$\Phi_j(t) = \sum_{i=1}^{k} \left(\int_0^t \hat{h}_{\mathrm{ru}}^{(ji)}(t-\tau) u_i(\tau) \mathrm{d}\tau \right) + \beta_{\mathrm{d}}^{(j)}, \tag{6.103}$$

where $\hat{h}_{\mathrm{ru}}^{(ji)}$ is an estimate of the maximum (envelope) of impulse response functions from input i to residual j for a given model uncertainty.

Example 6.10 Ship example (LTI case)
Assume the ship was LTI,

$$y_1(s) = \omega_3(s) + \omega_w(s) + f_\omega(s) = \frac{b}{s - b\eta_1}\delta(s) + \omega_w(s) + f_\omega(s) \tag{6.104}$$
$$y_2(s) = \psi(s) + f_\psi(s) = \frac{1}{s}(\omega_3(s) + \omega_w(s)) + f_\psi(s)$$

the design model was

$$\hat{\omega}_3(s) = \frac{\hat{b}}{s - \hat{b}\hat{\eta}_1}\delta$$
$$\hat{\psi}(s) = \frac{1}{s}\hat{\omega}_3(s)$$

and a residual generator is chosen as

$$r_1(s) = y_1(s) - \hat{\omega}_3(s) = \left(\frac{b}{s - b\eta_1} - \frac{\hat{b}}{s - \hat{b}\hat{\eta}_1} \right) \delta(s) + \omega_w(s) + f_\omega(s)$$
$$r_2(s) = \frac{\tau}{1 + s\tau}(s y_2(s) - y_1(s)) = \frac{s\tau}{1 + s\tau} f_\psi(s) - \frac{\tau}{1 + s\tau} f_\omega(s).$$

With no faults

$$\|r_1(\mathrm{j}\omega)\|_2 \leq \left\| \frac{b}{j\omega - b\eta_1} - \frac{\hat{b}}{j\omega - \hat{b}\hat{\eta}_1} \right\|_\infty \|\delta(\mathrm{j}\omega)\|_2 + \|\omega_w(\mathrm{j}\omega)\|_\infty$$

and

$$\|r_1(\mathrm{j}\omega)\|_2 \leq \left\| \frac{b}{j\omega + b\eta_1} - \frac{\hat{b}}{j\omega + \hat{b}\hat{\eta}_1} \right\|_\infty \|\delta(t)\|_2 + \|\omega_w(\mathrm{j}\omega)\|_\infty \tag{6.105}$$

$$\Phi_1(t) = \left\| \frac{b}{j\omega - b\eta_1} - \frac{\hat{j\omega}}{j\omega - \hat{b}\hat{\eta}_1} \right\|_\infty \|\delta(t)\|_2 + \|\omega_w(\mathrm{j}\omega)\|_\infty \tag{6.106}$$

$$\text{with } \|\omega_w(\mathrm{j}\omega)\|_\infty \leq \beta_\mathrm{d},$$

$$\Phi_1(t) = \left| \frac{\hat{\eta}_1 - \eta_1}{\eta_1 \hat{\eta}_1} \right| \|\delta(t)\|_2 + \beta_\mathrm{d} = \beta_u \|\delta(t)\|_2 + \beta_\mathrm{d}. \tag{6.107}$$

In real time, we evaluate

$$\|r(t)\|_2 \leq \beta_u \|\delta(t)\|_2 + \beta_\mathrm{d} \tag{6.108}$$

using Eq. (6.97) as an approximation to the two norm. □

General case. In the general case, if the parity relation is bounded by

$$\varphi(p_j(u_i, y_i, \hat{\theta}_i, \hat{c}_i, t)) \leq \alpha_j(u_i, y_i, t) \wedge 0 < \alpha(u_i, y_i, t) < \infty. \tag{6.109}$$

The threshold function can obviously be chosen as

$$\Phi_j(t) \geq \alpha_j(u_i, y_i, t) \tag{6.110}$$

If more detailed information is available, e.g.

$$\alpha_j(u_i, y_i, t) \leq \beta_0 + \sum_{i=1}^{k} \beta_{ji} |u_i| \tag{6.111}$$

such information should be utilised when specifying the threshold, in this case as

$$\Phi_j(t) = \beta_0 + \sum_{i=1}^{k} \beta_{ji} |u_i| \,. \tag{6.112}$$

Return to normal. The above procedure tested for the change H_0 to H_1. When a fault has been detected, $\varphi(r_j(t)) \geq \Phi_j(t) \Rightarrow H^{(j)} = H_1$, change to normal is usually made with a hysteresis, $\gamma : \varphi(r_j(t)) < \gamma\Phi_j(t) \Rightarrow H^{(j)} = H_0$. A common choice of hysteresis is $\gamma \subset [0.5, 0.8]$.

If a fault is only weakly detectable in residual j, but strongly detectable in other residuals, $\forall j : \varphi(r_j(t)) < \gamma\Phi_j(t) \Rightarrow H^{(j)} = H_0$ should be used.

It is obvious that simulation and tests in the real environment some engineering judgement need be employed before good choices can be made of the time-varying threshold function $\Phi_j(t)$ and of the hysteresis γ.

This leads to algorithms for deterministic change detection,

Algorithm 6.5 *Test against time-varying threshold*

Given: A residual $r_j = p_j(u_i, y_i, \hat{\theta}_i, \hat{c}_i, t)$ and the object for diagnosis assumed in the no-fault condition.

1. Determine a test function $\varphi(r(t))$ according to Eqs. (6.96) to (6.99).
2. Determine a threshold function $\Phi_j(t)$ for the LTI case according to Eq. (6.102), for the general case according to Eq. (6.109) or Eq. (6.102) when specific information is available.

Initialise: $H^{(j)} = H_0$.

Do:

1. Calculate $\varphi(r_j(t))$ and $\Phi_j(t)$.
2. If $H^{(j)} = H_0, \forall j$:.
   If $\varphi(r_j(t)) \geq \Phi_j(t)$ set hypothesis to $H^{(j)} = H_1$.
   Else:
   If $\varphi(r_j(t)) < \gamma\ \Phi_j(t)$ for $\forall j$ set hypothesis to $H^{(j)} = H_0$.

Example 6.11 Time-varying threshold for ship

Let the ship's true constraints be:

$$\begin{aligned} c_1 &: \dot{\omega}_3 = b\eta_1\omega_3 + b\eta_3\omega_3^3 + b\delta \\ c_2 &: \dot{\psi} = \omega_3 + \omega_w \\ m_1 &: y_1 = \dot{\psi} \\ m_2 &: y_2 = \psi \end{aligned} \tag{6.113}$$

And let a model used for design be

$$\begin{aligned} \hat{c}_1 &: \dot{\omega}_3 = \hat{b}\hat{\eta}_1\omega_3 + \hat{b}\delta \\ \hat{c}_2 &: \dot{\psi} = \omega_3 \\ m_1 &: y_1 = \dot{\psi} \\ m_2 &: y_2 = \psi \end{aligned} \tag{6.114}$$

Using the model for design, a residual generator is suggested as

$$\begin{aligned} r_1 &= \tfrac{\mathrm{d}}{\mathrm{d}t} y_1 - \tfrac{\mathrm{d}}{\mathrm{d}t} \hat{y}_1 \\ r_2 &= \tfrac{\mathrm{d}}{\mathrm{d}t} y_2 - y_1 \end{aligned} \tag{6.115}$$

then, the real residual will vary with input and

$$\begin{aligned} r_1(t) &= (b\eta_1 - \hat{b}\hat{\eta}_1)\omega_3 + b\eta_3\omega_3^3 + (b - \hat{b})\delta(t) + \tfrac{\mathrm{d}}{\mathrm{d}t}\omega_w(t) \\ r_2(t) &= 0 \end{aligned}$$

$$\begin{aligned} |r_1(t)| \;&\leq \left|b\eta_1 - \hat{b}\hat{\eta}_1\right| |y_1| + |b\eta_3| \left|y_1^3\right| + \left|b - \hat{b}\right| |\delta| + \left|\frac{\mathrm{d}}{\mathrm{d}t}\omega_w(t)\right|_{\sup} \\ &\leq \beta_1 |y_1| + \beta_3 \left|y_1^3\right| + \alpha_1 |\delta| + \beta_{\mathrm{d}} \leq \alpha_2 |\delta| + \beta_{\mathrm{d}}. \; \square \end{aligned}$$

6.7 Exercises

Exercise 6.1 Residual generator for position actuator
Consider the system in Fig. 3.8 and parameters given in Exercise 3.3. There is no measurement noise in the exercise.

1. Implement a candidate residual generator. Use the parity equations

$$e(s) = y_m(s) - \hat{y}(s),$$

where

$$\hat{y}_1(s) = \frac{1}{sI_{\mathrm{tot}} + \alpha} \left(k_q \eta \, i_m(s)\right),$$

and

$$\hat{y}_2(s) = \frac{1}{Ns} \left(n_m(s)\right).$$

Investigate the properties of these potential residual generators by applying step changes on either of the faults.
2. Consider, further the possible fault in the shaft speed sensor. Investigate experimentally whether all three faults can be detected and isolated.
3. Derive the transfer function matrix $\boldsymbol{H}_{\mathrm{yf}}(s)$ and use this to explain the observations. $\square$

Exercise 6.2 Residual generation using the parity space approach
This exercise deals with residual generator for the industrial actuator. Refer to Fig. 3.7. The disturbance is Q_1. The input is i_{com}. The measurements are n_m and θ_m.

1. Determine the transfer function matrices $\boldsymbol{H}_{\mathrm{yu}}(s)$ and $\boldsymbol{H}_{\mathrm{yd}}(s)$.
2. Write the transfer function matrix

$$\boldsymbol{H}(s) = \begin{pmatrix} \boldsymbol{H}_{\mathrm{yu}}(s) & \boldsymbol{H}_{\mathrm{yd}}(s) \\ \boldsymbol{I} & \boldsymbol{O} \end{pmatrix}.$$

3. Write $\boldsymbol{H}(s)$ in the form

$$\boldsymbol{H}(s) = \frac{1}{h(s)}\tilde{\boldsymbol{H}}(s)$$

   where $\tilde{H}$ is a polynomial matrix.
4. Determine the rank of $\tilde{\boldsymbol{H}}(s)$.
5. Determine the nullspace of $\tilde{\boldsymbol{H}}^{\mathrm{T}}(s)$.
6. From the nullspace of $\tilde{\boldsymbol{H}}^{\mathrm{T}}(s)$, determine residual generator(s)

$$\boldsymbol{r}(s) = \boldsymbol{V}_{\mathrm{ru}}(s)\boldsymbol{u}(s) + \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{y}(s)$$

   that make the residual independent of unknown input. Verify this property by showing that

$$\boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yd}} = \boldsymbol{O}.$$

7. Determine the transfer function between $\boldsymbol{f}(s)$ and $\boldsymbol{r}(s)$ $\boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yf}}(s)$ and test which of the three faults f_i, f_n and f_θ are detectable. □

Exercise 6.3 Residual generation for single-axis satellite
In continuation of Exercise 5.3 this exercise deals with residual generation for the single-axis satellite.

A state-space model for the single axis is given by

$$\begin{aligned} \dot{x}_1 &= \tfrac{1}{I}(k_1u_1 + k_2u_2 + w_0) \\ \dot{x}_2 &= x_1 \\ y_1 &= x_1 + f_1 \\ y_2 &= x_2 + f_2 \\ y_3 &= x_2 + f_3 \\ y_4 &= k_1u_1 + f_4 \\ y_5 &= k_2u_2 + f_5, \end{aligned}$$

where x_1 is the angular velocity, x_2 the angle of the satellite and nominal parameters are

$$\begin{aligned} I &= 14.33\ \mathrm{kg\,m^2} \\ k_1 &= k_2 = 0.5. \end{aligned}$$

There are two input signals, u_1 and u_2 to actuators 1 and 2, respectively. There is one unknown input d. The magnitude of d is not known prior to the launch of the satellite, but it is known that d is constant over time.

There are five measurements: y_1 measures the state x_1, y_2 and y_3 measure the state x_2. y_4 measures the actual torque from actuator 1, y_5 measures the actual torque from actuator 2.

1. Determine the transfer function matrices $\boldsymbol{H}_{\mathrm{yu}}(s)$ and $\boldsymbol{H}_{\mathrm{yd}}(s)$.
2. Determine the transfer function matrix

$$\boldsymbol{H}(s) = \begin{pmatrix} \boldsymbol{H}_{\mathrm{yu}}(s) & \boldsymbol{H}_{\mathrm{yd}}(s) \\ \boldsymbol{I} & \boldsymbol{O} \end{pmatrix}.$$

3. Write $\boldsymbol{H}(s)$ in the form

$$\boldsymbol{H}(s) = \frac{1}{h(s)} \tilde{\boldsymbol{H}}(s)$$

   where $\tilde{\boldsymbol{H}}$ is a polynomial matrix.
4. Determine the rank of $\tilde{\boldsymbol{H}}(s)$.
5. How many independent residual generators can be expected that are independent of input $\boldsymbol{u}(s)$ and of disturbances $d(s)$.
6. Find the left nullspace of $\tilde{\boldsymbol{H}}(s)$.
7. Determine a residual generator based on the nullspace. □

Exercise 6.4 Properties of residual generators for single-axis satellite
This exercise is a continuation of Exercise 6.3.

1. Determine the response of the residual vector to the additive faults on y_1 to y_5 by calculating

$$\boldsymbol{r}(s) = \boldsymbol{V}_{\mathrm{ry}}(s)\boldsymbol{H}_{\mathrm{yf}}(s)\boldsymbol{f}(s).$$

2. Determine which of the above faults are detectable and which are strongly detectable.
3. Determine which of above faults can be isolated.
   As pure differentiation or integration are not feasible in the presence of measurement noise, a filter is applied on one of the residuals. Investigate the features of two proposed residual generators. Both have the form

$$\begin{aligned} r_{12}(s) &= \frac{1}{s+\alpha} y_1(s) - \frac{s}{s+\alpha} y_2(s) \\ r_{23}(s) &= y_2(s) - y_3(s). \end{aligned}$$

   Version (a) has $\alpha = 0.01$, version (b) has $\alpha = 10$.
4. Discuss the properties of the two residual generators (detectability, strong detectability, isolability). Apply a fixed threshold on either set of generators to detect if a fault is present and verify your results by simulation. □

Exercise 6.5 Residual generator design - optimisation method
This exercise addresses the position servo from Exercise 3.2, (Fig. 5.36 on p. 207). The exercise is to design residual generators based on the standard setup used in robust control. It is assumed that only a single fault can appear at a time.

1. Formulate the FDI problem for the system as a standard problem. Identify the matrices that need to be selected in connection with the design.
2. Design residual generators for fault detection using the standard setup and standard design methods.
3. Design a residual generator for fault isolation and fault estimation using the standard setup and standard design methods. □

Exercise 6.6 Residual generator with an explicit specification
This exercise addresses the position servo from Exercise 3.2, (Fig. 5.37 on p. 207). Assume the load possess a dominant disturbance above 0.5 rad/s.

1. Formulate a specification $\boldsymbol{H}_{\mathrm{zd}}(s)$ and $\boldsymbol{H}_{\mathrm{zf}}(s)$ for the design.
2. Formulate the fault detection and isolation problem for the system as a standard problem. Identify the matrices that need to be selected in connection with the design.
3. Design residual generators for fault detection using the standard setup and standard design methods.
4. Design a residual generator for fault isolation and fault estimation using the standard setup and standard design methods. □

Exercise 6.7 Residual generation by a Luenberger observer
Consider the following linear time-invariant system

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -2 & -3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$$
$$y = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + f_3$$

where $\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ is the state, $\boldsymbol{f} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}$ is the fault vector ($f = 0 \iff$ normal operation) and y is the measured output.

1. Define the parameters k_1 and k_2 of a Luenberger observer

$$\begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -2 & -3 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} + \begin{pmatrix} k_1 \\ k_2 \end{pmatrix} (y - \hat{y})$$
$$\hat{y} = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$$

   which has the following property: in the absence of faults, the estimation error

$$\begin{pmatrix} z_1 - x_1 \\ z_2 - x_2 \end{pmatrix}$$

   converges to zero with a dynamics associated with the two eigenvalues $\lambda_1 = \lambda_2 = -5$.
2. Determine the transfer function between the residual $r = y - \hat{y}$ and the fault vector $\boldsymbol{f}$ under the form

$$r = G_1(s) f_1 + G_2(s) f_2 + G_3(s) f_3. \square \tag{6.116}$$

Exercise 6.8 Static and dynamical redundancy
Consider the following system composed of four components: process, sensor 1, sensor 2, sensor 3 (see Fig. 6.7). It is assumed that it can be described by the following linear time-invariant model

$$\begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -2 & -3 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t) + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f_1(t) \\ f_2(t) \\ f_3(t) \\ f_4(t) \end{pmatrix}$$

$$\begin{pmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} f_1(t) \\ f_2(t) \\ f_3(t) \\ f_4(t) \end{pmatrix}$$

where

$$\boldsymbol{x}(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}$$

characterises the state of the process component, $u(t)$ is the control input,

$$\boldsymbol{y}(t) = \begin{pmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{pmatrix}$$

is the vector of all measurements and

$$\boldsymbol{f}(t) = \begin{pmatrix} f_1(t) \\ f_2(t) \\ f_3(t) \\ f_4(t) \end{pmatrix}$$

is the fault vector.

1. What is the association between the faults f_i, $(i = 1, 2, 3, 4)$ and the system components.
2. Is the state $\boldsymbol{x}(t)$ observable?
3. Is there any static redundancy in this system? What are the detectable or isolable faults?
4. Assume that during a given period of time, only sensor y_1 is operational (for example, y_2 and y_3 are disconnected for maintenance). Is it still possible to estimate the state or to detect and isolate the faults? □

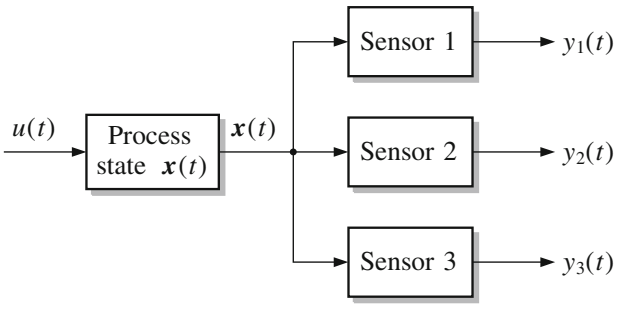


Fig. 6.7 System with three sensors

6.8 Bibliographical Notes

Parity relations. The parity relations that were initially studied in [65, 125, 205] are functions of a sliding window of the most recent sensor output and actuator input values. The idea used to develop parity relations in the time domain was extended to the frequency domain. This has lead to the generalised parity relations [376] which do not necessarily involve only the data of a sliding window. Later, this distinction between parity relations and generalised parity relations tended to disappear. The presentation given here is in the line of [256]. A way to assure causality and stability of a filter involving the inverse of a transfer matrix can be found in [177]. A thorough study of the parity space approach to residual generation can also be found in [124]. The equivalence between observer-based and parity space approaches is developed in [225] for instance. Further results on the design of residual generators in the frequency domain can be found in [177].

Analytic redundancy relations. The systematic computation of analytical redundancy relations for polynomial nonlinear models was developed in [326, 409]. Details on elimination theory may be found in [82, 309]. For Gröbner bases used in Buchberger's algorithm [44], details and definitions can be found in [71]. The reader is referred to [129] for details as the use of characteristic sets.

Diagnosis as an optimisation problem. A comprehensive reference to fault diagnosis treated as an optimisation problem is [228]. Earlier research results, that relate to the presentation in this book, were published in [103, 105]. The book [64] has a chapter devoted to this subject. The design of fault diagnosis filters using the standard setup presented in this book originates in [249, 343].

Time-varying thresholds. More information on threshold selection can be found in the classical presentation of this subject of [92] and, for later results, in [80, 161].

Observer-based residual generation. The observer-based approach for residual generation has been the object of numerous studies. The reader can refer to the book [265] for an introductory treatment and references on early works in this area. Reference [64] provides more recent developments on the topic as well as a very complete list of references.

Active detection and isolation. Active fault detection and isolation has been briefly mentioned in this chapter. The problem of determining an optimal input signal to distinguish between different models (representing healthy and faulty modes) for a given process has been the object of a thorough study in [54, 239]. Reference [242] suggested novel ways to achieve active fault isolation while a plant is running. Recently, [308] has proposed a method for input design that guarantees fault diagnosis using zonotopes.

Chapter 7
Fault Diagnosis of Stochastic Systems

Abstract Solutions to the fault detection, estimation and isolation problems are presented when the model of the supervised system is a linear stochastic continuous-variable system. Faults are modelled as additive signals. The resulting diagnosis system is separated in two parts: a residual generator based on Kalman filters, and a decision system based on stochastic change detection/isolation algorithms. The link between these two parts is the object of particular attention.

7.1 Introduction

In the previous chapter, measurement noise and process disturbances could be introduced as inputs with bounded energy and handled through the optimization-based design. This might, however, lead to a conservative design. An alternative way to account for measurement noise is to resort to stochastic sequences. Such random inputs can also be used to represent various disturbances like wind turbulence affecting the operation of wind turbines and airplanes notably. As dealing with random inputs is easier in a discrete-time framework than in a continuous-time framework, a discrete-time model of the supervised process will be considered in this chapter. Such a model can be deduced from a linear stochastic differential equation via the procedure recalled in the appendix on random variables and stochastic processes.

The fault detection, isolation and/or estimation systems based on stochastic models keep the same structure as in the deterministic case (see Fig. 6.1 in Chap. 6). They are thus made of a residual generator and a decision system. As measurement noise typically affects all sensor outputs, perfect decoupling of the residual with respect to these signals cannot be achieved. Residuals are thus stochastic signals and, to process them properly, statistical change detection/isolation algorithms are needed.

There is a well-developed theoretical framework for statistical change detection/isolation, and we will provide an introduction to this topic before applying it to fault diagnosis. The presentation aims at providing detailed algorithms for implementing the methods and for tuning the design parameters according to typical specifications. The latter consist of false alarm and missed alarm probabilities,

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8_7

for non-sequential tests, and of mean detection delay and mean time between false alarms, for online change detection algorithms.

The chapter is organised as follows. Section 7.2 presents statistical change detection/isolation algorithms and their tuning, with particular emphasis on methods for detecting a change in the mean of a normally distributed independent sequence. Residual generation on the basis of a linear stochastic model is addressed in Sect. 7.3. Next, the statistical properties of the residuals are investigated and appropriate residual evaluation methods are developed by resorting to the statistical change detection/isolation algorithms presented in Sect. 7.2.

7.2 Change Detection Algorithms

When the residual generator is designed on the basis of a linear stochastic model, residual evaluation reduces, under suitable hypotheses, to the problem of detecting a change in the mean of a normally distributed random sequence. This can be achieved by sequential change detection algorithms. Therefore, this topic is considered before addressing fault detection, isolation and estimation in the case of additive faults.

7.2.1 Sequential Change Detection: The Scalar Case

Introduction. The sequential change detection algorithms are first derived in the simple case of processing a sequence of independent random variables with probability density function depending on a scalar parameter θ. The situation where θ is the mean of a Gaussian distribution is used to illustrate the theory, since this is the problem often encountered in residual evaluation. As the sequential algorithms will be used to process residuals, the above theory is generalised to the case of detecting changes in the mean of sequences of Gaussian vectors, which is done in a subsequent paragraph.

Problem statement. Consider a sequence of independent random variables $z(i)$, $i = 1, 2, \ldots$, with probability density function $p_\theta(z)$ depending upon one scalar parameter θ. Before an unknown change time, k_0, θ is equal to θ_0. At time k_0, it changes to $\theta = \theta_1 \neq \theta_0$. The change detection problem is then threefold:

- detect whether the condition is normal, $\theta = \theta_0$, or the parameter θ has changed to θ_1,
- estimate the change time,
- estimate the value of the change in the parameter if the change magnitude is unknown.

The detection problem in the first bullet point is referred to as distinguishing between two hypotheses: $\mathcal{H}_0$ - the nominal case, $\mathcal{H}_1$ - a change has taken place. The

condition under $\mathcal{H}_0$ are assumed to be known so that the parameter θ_0 is known. Two situations are then considered for θ_1, namely θ_1 *known* and θ_1 *unknown*. This will be shown to lead to two different classes of change detection algorithms:

-detecting a known change leads to the cumulative sum (CUSUM) algorithm.
-detecting an unknown change leads to the generalised likelihood ratio (GLR) algorithm.

Both the CUSUM and the GLR algorithms rely on the Neyman–Pearson's approach to detection. It is thus assumed that no prior knowledge is available about the probability for the system to be in the condition $\mathcal{H}_0 : p(z) = p_{\theta_0}(z)$ or $\mathcal{H}_1 : p(z) = p_{\theta_1}(z)$. In other words no prior information on the probability distribution of the change time is assumed. A fundamental measure to investigate whether data correspond to a probability density function $p_{\theta_0}(z)$ or $p_{\theta_1}(z)$ is the ratio between the two probability density functions. Since probability distributions are often assumed to be Gaussian (Appendix B), the logarithm of this probability ratio gives very convenient calculations. The *log-likelihood ratio* of an observation z, is defined as:

$$s(z) = \ln \frac{p_{\theta_1}(z)}{p_{\theta_0}(z)}. \tag{7.1}$$

The name comes from the fact that the likelihood function of the observation z is by definition equal to the probability density $p_\theta(z)$ of the underlying random variable evaluated at z. Given the observation z, the log-likelihood function is thus a deterministic function of θ.

The log-likelihood ratio has the following fundamental statistical property:

$$E_{\theta_0}(s) = \int_{-\infty}^{\infty} s(z)\, p_{\theta_0}(z)\, \mathrm{d}z < 0, \tag{7.2}$$

$$E_{\theta_1}(s) = \int_{-\infty}^{\infty} s(z)\, p_{\theta_1}(z)\, \mathrm{d}z > 0. \tag{7.3}$$

E_{θ_0} (E_{θ_1}) denotes expectation of $s(z)$ under the distribution associated to $p_{\theta_0}(z)$ ($p_{\theta_1}(z)$). This property can be easily understood from the following example. Assume that $p_\theta(z)$ is a Gaussian probability density function and that the parameter θ is the mean of this distribution, which will be denoted μ.

Consider Fig. 7.1. When the random variable z has $p_{\mu_0}(z)$ ($p_{\mu_1}(z)$) as probability density function, its realisations are most often in the "neighbourhood" of μ_0 (μ_1). Take the realisation z_1 for instance. Clearly $\frac{p_{\mu_1}(z_1)}{p_{\mu_0}(z_1)} < 1$. As z_1 is most probably obtained when the random variable z has $p_{\mu_0}(z)$ as probability density function, this illustrates that the logarithm of $\frac{p_{\mu_1}(z)}{p_{\mu_0}(z)}$ is on the average negative when z has $p_{\mu_0}(z)$ as probability density function. The property described by (7.2), (7.3) is exploited in the next section to provide an intuitive derivation of the CUSUM algorithm.

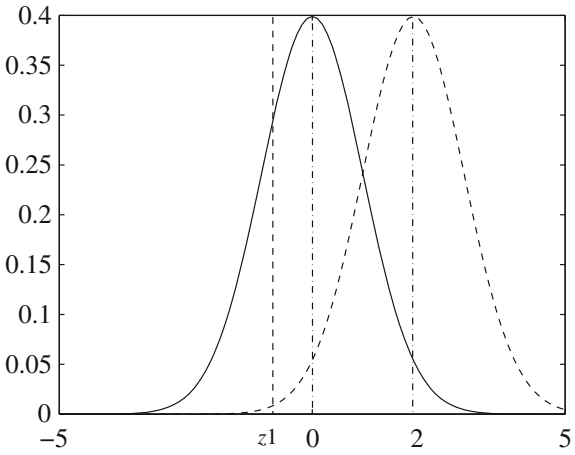


Fig. 7.1 Two Gaussian probability density functions with mean $\mu_0 = 0$ and $\mu_1 = 2$, and with the same variance $\sigma^2 = 1$

7.2.2 *Detection of a* Known *Change - The CUSUM Algorithm*

The problem stated in the previous section, with θ_1 known, is addressed. Consider the cumulative sum:

$$\text{Cumulative sum:} \quad S(k) = \sum_{i=1}^{k} s(z(i)) = \sum_{i=1}^{k} \ln \frac{p_{\theta_1}(z(i))}{p_{\theta_0}(z(i))}. \tag{7.4}$$

In this expression, and in the remaining part of this section, k denotes the present time instant. From (7.2) and (7.3), $S(k)$ is expected to exhibit a negative drift before change, and a positive drift after change. This is illustrated in Figs. 7.2 and 7.3.

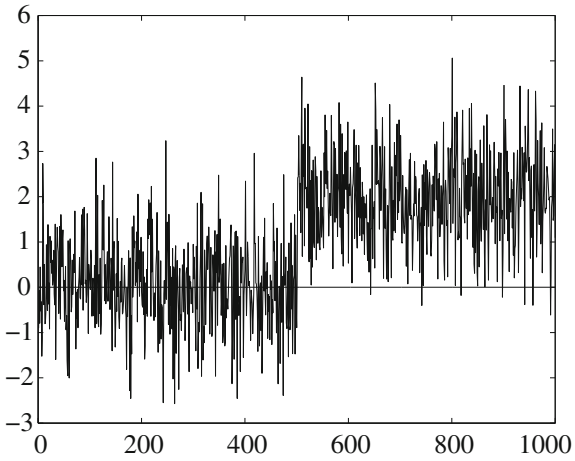


Fig. 7.2 Realisation of a sequence of independent random variables with distributions depicted in Fig. 7.1. Time on the x-axis is expressed in number of samples

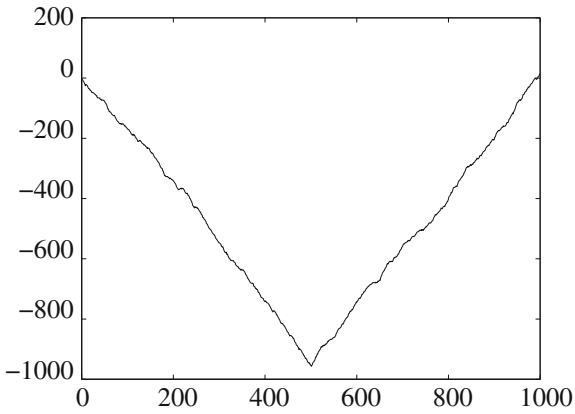


Fig. 7.3 Evolution of $S(k)$ for the sequence of Fig. 7.2, as a function of time in number of samples

In Fig. 7.2, a realisation of a sequence of independent random variables with distribution $p_{\theta_0}(z)$ before $k = 500$ and $p_{\theta_1}(z)$ after $k = 500$ is depicted. Here $p_{\theta_0}(z)$ and $p_{\theta_1}(z)$ correspond to the Gaussian distributions $p_{\mu_0}(z)$ and $p_{\mu_1}(z)$ of Fig. 7.1. Figure 7.3 gives the evolution of $S(k)$, which behaves as expected. The difference between $S(k)$ and the minimum value of $S(j)$, $1 \leq j \leq k$ yields relevant information on the change. Hence the following decision function $g(k)$ is considered

$$g(k) = S(k) - m(k) \tag{7.5}$$

with $m(k) = \min_{1 \leq j \leq k} S(j)$. The stopping time (also called alarm time), k_a is the time instant at which $g(k)$ crosses a user-defined positive threshold h. The fault occurrence time, k_0, can be estimated as the time instant $\hat{k}_0$ at which $S(k)$ has changed from negative to positive slope. It is formally expressed by

$$\hat{k}_0 = \text{argmin}_{1 \leq j \leq k_a} S(j).$$

The expression of the cumulative sum (7.4) can easily be computed for the distributions considered in Fig. 7.1, as shown in the example below.

Example 7.1 Change in the mean of a Gaussian sequence
Remember that the Gaussian probability density function for a random variable with mean μ and variance σ is

$$p_\mu(z) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right). \tag{7.6}$$

The resulting likelihood ratio for detecting a change in the mean from μ_0 to μ_1 is

$$\frac{p_{\mu_1}(z)}{p_{\mu_0}(z)} = \exp\left(-\frac{(z-\mu_1)^2}{2\sigma^2} + \frac{(z-\mu_0)^2}{2\sigma^2}\right).$$

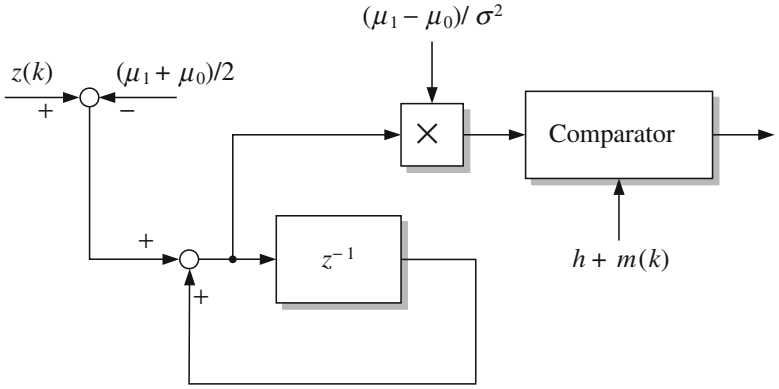


Fig. 7.4 Block diagram for the CUSUM test (7.4), (7.5), (7.7)

Hence straightforward computations yield the following expression for the log-likelihood ratio $s(z)$:

$$s(z) = \frac{2(\mu_1 - \mu_0)z + (\mu_0^2 - \mu_1^2)}{2\sigma^2} = \frac{\mu_1 - \mu_0}{\sigma^2}\left(z - \frac{\mu_0 + \mu_1}{2}\right). \tag{7.7}$$

Figure 7.3 has been obtained by substituting (7.7) (with $z = z(i)$) for $s(z(i))$ in (7.4), which yields the algorithm depicted in the block diagram of Fig. 7.4. Note that the signal-to-noise ratio $\frac{\mu_1 - \mu_0}{\sigma}$ appears in (7.7), and it is thus automatically accounted for in the testing procedure.
Alternatively, the situation where the mean of the signal remains constant but its variance changes is considered as the following case:

Example 7.2 Change in the variance
If the variance changes after a fault, the following relation

$$\frac{p_{\sigma_1}(z)}{p_{\sigma_0}(z)} = \frac{\sigma_0}{\sigma_1}\exp\left(-\frac{(z-\mu)^2}{2\sigma_1^2} + \frac{(z-\mu)^2}{2\sigma_0^2}\right)$$

holds and the log-likelihood ratio is

$$s(z) = \ln\frac{\sigma_0}{\sigma_1} + \frac{(z-\mu)^2}{2}\left(\frac{1}{\sigma_0^2} - \frac{1}{\sigma_1^2}\right).$$

Such a variance change may correspond to increased measurement noise, due to a deteriorated connection for instance. □

Remark 7.1 (*Mean and variance of cumulative sum increments*) Important properties of any detector are related to the properties of the test statistics, $g(k)$. For the CUSUM algorithm, and the case of $z(i)$ being a Gaussian sequence with mean μ and variance σ^2, we will later need expressions for the mean and variance of the

individual increments $s(z)$ used to calculate $g(k)$. The mean μ_s and the variance σ_s^2 of the cumulative sum increments (7.7) can be computed in a straightforward way as

$$\mu_s = \frac{\mu_1 - \mu_0}{\sigma^2}\left(\mu - \frac{\mu_1 + \mu_0}{2}\right) \tag{7.8}$$

and

$$\sigma_s^2 = \frac{(\mu_1 - \mu_0)^2}{\sigma^2} \tag{7.9}$$

In particular,

$$\text{when} \quad \mu = \mu_0,\ E_{\mu_0}(z) = \mu_0 \quad \text{and} \quad \mu_s = E_{\mu_0}(s(z)) = -\frac{(\mu_1 - \mu_0)^2}{2\sigma^2},$$

$$\text{when} \quad \mu = \mu_1,\ E_{\mu_1}(z) = \mu_1 \quad \text{and} \quad \mu_s = E_{\mu_1}(s(z)) = \frac{(\mu_1 - \mu_0)^2}{2\sigma^2}. \quad \square$$

The CUSUM algorithm - formal derivation. A more formal derivation of the CUSUM algorithm which is helpful for the subsequent description of the GLR algorithm is now presented. It is called the offline statistical derivation, and it is based on the following reformulation of the problem:

Problem 7.1 (*Offline statistical formulation*) Consider the sequence of independent random variables $z(1), \ldots, z(k)$ with probability density function $p_\theta(z)$ depending on one scalar parameter θ. Choose at time instant k between the hypotheses:

$\mathcal{H}_0 : \theta = \theta_0$ for $1 \leq i \leq k$.
$\mathcal{H}_1 : \theta = \theta_0$ for $1 \leq i \leq k_0 - 1$ and $\theta = \theta_1$ for $k_0 \leq i \leq k$, where the time instant k_0 is unknown.

From classical results in hypothesis testing due to Neyman and Pearson, it is known that tests to decide between $\mathcal{H}_0$ and $\mathcal{H}_1$ that are optimal in some sense are based on the log-likelihood ratio between both hypotheses. As k_0 is unknown, let j be a hypothetical change time. The log-likelihood ratio between $\mathcal{H}_0$ and $\mathcal{H}_1$ with $k_0 = j$ is given as

$$\Lambda_1^k(j) = \frac{\prod_{i=1}^{j-1} p_{\theta_0}(z(i)) \prod_{i=j}^{k} p_{\theta_1}(z(i))}{\prod_{i=1}^{k} p_{\theta_0}(z(i))}. \tag{7.10}$$

The independence between the random variables $z(i), i = 1, \ldots, k$ was used to express $\Lambda_1^k(j)$ in terms of the marginal probability density function $p_\theta(z(i))$. From (7.10), the following cumulative sum of log-likelihood ratios is deduced:

$$S_j^k = \ln \Lambda_1^k(j) = \sum_{i=j}^{k} \ln \frac{p_{\theta_1}(z(i))}{p_{\theta_0}(z(i))}. \tag{7.11}$$

As the change time is unknown, the standard statistical approach consists of replacing it by its maximum likelihood estimate, namely, in looking for the value of j that maximises the numerator in (7.10). This is also the value of j that maximises (7.11). The log-likelihood ratio between $\mathcal{H}_0$ and $\mathcal{H}_1$ is thus estimated by $\max_{1\le j\le k} S_j^k$. The result due to Neyman and Pearson invoked above actually states that the optimal decision function for Problem 7.1 is

$$g(k) = \max_{1\le j\le k} \sum_{i=j}^{k} \ln \frac{p_{\theta_1}(z(i))}{p_{\theta_0}(z(i))} = \max_{1\le j\le k} \sum_{i=j}^{k} s(z(i)) \tag{7.12}$$

and the optimal test consists of the following decisions:

$$\begin{array}{lll} \text{if} & g(k) \le h & \text{accept} \quad \mathcal{H}_0 \\ \text{if} & g(k) > h & \text{accept} \quad \mathcal{H}_1. \end{array} \tag{7.13}$$

The way optimality is understood here involves several concepts. The reader should consult the reference section for precisions on this topic.

When $\mathcal{H}_1$ is accepted, an estimate of the change time is provided by:

$$\hat{k}_0 = \operatorname{argmax}_{\ 1\le j\le k_a} S_j^k,$$

where k_a is the alarm time, namely the value of k for which $g(k)$ crosses the threshold h.

The decision functions (7.5) and (7.12) are identical. Indeed, with reference to Fig. 7.3, $\sum_{i=j}^{k} \ln \frac{p_{\theta_1}(z(i))}{p_{\theta_0}(z(i)}$ is maximum when all the successive likelihood ratios which correspond to a positive slope on average are considered. This is precisely the way (7.5) was obtained.

CUSUM recursive implementation. An efficient way to implement the CUSUM algorithm is to use its recursive form. From (7.5) and Fig. 7.3 or from (7.12), and from the fact that the threshold h is positive, it is seen that only the contributions to the cumulative sum that add up to a positive number must be taken into account in order to determine the decision function. It justifies the following recursive computation of this function:

$$g(k) = \max(0,\ g(k-1) + s(z(k))). \tag{7.14}$$

To obtain an estimate of the fault occurrence time, the number of successive observations for which the decision function remains strictly positive is computed as:

$$N(k) = N(k-1)\, 1_{\{g(k-1)>0\}} + 1, \tag{7.15}$$

where $1_{\{x\}}$ is the indicator of event x, namely, $1_{\{x\}} = 1$ when x is true, and $1_{\{x\}} = 0$ otherwise. An estimate for the fault occurrence time is then given as

$$\hat{k}_0 = k_a - N(k_a), \tag{7.16}$$

where k_a is the stopping or alarm time.

Example 7.1 (cont.) Change in the mean of a Gaussian sequence
Considering again the detection of a change in the mean of a Gaussian sequence, (7.14) together with (7.7) yields:

$$g(k) = \max\left(0,\ g(k-1) + \frac{\mu_1 - \mu_0}{\sigma^2}\left(z(k) - \frac{\mu_0 + \mu_1}{2}\right)\right) \tag{7.17}$$

which must be introduced in the decision logic (7.13). □

Remark 7.2 (*Two-sided CUSUM algorithm*) Quite often, both positive and negative changes in the mean of a Gaussian sequence with mean μ_0 and variance σ^2 have to be detected. Letting β denote the magnitude of this change, the following two-sided CUSUM algorithm can be used for this purpose.

$$g^+(k) = \max\left(0,\ g^+(k-1) + z(k) - \mu_0 - \frac{\beta}{2}\right) \tag{7.18}$$

$$g^-(k) = \max\left(0,\ g^-(k-1) - z(k) + \mu_0 - \frac{\beta}{2}\right). \tag{7.19}$$

An alarm is generated when either $g^+(k)$ or $g^-(k)$ reaches the threshold $\bar{h} = h\sigma^2/\beta$. Note that the factor $\frac{\mu_1 - \mu_0}{\sigma^2}$ that appears in (7.17) has been omitted from the decision functions $g^+(k)$ and $g^-(k)$ in (7.18) and (7.19). Instead equivalently, it is now included in the threshold $\bar{h}$. The expression for $g^-(k)$ is deduced from (7.17) by looking for a positive change in the mean of the sequence $-z(i)$, $i = 1, 2, \ldots$ □

7.2.3 Detection Properties for the CUSUM Algorithm

In this section, the focus is on the case of a change in the mean, μ, of a Gaussian sequence.

Parameters of the CUSUM algorithm. Normally, the data associated to hypothesis $\mathcal{H}_0$ correspond to a fault free working mode. Hence parameter μ_0 can be estimated from a set of experimental data, $\mathcal{Z}_0 = \{z_0(1), \ldots, z_0(N_0)\}$, obtained in the absence of fault by taking the empirical mean of these data. The variance σ^2 can also be estimated in this way. The estimates are denoted $\hat{\mu}_0$ or $\hat{\sigma}^2$, respectively.

There are thus two design parameters left in the CUSUM algorithm, h and μ_1. Indeed, although the algorithm was derived under the hypothesis that μ_1 is known, this is seldom the case in practice. Nevertheless, the algorithm can be useful even

when μ_1 is replaced by an approximate value. These parameters can be determined to meet specifications in terms of mean delay for detection and mean time between false alarms, as explained below.

Average Run Length (ARL) for CUSUM algorithm. Exact computation of the mean delay for detection and the mean time between false alarms is involved, but approximate expressions and bounds are available in the case of the detection of a change in the mean of a Gaussian sequence. Both quantities can be determined from the *average run length* (ARL) function defined as

$$L(\mu) = E_\mu\,(k_\mathrm{a})$$

which is thus the expected value of the alarm time instant of the CUSUM algorithm when the data sequence is normally distributed with mean μ and variance σ^2. It is a function of the mean μ. When $\mu = \mu_0$ (data recorded in healthy conditions), the value of the ARL function $L(\mu_0)$ is equal to the mean time between false alarms, $\bar{T}$. On the other hand, $L(\mu_1)$ gives the mean delay for detection, $\bar{\tau}$. An approximation for the ARL function is given by the following expression: ([11], p. 219)

$$\hat{L}(\mu) = \left(\exp\left[-2\left(\frac{\mu_s\, h}{\sigma_s^2} + 1.166\frac{\mu_s}{\sigma_s}\right)\right] - 1 + 2\left(\frac{\mu_s\, h}{\sigma_s^2} + 1.166\frac{\mu_s}{\sigma_s}\right)\right)\left(\frac{\sigma_s^2}{2\mu_s^2}\right) \tag{7.20}$$

where $\mu_s \neq 0$ is linked to μ by

$$\mu_s = \frac{\mu_1 - \mu_0}{\sigma^2}\left(\mu - \frac{\mu_1 + \mu_0}{2}\right). \tag{7.21}$$

μ_s and σ_s are the mean and the standard deviation of the increments of the cumulative sum, respectively, as computed in Remark 7.1.

Hence the mean time for detection can be estimated as

$$\hat{\bar{\tau}} = \hat{L}\left(\frac{(\mu_1 - \mu_0)^2}{2\sigma^2}\right) = \hat{L}\left(\frac{\beta^2}{2\sigma^2}\right), \tag{7.22}$$

where $\beta = \mu_1 - \mu_0$ and the estimated mean time between false alarms is obtained as

$$\hat{\bar{T}} = \hat{L}\left(-\frac{(\mu_1 - \mu_0)^2}{2\sigma^2}\right) = \hat{L}\left(-\frac{\beta^2}{2\sigma^2}\right). \tag{7.23}$$

These expressions will be used for tuning the parameters of the CUSUM algorithm.

Tuning of CUSUM parameters. Two situations can be distinguished.

1. μ_1 and either an acceptable mean detection delay, $\bar{\tau}_{acc}$, or an acceptable mean time between false alarms, $\bar{T}_{acc}$, are given. The tuning then amounts to determining h.

2. Acceptable values for the mean detection delay and a lower bound for the mean time between false alarms are given. The tuning then amounts to determining μ_1 and h.

They are successively considered below.

A user specified value for μ_1 (or equivalently β) can correspond to the most likely magnitude of the change, or it can be deduced from a minimum value of the change for which one wishes the algorithm to generate an alarm. In the latter case, if we let $\beta_{\min}$ denote this value. It is then advisable to choose $\mu_1 = \hat{\mu}_0 + 2\beta_{\min}$. Indeed, let $p_{\hat{\mu}_0}(z)$ and $p_{\hat{\mu}_0+2\beta_{\min}}(z)$ denote the Gaussian probability density functions with respective mean $\hat{\mu}_0$ and $\hat{\mu}_0 + 2\beta_{\min}$ and with variance σ^2. It is easy to check that $p_{\hat{\mu}_0}(z) = p_{\hat{\mu}_0+2\beta_{\min}}(z)$ is achieved for $z = \hat{\mu}_0 + \beta_{\min}$. Thus any sequence of values of z greater than $\hat{\mu}_0 + \beta_{\min}$ on average will yield a sequence of positive log-likelihood ratio $\ln \frac{p_{\hat{\mu}_0+2\beta_{\min}}(z)}{p_{\hat{\mu}_0}(z)}$ on average, and an alarm will be triggered after some time for such a sequence.

Once μ_1 is fixed, a simple way to determine the test threshold from (7.22) or (7.23) is to plot $\hat{\bar{\tau}}$ and $\hat{\bar{T}}$ as a function of h. To this end, considering (7.22) for instance, $\frac{(\mu_1-\hat{\mu}_0)^2}{2\hat{\sigma}^2}$ is substituted for μ_s and $\frac{(\mu_1-\hat{\mu}_0)^2}{\hat{\sigma}^2}$ is substituted for σ_s^2 in (7.20). Knowing the acceptable value for $\hat{\bar{\tau}}$ or $\hat{\bar{T}}$, one then determines from the plot an appropriate value for h.

In the second situation, $\bar{\tau}_{acc}$ and $\bar{T}_{acc}$ can be substituted for $\hat{\bar{\tau}}$ and $\hat{\bar{T}}$ in (7.22) and (7.23) respectively, as well as the estimates, $\hat{\mu}_0$ and $\hat{\sigma}^2$. This yields a system of two equations in two unknowns, h and μ_1. The solution provides the lowest change magnitude $(\mu_1 - \mu_0)$ that can be detected while meeting the specifications.

A qualitative analysis of the effect of an error on μ_1 is provided in the following remark:

Remark 7.3 (*Effect of an error on* μ_1) The objective of this remark is to illustrate that the CUSUM algorithm for detection of a change in the mean of a Gaussian sequence can detect changes even when μ_1 is overestimated. To this end, let us consider Fig. 7.5.

In the left-hand figure, the density functions represented by continuous lines correspond to the actual data, which have mean $\mu_0 = 0$ before the change and mean $\mu_1 = 2$ after the change. Figure 7.2 represents a data sequence which was generated from these density functions. The evolution of the recursive CUSUM decision function tuned with $\mu_0 = 0$ and $\mu_1 = 2$ obtained by processing the data of Fig. 7.2 is the upper line in Fig. 7.5b. Let us now process the same data with a CUSUM algorithm tuned with a mean value after change equal to 3 (instead of 2). The resulting decision function is represented by the lower line in Fig. 7.5b. One notices that, when the value of μ_1 in the function $g(k)$ is higher than the real μ, the decision function still increases on average upon occurrence of a change, however, the slope of the decision function is lower than with the correct value of μ_1. To understand this phenomenon, let us look again at Fig. 7.5a, where the Gaussian density function with mean equal to 3 is plotted with a dash-dotted line. Let $p_0(z)$, $p_2(z)$ and $p_3(z)$ denote the

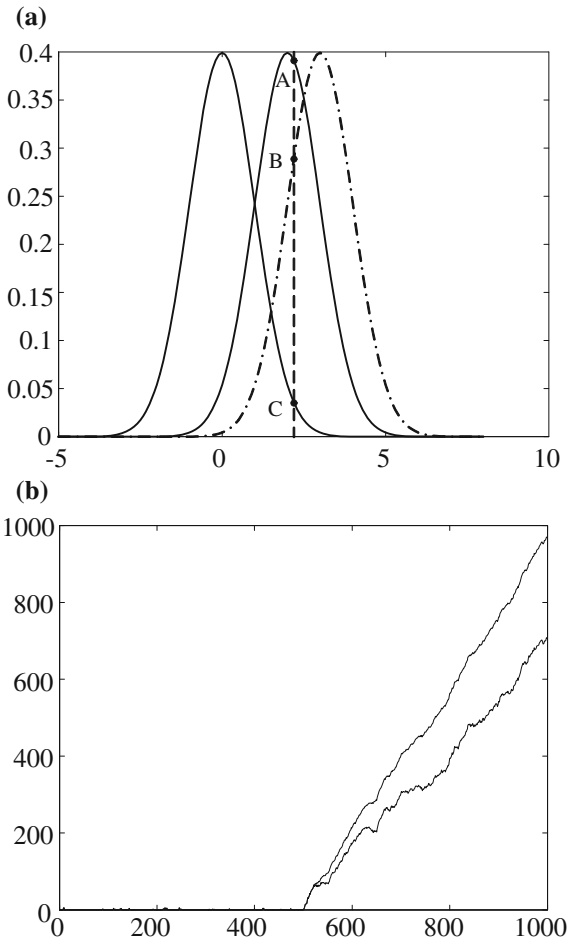


Fig. 7.5 **a** Gaussian probability density functions with actual (*continuous line*) and overestimated means (*dash-dotted line*), **b** evolution of the recursive CUSUM decision functions computed with the exact (*continuous line*) and approximated likelihood ratios (*dash-dotted line*) for the data sequence of Fig. 7.2

density functions with mean 0, 2 and 3, respectively. After the change in the mean, a typical data sample from the actual data sequence, says $\tilde{z}$ will have a value in the neighbourhood of 2. The associated values of the density functions are represented by the points A ($p_2(\tilde{z})$), B($p_3(\tilde{z})$) or C ($p_0(\tilde{z})$), respectively. The contribution to the CUSUM decision function associated to $\tilde{z}$ is equal to $\frac{p_2(\tilde{z})}{p_0(\tilde{z})}$ when the correct tuning is used, and to $\frac{p_3(\tilde{z})}{p_0(\tilde{z})}$ when μ_1 is overestimated. Both values are clearly larger than one, but $\frac{p_3(\tilde{z})}{p_0(\tilde{z})} < \frac{p_2(\tilde{z})}{p_0(\tilde{z})}$ which explains the lower slope of the CUSUM decision function when μ_1 is overestimated. □

The configuration and the implementation of the CUSUM algorithm to detect changes in the mean of a Gaussian sequence can be summarised as follows if the first situation is considered for the specifications:

Algorithm 7.1 *CUSUM algorithm for detection of a change in the mean of a Gaussian sequence*

Given: A set of experimental data $\mathcal{Z}_0$, a change magnitude β, and a specified mean time for detection, $\bar{\tau}_{acc}$, or a specified mean time between false alarms, $\bar{T}_{acc}$.

Initialisation:
1. Determine $\hat{\mu}_0$ and $\hat{\sigma}^2$ from $\mathcal{Z}_0$
2. Choose h to meet either $\bar{\tau}_{acc}$ or $\bar{T}_{acc}$ using (7.22) or (7.23).

At each sample time:
1. Acquire the new data $z(k)$
2. Compute $g(k)$ by (7.17) and $N(k)$ by (7.15)
3. If $g(k) > h$, issue an alarm, provide an estimate of the change occurrence time $\hat{k}_0$ by (7.16) and reinitialise the decision function to 0.

Result: A sequence of alarm time instants k_a and estimated change occurrence times $\hat{k}_0$, for increasing time horizon k.

The reinitialisation after an alarm allows one to check whether the change in the mean persists as time elapses. More on this issue will be said when the algorithm will be used for fault detection applications.

Example 7.1 (**cont.**) **Change in the mean of a Gaussian sequence**
From the first 500 data samples plotted in Fig. 7.2, the following estimates were obtained

$$\hat{\mu}_0 = 0.0445 \qquad \hat{\sigma}^2 = 0.946.$$

Letting $\beta = 2$ yields $\hat{\mu}_1 = 2.0445$. Figure 7.6 gives the mean detection delay and the mean time between false alarms as a function of the threshold h, computed from (7.22) and (7.23) were the estimated values are substituted for μ_0, μ_1 and σ^2. The threshold $h = 10$ gives an estimated mean time between false alarms larger than 10^5, while assuring an estimated mean detection delay lower than 6 samples. The CUSUM algorithm (7.17) is applied to the data of Fig. 7.2 with the above parameter settings. Figure 7.7 gives a zoom of the decision function in the vicinity of the change time (namely time 500). The alarm will be issued at time 503 which corresponds to a detection delay of three samples (of the order of magnitude of the estimated one). □

Another option regarding the choice of θ_1 (the value of the parameter after change) consists in replacing it by the most likely value computed a posteriori from experimental data. This leads to the generalised likelihood ratio algorithm described in the next section.

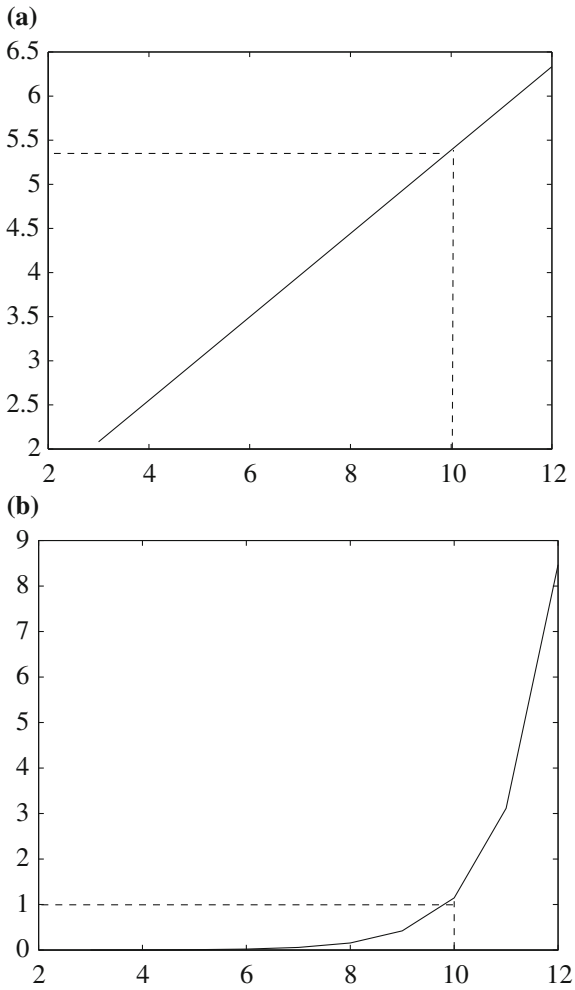


Fig. 7.6 Estimated mean detection delay in number of samples, as a function of h (**a**) and mean time between false alarms expressed in multiples of 10^5 samples as a function of h (**b**)

7.2.4 *Detection of an* Unknown *Change - The Generalised Likelihood Ratio Algorithm*

The problem can be stated in a similar way as for the offline derivation of the CUSUM algorithm (cf. Problem 7.1), except that θ_1 is unknown. By the same reasoning as above, the log-likelihood between hypotheses $\mathcal{H}_0$ and $\mathcal{H}_1$, with an hypothetical change time j, can be computed as

$$S_j^k(\theta_1) = \sum_{i=j}^{k} \ln \frac{p_{\theta_1}(z(i))}{p_{\theta_0}(z(i))}. \tag{7.24}$$

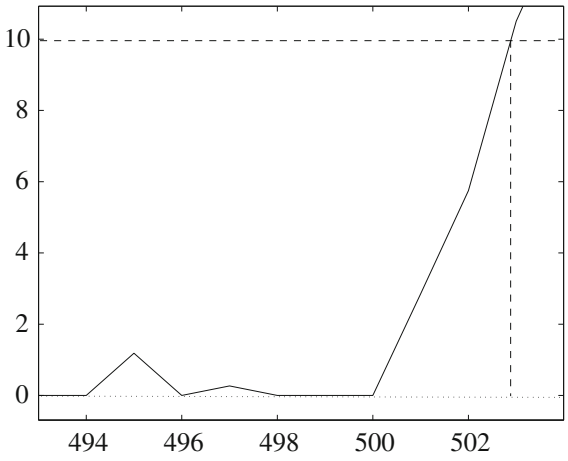


Fig. 7.7 Zoom on the decision function resulting from the recursive algorithm for the data of Fig. 7.2

For a given time instant k, it is a function of both j, the change time, and θ_1, the value of the parameter θ after the change. The standard statistical approach to estimate (7.24) is to replace j and θ_1 by their maximum likelihood estimates. The latter are obtained by solving the following double maximisation problem:

$$(\hat{k}_0, \hat{\theta}_1) = \arg \left\{ \max_{1 \le j \le k} \max_{\theta_1} S_j^k(\theta_1) \right\}, \tag{7.25}$$

where $\hat{k}_0$ denotes the estimate of the change time. The GLR decision function takes the form:

$$g(k) = \max_{1 \le j \le k} \max_{\theta_1} S_j^k(\theta_1). \tag{7.26}$$

The configuration and the implementation of the algorithm can be summarised as follows:

Algorithm 7.2 *GLR algorithm with scalar parameter*

Given:

1. A sequence of data $z(1), \ldots, z(k)$ with probability density function $p_\theta(z)$ depending on the scalar parameter θ.
2. A threshold h.

Compute: $g(k)$ using (7.24), (7.26).

Decide to

1. accept $\mathcal{H}_0$ if $g(k) \le h$.
2. accept $\mathcal{H}_1$ if $g(k) > h$.

The maximisation in (7.26) is performed over all possible past time instants. As time elapses, the considered time span increases which induces an increasing search duration for finding the optimum. To avoid that problem, the fault occurrence time is restricted to the last M time instants in practice. This amounts to assuming that the delay for detection is lower than M so that faults can be detected in M sampling periods at most. The actual decision function obtained from (7.26) is thus:

$$g(k) = \max_{k-M+1 \leq j \leq k} \max_{\theta_1} S_j^k(\theta_1). \tag{7.27}$$

Example 7.1 (cont.) Change in the mean of a Gaussian sequence
In the particular case of z being a Gaussian sequence with independent and identically distributed increments (IID), it is possible to find an explicit expression for $\hat{\mu}_1(k,\ j)$, the maximum likelihood estimate of μ_1 at the present time instant k, assuming that the fault occurred at time instant j. Indeed, from (7.7), $S_j^k(\mu_1)$ takes the following form:

$$S_j^k(\mu_1) = \frac{\mu_1 - \mu_0}{\sigma^2} \sum_{i=j}^{k} \left(z(i) - \frac{\mu_0 + \mu_1}{2} \right) \tag{7.28}$$

In order to maximise this expression with respect to μ_1, one has to take the derivative of $S_j^k(\mu_1)$ with respect to μ_1 and equate that expression to zero:

$$\frac{\partial S_j^k(\mu_1)}{\partial \mu_1} = \frac{1}{\sigma^2} \sum_{i=j}^{k} \left(z(i) - \frac{\mu_0 + \mu_1}{2} \right) - \frac{k-j+1}{2} \frac{(\mu_1 - \mu_0)}{\sigma^2} = 0. \tag{7.29}$$

Equation (7.29) yields the following estimate for μ_1:

$$\hat{\mu}_1(k,\ j) = \frac{1}{k-j+1} \sum_{i=j}^{k} z(i). \tag{7.30}$$

Substituting this expression for μ_1 in (7.28) results, after straightforward computations, in:

$$S_j^k(\hat{\mu}_1(k,\ j)) = \frac{1}{2\sigma^2} \frac{1}{k-j+1} \left[\sum_{i=j}^{k} (z(i) - \mu_0) \right]^2. \tag{7.31}$$

Hence the GLR decision function can be written:

$$g(k) = \frac{1}{2\sigma^2} \max_{k-M+1 \leq j \leq k} \frac{1}{k-j+1} \left[\sum_{i=j}^{k} (z(i) - \mu_0) \right]^2. \tag{7.32}$$

If $\mathcal{H}_1$ is accepted in the above GLR algorithm, at the alarm time k_a, the estimated change occurrence time is given as:

$$\hat{k}_0 = \arg \left\{ \frac{1}{2\sigma^2} \max_{k_a - M + 1 \leq j \leq k_a} \frac{1}{k_a - j + 1} \left[\sum_{i=j}^{k_a} (z(i) - \mu_0) \right]^2 \right\}. \tag{7.33}$$

□

Computing the most likely fault occurrence time might not be necessary. In this case, a cheap way to proceed, from a computational point of view consists of taking samples of fixed size, in a moving window say of length M, and in deciding between the following two hypotheses at the current time instant k (larger than $M - 1$):

$\mathcal{H}_0$: $\theta = \theta_0$ for $k - M + 1 \leq i \leq k$,
$\mathcal{H}_1$: $\theta = \theta_1$ for $k - M + 1 \leq i \leq k$.

The decision function becomes:

$$g_M(k) = \max_{\theta_1} S_{k-M+1}^k(\theta_1) \tag{7.34}$$

In the particular case of a change in the mean of a Gaussian sequence, it is easy to check that $g_M(k)$ takes the following form:

$$g_M(k) = \frac{1}{2\sigma^2 M} \left[\sum_{i=k-M+1}^{k} (z(i) - \mu_0) \right]^2. \tag{7.35}$$

Parameter tuning for the generalised likelihood ratio algorithm. Two approaches will be considered, depending on the specifications and the need to estimate the most likely fault occurrence time or not. In all cases, a change in the mean of a Gaussian sequence is considered.

No estimation of the fault occurrence time - Tuning from the statistical distribution of the test statistics. In this case, the probability law of the decision function $g_M(k)$ can be determined. This allows specifying the threshold h and the horizon M of the GLR test function (7.35) in order to meet a given probability of false alarm, α, and a given probability of correct detection, β.

To follow this approach, notice that

$$S_M(k) = \frac{1}{\sqrt{M}\sigma} \left[\sum_{i=k-M+1}^{k} (z(i) - \mu_0) \right]$$

has the following probability law:

$$\mathcal{L}(S_M(k)) = \mathcal{N}(0, 1) \text{ under } \mathcal{H}_0 \tag{7.36}$$

$$\mathcal{L}(S_M(k)) = \mathcal{N}\left(\frac{\sqrt{M}(\mu_1 - \mu_0)}{\sigma}, 1 \right) \text{ under } \mathcal{H}_1 \tag{7.37}$$

From the definition of the χ^2 distribution recalled below, the probability law for $g_M(k)$ is deduced:

$$\mathcal{L}(2g_M(k)) = \chi^2(1) \text{ under } \mathcal{H}_0 \tag{7.38}$$

$$\mathcal{L}(2g_M(k)) = \chi^2(1, \frac{M(\mu_1 - \mu_0)^2}{\sigma^2}) \text{ under } \mathcal{H}_1 \tag{7.39}$$

where $\chi^2(1)$ denotes the chi-square distribution with one degree of freedom, and $\chi^2(1, \lambda)$ denotes the chi-square distribution with one degree of freedom and non-centrality parameter λ.

Remark 7.4 (*Chi-square distribution*) If $\xi(i)$, $i = 1, \ldots, N$ are independent normally distributed random variables with zero mean and unit variance, then

$$X = \sum_{i=1}^{N} \xi(i)^2$$

has a chi-square distribution with N degrees of freedom. Its probability density function is

$$p_{\chi^2}(X; N) = \begin{cases} \frac{1}{2^{\frac{N}{2}}\Gamma(\frac{N}{2})} X^{\frac{N}{2}-1} e^{-\frac{X}{2}} & X \geq 0 \\ 0 & X < 0 \end{cases}. \tag{7.40}$$

where $\Gamma(u)$ denotes the gamma function. The mean of X is equal to N, and its variance is $2N$.

If the mean of $\xi(i)$ changes to μ_i, X has a chi-square distribution with N degrees of freedom and with non-centrality parameter $\lambda = \sum_{i=1}^{N} \mu_i^2$. Its mean is equal to $N + \lambda$, and its variance is $2N + 4\lambda$. The probability density function of the non-central chi-square distribution is

$$p_{\chi^2}(X; N, \lambda) = \frac{1}{2}\left(\frac{X}{\lambda}\right)^{\frac{N-2}{4}} \exp\left(-\frac{X+\lambda}{2}\right) \mathcal{I}_{\frac{N}{2}-1}(\sqrt{\lambda X}), \tag{7.41}$$

where $\mathcal{I}_r(u)$ is the modified Bessel function of first kind and order r. This probability density function is available in numerical form in software for statistical calculations. □

In order to enforce the given probabilities of false and correct detection, notice that the first is given as

$$P_F = P(g > h|\mathcal{H}_0) = \int_h^\infty p(g|\mathcal{H}_0)\,\mathrm{d}g \tag{7.42}$$

and the second as

$$P_D = P(g \geq h|\mathcal{H}_1) = \int_h^\infty p(g|\mathcal{H}_1)\,\mathrm{d}g \tag{7.43}$$

where $p(g|\mathcal{H}_0)$ ($p(g|\mathcal{H}_1)$) denotes de probability density function of the test function, g, conditioned on $\mathcal{H}_0$ ($\mathcal{H}_1$). Accounting to the previously determined probability density functions, Eqs. (7.42) and (7.43) yield

$$\int_{2h}^{\infty} p_{\chi^2}(X;1)dX = \alpha \tag{7.44}$$

and

$$\int_{2h}^{\infty} p_{\chi^2}\left(X;1,\frac{M(\mu_1-\mu_0)^2}{\sigma^2}\right)dX = \beta \tag{7.45}$$

Equations (7.44), (7.45) is a system of two nonlinear equations for the two unknowns h and M.

In industrial applications, data often violate the theoretical assumptions above. They are not IID and they are not Gaussian distributed. The actual distribution of the test statistic can be very different from the χ^2 statistics obtained from the theory above. One approach to address this issue is to identify an appropriate distribution for the test statistics from experimental data. This approach is explained and illustrated by an example below.

If data are available for both $\mathcal{H}_0$ and $\mathcal{H}_1$ cases, the cumulative density functions (CDF) $F(g_M|\mathcal{H}_0;M)$ and $F(g_M|\mathcal{H}_1;\mu_1,M)$ can be estimated from these data for different values of the horizon M. From these CDFs, the test threshold h and horizon M can be determined to achieve a required probability of false alarm α and a required probability of correct detection β. This can be expressed formally as:

$$\alpha = 1 - F(h|\mathcal{H}_0;M) \Rightarrow h = F^{-1}(1-\alpha;M) \tag{7.46}$$

and

$$\beta = 1 - F(h|\mathcal{H}_1;\mu_1,M) \Rightarrow M = F^{-1}(1-\beta;\mu_1,h) \tag{7.47}$$

Hence it is possible to determine a window size that provides a desired probability of detection.

Enforcing a probability of false alarm and correct detection is in the spirit of the design of non-sequential tests. When specifications are in terms of mean detection delay and/or mean time between false alarms, and the estimate of the fault occurrence time matters, we propose to resort to an experimental approach to adjust the parameters h and M.

Estimation of the fault occurrence time - Tuning from experimental data. Although the method is described with reference to Example 7.1, it can be generalised easily to other types of changes than jumps in the mean of a Gaussian sequence. The threshold should be determined on the basis of healthy and faulty process data. A computation of the decision function based on a set of healthy data, $\mathcal{Z}_0 = \{z_0(1),\ldots,z_0(N_0)\}$, allows one to determine the typical range of values of this function in the absence of fault, and to set the threshold in such a way that the time between false alarms is very high. This choice can then be validated by

processing data obtained in faulty working mode, $\mathcal{Z}_1 = \{z_1(1), \ldots, z_1(N_1)\}$, and checking that detection is achieved with an acceptable delay. Should experimental data corresponding to a faulty behaviour not be available, a simulator could possibly be used to obtain data that could be used as a substitute. An adjustment of the horizon M and the threshold h may be needed to obtain the right compromise between false alarm rate and detection delay. Indeed, the lower M, the lower h has to be chosen in order to achieve detection in the window $[k - M + 1,\ k]$. Decreasing the detection delay thus increases the false alarm rate. For the evaluation of the GLR decision function (7.32) required in the above procedure, empirical estimates $\hat{\mu}_0$ and $\hat{\sigma}^2$ should be substituted for μ_0 and σ^2.

The initialisation procedure thus takes the form:

Algorithm 7.3 *Initialisation procedure*

Given: Data sets $\mathcal{Z}_0$ and $\mathcal{Z}_1$, and an acceptable maximum detection delay τ_{max}.

Initialisation:

1. Choose M larger than or equal to τ_{max}.
2. Determine $\hat{\mu}_0$ and $\hat{\sigma}^2$ from $\mathcal{Z}_0$.
3. Compute the decision function $g(i), i = M + 1, \ldots, N_0$ for the data set $\mathcal{Z}_0$ by using (7.32) and choose the threshold h so that $g(i) < h,\ i = M + 1, \ldots, N_0$.
4. Compute the decision function $g(i),\ i = M + 1, \ldots, N_1$ for the data set $\mathcal{Z}_1$ by (7.32) and the estimated change magnitude by (7.30). Check that the fault is detected and that the delay for detection is acceptable.
5. Possibly iterate on the choice of M and h.
6. Acquire $M - 1$ data samples.

Reinitialisation. Again the particular case of a change in the mean of a Gaussian sequence is considered here. The reinitialisation allows one to detect a new change in the mean. The mean value of the data after change is thus considered as the new value of μ_0. This reinitialisation could use $\hat{\mu}_1(k_a, \hat{k}_0)$ as an estimate of the mean after the change occurred. However, if the delay for detection is short (one or a few samples), very few data are used to compute $\hat{\mu}_1(k_a, \hat{k}_0)$, and this estimate of the mean might be poor when the noise on the data is significant. It is the reason why the reinitialisation is based on a data set of fixed length obtained by collecting additional data. Here the length of the data set is chosen equal to M, but an additional parameter different from M might be introduced. It should be determined in such a way that a reliable estimate of the mean after change is obtained. More on this can be found in the appendix on random variables and stochastic processes, where the statistics of the empirical mean is studied.

With this in mind, the global GLR algorithm to detect a change in the mean of a Gaussian sequence can be summarised as follows:

Algorithm 7.4 *GLR algorithm to detect and estimate changes in the mean of a Gaussian sequence*

Initialisation

1. Run the above described initialisation procedure (Algorithm 7.3) from available fault free and faulty data sets.

At each sampling time:

R1. Acquire the new data $z(k)$.

R2. Compute $g(k)$ from (7.32).

R3. If $g(k) > h$, generate an alarm, provide the alarm time instant $k_a = k$, the estimate of the change occurrence time $\hat{k}_0$ by (7.33), and compute $\hat{\mu}_1(k_a, \hat{k}_0)$ by (7.30).

Reinitialisation:

1. Collect a set of M data from time $\hat{k}_0$ to $\hat{k}_0 + M - 1$.

2. Compute the new value of $\hat{\mu}_0$ from these data.

3. Restart the online algorithm from $k = \hat{k}_0 + M$ onwards (step R1).

Result: A sequence of alarm time instants k_a, estimated change occurrence times $\hat{k}_0$ and mean signal values $\hat{\mu}_1(k_a, \hat{k}_0)$, for increasing time horizon k.

Example 7.1 (cont.) Change in the mean of a Gaussian sequence
Consider again the data of Fig. 7.2. Let the set $\mathcal{Z}_0$ be made of the first 500 samples, while $\mathcal{Z}_1$ consists of samples 400–1000. One gets, as before, $\hat{\mu}_0 = 0.0445$, $\hat{\sigma}^2 = 0.9455$. Figure 7.8a gives the value of the GLR decision function obtained by processing the sequence $z_0(1), \ldots, z_0(N_0)$ with a window M of length 10 samples. A threshold above 15 appears to be suitable in this case. Hence, h is set to 20. Running the algorithm on the set $z_1(1), \ldots, z_1(N_1)$, one observes that an alarm is generated at time 105 (Fig. 7.8b). The estimate of the change magnitude is 2.69, and the estimate of the change occurrence time is 103, while the actual change occurred at the 101st sample in the set. Due to the noise on the signal, the estimate of the change magnitude is in error by 30 %. This could be partly alleviated by increasing the threshold, so that more data are used to estimate the fault magnitude; this would increase the detection delay, however. □

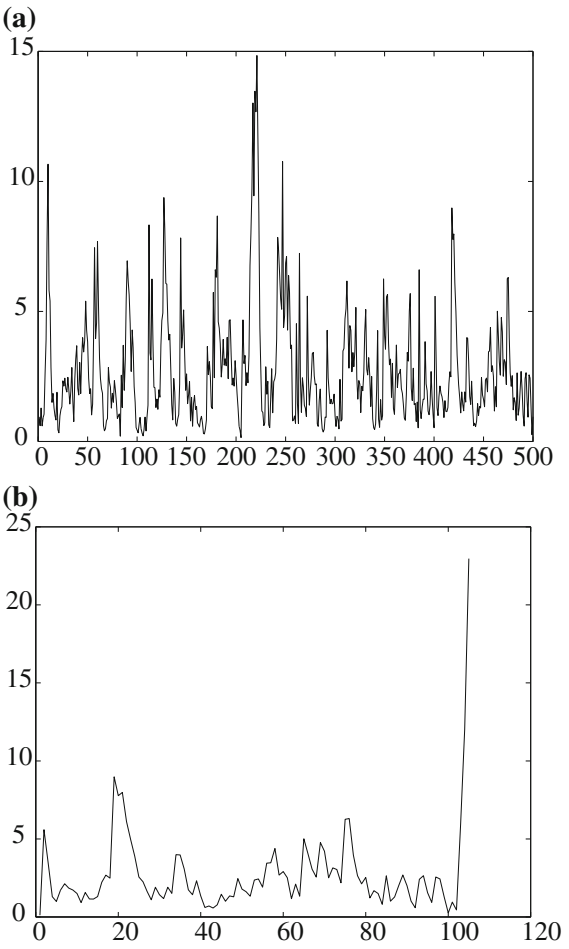


Fig. 7.8 Two GLR decision functions

7.2.5 Sequential Change Detection: The Vector Case

Problem statement. The previous discussion dealt with detection of changes in a scalar signal. In the fault detection applications, the signal to be processed is issued by a residual generator, and it is generally a vector signal. The combined information comprised in this vector should be considered in our algorithm. Since fault detection will be reduced to the detection of changes in the mean of a Gaussian vector sequence, the solution to the following problem will be needed.

Problem 7.2 (*Detection of a change in the mean of a Gaussian vector sequence*) Consider a sequence of n_z-dimensional random vectors $z(1), \ldots, z(k)$ that are independent and distributed as $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{Q})$, where $\boldsymbol{Q}$ is known, as well as the nominal value for $\boldsymbol{\mu}$, $\boldsymbol{\mu}_0$. Choose between the following two hypotheses:

- $\mathcal{H}_0 : \mathcal{L}(z(i)) = \mathcal{N}(\boldsymbol{\mu}_0,\ \boldsymbol{Q})$, $(i = 1, \ldots, k)$
- $\mathcal{H}_1$: From time instant 1 up to an unknown time instant k_0, $z(i)$, $i = 1, \ldots, k_0 - 1$ is distributed as

$$\mathcal{L}(z(i)) = \mathcal{N}(\boldsymbol{\mu}_0,\ \boldsymbol{Q}) \tag{7.48}$$

while for time instants $i \geq k_0$

$$\mathcal{L}(z(i)) = \mathcal{N}(\boldsymbol{\mu}_1,\ \boldsymbol{Q}) \tag{7.49}$$

with $\boldsymbol{\mu}_1 \neq \boldsymbol{\mu}_0$.

Besides detecting the possible change in the mean, one should also estimate its time of occurrence, and possibly its magnitude.

When $\boldsymbol{\mu}_1$ is known, the detection algorithm is a direct generalisation of the CUSUM algorithm. In the second situation which is considered in this paragraph, the change in $\boldsymbol{\mu}$ has known direction but unknown magnitude. This yields a GLR algorithm. Finally, the situation, where $\boldsymbol{\mu}_1$ is replaced by a dynamical profile of change will be considered, as this is a result needed at a later stage.

$\boldsymbol{\mu}_1$ known - CUSUM algorithm. By using the expression of the probability density function of a n-dimensional Gaussian vector z with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{Q}$

$$p_{\boldsymbol{\mu}}(z) = \frac{1}{\sqrt{(2\pi)^n det\, \boldsymbol{Q}}} \exp\left(-\frac{1}{2}\,(z-\boldsymbol{\mu})^{\mathrm{T}}\ \boldsymbol{Q}^{-1}(z-\boldsymbol{\mu})\right), \tag{7.50}$$

the following expression is obtained for the log-likelihood ratio associated to the above problem:

$$\begin{aligned} s(z(k)) &= \ln \frac{p_{\boldsymbol{\mu}_1}(z(k))}{p_{\boldsymbol{\mu}_0}(z(k))} \\ &= -\frac{1}{2}\left(z(k)-\boldsymbol{\mu}_1\right)^{\mathrm{T}}\ \boldsymbol{Q}^{-1}\left(z(k)-\boldsymbol{\mu}_1\right) + \frac{1}{2}\left(z(k)-\boldsymbol{\mu}_0\right)^{\mathrm{T}}\ \boldsymbol{Q}^{-1}\left(z(k)-\boldsymbol{\mu}_0\right) \\ &= \left(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0\right)^{\mathrm{T}}\ \boldsymbol{Q}^{-1}\left(z(k)-\frac{1}{2}(\boldsymbol{\mu}_0+\boldsymbol{\mu}_1)\right) \end{aligned} \tag{7.51}$$

This log-likelihood ratio is scalar, so the recursive computation of the CUSUM decision function can be performed in a similar way as for the scalar case (cf. Eq. (7.14))

$$g(k) = \max(0,\ g(k-1) + s(z(k))). \tag{7.52}$$

The alarm or stopping time, k_a, is the smallest time instant at which $g(k)$ crosses a given threshold.

Known direction of change - GLR algorithm. Let $\boldsymbol{\mu}_1$ be of the form

$$\boldsymbol{\mu}_1 = \boldsymbol{\mu}_0 + \boldsymbol{\Gamma}\nu,$$

where $\boldsymbol{\Gamma}$ is a known vector, and ν is an unknown scalar change magnitude. Substituting this expression for $\boldsymbol{\mu}_1$ in (7.51) allows one to deduce the following expression of the cumulative sum $S_j^k(\nu)$, where j denotes an hypothetical value of the change time k_0,

$$\begin{aligned} S_j^k(\nu) &= \sum_{i=j}^{k} \ln \frac{p_{\boldsymbol{\mu}_0+\boldsymbol{\Gamma}\nu}(\boldsymbol{z}(i))}{p_{\boldsymbol{\mu}_0}(\boldsymbol{z}(i))} \\ &= \sum_{i=j}^{k} \left(\nu \boldsymbol{\Gamma}^{\mathrm{T}} \boldsymbol{Q}^{-1} \left(\boldsymbol{z}(i) - \boldsymbol{\mu}_0 \right) - \frac{1}{2} \nu^2 \boldsymbol{\Gamma}^{\mathrm{T}} \boldsymbol{Q}^{-1} \boldsymbol{\Gamma} \right). \end{aligned} \tag{7.53}$$

Equating $\frac{\partial S_j^k(\nu)}{\partial \nu}$ to zero yields

$$\begin{aligned} \frac{\partial S_j^k(\nu)}{\partial \nu} &= \sum_{i=j}^{k} \boldsymbol{\Gamma}^{\mathrm{T}} \boldsymbol{Q}^{-1} \left(\boldsymbol{z}(i) - \boldsymbol{\mu}_0 \right) - (k-j+1)\, \boldsymbol{\Gamma}^{\mathrm{T}} \boldsymbol{Q}^{-1} \boldsymbol{\Gamma} \nu \\ &= (k-j+1)\, \boldsymbol{\Gamma}^{\mathrm{T}} \boldsymbol{Q}^{-1} \left(\bar{\boldsymbol{Z}}_j^k - \boldsymbol{\mu}_0 \right) - (k-j+1)\, \boldsymbol{\Gamma}^{\mathrm{T}} \boldsymbol{Q}^{-1} \boldsymbol{\Gamma} \nu \\ &= 0 \end{aligned} \tag{7.54}$$

with $\bar{\boldsymbol{Z}}_j^k = \frac{1}{k-j+1} \sum_{i=j}^{k} \boldsymbol{z}(i)$.

Hence, the maximum likelihood estimate of ν at time k, assuming the fault occurred at time j is obtained from (7.54) as

$$\hat{\nu}(k,\ j) = \frac{\boldsymbol{\Gamma}^{\mathrm{T}} \boldsymbol{Q}^{-1} (\bar{\boldsymbol{Z}}_j^k - \boldsymbol{\mu}_0)}{\boldsymbol{\Gamma}^{\mathrm{T}} \boldsymbol{Q}^{-1} \boldsymbol{\Gamma}}. \tag{7.55}$$

Substituting (7.55) for ν into (7.53) finally yields the GLR decision function in a similar way as (7.32) was deduced from (7.30) and (7.31)

$$\begin{aligned} g(k) = \max_{k-M+1 \le j \le k} S_j^k(\hat{\nu}(k,\ j)) = \max_{k-M+1 \le j \le k} (k-j+1) \cdot \\ \cdot \left(\hat{\nu}(k,\ j)\, \boldsymbol{\Gamma}^{\mathrm{T}} \boldsymbol{Q}^{-1} \left(\bar{\boldsymbol{Z}}_j^k - \boldsymbol{\mu}_0 \right) - \frac{1}{2} \hat{\nu}(k,\ j)^2\, \boldsymbol{\Gamma}^{\mathrm{T}} \boldsymbol{Q}^{-1} \boldsymbol{\Gamma} \right). \end{aligned}$$

The estimated fault occurrence time upon acceptance of hypothesis $\mathcal{H}_1$ at time instant k_{a} is given as

$$\begin{aligned} \hat{k}_0 = \arg \Bigg\{ \max_{k_{\mathrm{a}}-M+1 \le j \le k_{\mathrm{a}}} (k_{\mathrm{a}} - j + 1) \cdot \\ \cdot \left(\hat{\nu}(k_{\mathrm{a}},\ j) \boldsymbol{\Gamma}^{\mathrm{T}} \boldsymbol{Q}^{-1} \left(\bar{\boldsymbol{Z}}_j^{k_{\mathrm{a}}} - \boldsymbol{\mu}_0 \right) - \frac{1}{2} \hat{\nu}(k_{\mathrm{a}},\ j)^2\, \boldsymbol{\Gamma}^{\mathrm{T}} \boldsymbol{Q}^{-1} \boldsymbol{\Gamma} \right) \Bigg\}. \end{aligned}$$

Known dynamical profile of change - CUSUM algorithm. There is a need to generalise the previous result by replacing $\boldsymbol{\Gamma}\nu$ by a time-varying change direction, as this is precisely the situation which is encountered when detecting additive faults in linear systems. This leads to the following modified version of Problem 7.2.

Problem 7.3 (*Change detection, known dynamical profile of change*) Consider a sequence of n_z-dimensional random vectors $z(1), \ldots, z(k)$ that are independent and distributed as $\mathcal{N}(\boldsymbol{\mu},\ \boldsymbol{Q})$, where $\boldsymbol{Q}$ is known, as well as the nominal value for $\boldsymbol{\mu}$, $\boldsymbol{\mu}_0$. Choose between the following two hypotheses:

- $\mathcal{H}_0 : \mathcal{L}(z(i)) = \mathcal{N}(\boldsymbol{\mu}_0,\ \boldsymbol{Q}),\ (i = 1, \ldots, k)$
- $\mathcal{H}_1$: From time instant 1 up to an unknown time instant k_0, $z(i)$, $i = 1, \ldots, k_0$–1 is distributed as

$$\mathcal{L}(z(i)) = \mathcal{N}(\boldsymbol{\mu}_0,\ \boldsymbol{Q}) \tag{7.56}$$

while for time instants $i \geq k_0$:

$$\mathcal{L}(z(i)) = \mathcal{N}(\boldsymbol{\mu}_0 + \boldsymbol{\rho}(i - k_0),\ \boldsymbol{Q}), \tag{7.57}$$

where $\boldsymbol{\rho}(i - k_0)$ is a known vector profile which is non-zero only for $i \geq k_0$.

Besides detecting the possible change in the mean, one should also estimate its time of occurrence.

The cumulative sum for this problem setting, with j as an hypothetical value for k_0, is given as

$$\begin{aligned} S_j^k &= \sum_{i=j}^{k} \ln \frac{p_{\boldsymbol{\mu}_0 + \boldsymbol{\rho}(i-j)}(z(i))}{p_{\boldsymbol{\mu}_0}(z(i))} \\ &= \sum_{i=j}^{k} \left(-\frac{1}{2}(z(i) - \boldsymbol{\mu}_0 - \boldsymbol{\rho}(i-j))^{\mathrm{T}}\, \boldsymbol{Q}^{-1}(z(i) - \boldsymbol{\mu}_0 - \boldsymbol{\rho}(i-j)) \right) + \\ &\quad + \frac{1}{2}\sum_{i=j}^{k} (z(i) - \boldsymbol{\mu}_0)^{\mathrm{T}}\, \boldsymbol{Q}^{-1}\left(z(i) - \boldsymbol{\mu}_0\right) \\ &= \sum_{i=j}^{k} \boldsymbol{\rho}(i-j)^{\mathrm{T}}\, \boldsymbol{Q}^{-1}\left(z(i) - \boldsymbol{\mu}_0\right) - \frac{1}{2}\sum_{i=j}^{k} \boldsymbol{\rho}(i-j)^{\mathrm{T}}\, \boldsymbol{Q}^{-1}\boldsymbol{\rho}(i-j) \end{aligned} \tag{7.58}$$

and the decision function, obtained in a similar way as for the scalar case, is given as (cf. Eq. (7.12))

$$g(k) = \max_{1 \leq j \leq k} S_j^k$$

with S_j^k as in (7.58).

This algorithm can be written in a recursive form ([11], pp. 283–284):

$$g(k) = \max(0,\ S(k)) \tag{7.59}$$

$$N(k) = N(k-1)\, 1_{\{g(k-1)>0\}} + 1 \tag{7.60}$$

$$\begin{aligned} S(k) = {} & S(k-1)\, 1_{\{g(k-1)>0\}} + \boldsymbol{\rho}(N(k)-1)^{\mathrm{T}}\, \boldsymbol{Q}^{-1}(\boldsymbol{z}(k) - \boldsymbol{\mu}_0) - \\ & -0.5\boldsymbol{\rho}(N(k)-1)^{\mathrm{T}}\, \boldsymbol{Q}^{-1}\boldsymbol{\rho}\,(N(k)-1). \end{aligned} \tag{7.61}$$

$N(k)$ is thus the number of observations after the last time instant for which the decision function g was null. Note that this algorithm requires the hypothesis $\boldsymbol{\rho}(0) \neq 0$, which is included in the problem statement, otherwise the decision function would always remain equal to zero.

Example 7.3 Data exhibiting a dynamical profile of change
The aim of this example is to illustrate the type of vector signal on which the above algorithm can be applied. Consider a vector signal made of two components, z_1 and z_2. Suppose that the dynamical profile of the change in z_1 (z_2) can be modelled as the step response to a first-order system with transfer function $\frac{0.5}{z-0.5}$ ($\frac{1.4}{z-0.3}$). In other words, the sequence $z_j(i), i = 1, 2, \ldots, j = 1, 2$, takes the form

$$z_j(i) = z_j^0(i) + \rho_{s,j}(i - k_0)\, 1_{\{i \geq k_0\}}, \tag{7.62}$$

where

$$\mathcal{L}(z_1^0(i)) = \mathcal{L}(z_2^0(i)) = \mathcal{N}(0, 0.025),$$

hold and $\rho_{s,j}(\ell)$, which are tabulated below for $\ell = 0, \ldots, 9$, $j = 1, 2$, are the step responses (hence the index s) associated to $\frac{0.5}{z-0.5}$ and $\frac{1.4}{z-0.3}$. Figure 7.9 gives a realisation of the sequence (7.62) for $k_0 = 20$.

The superimposition of the deterministic step response and the stochastic sequence is clearly visible. To apply algorithm (7.59)–(7.61) to detect the change in the sequence, one should take $\rho_j(\ell) = \rho_{s,j}(\ell + 1)$, $(\ell = 0, 1, 2 \ldots, j = 1, 2)$ in order to assure that $\boldsymbol{\rho}(0)$ be non-zero. □

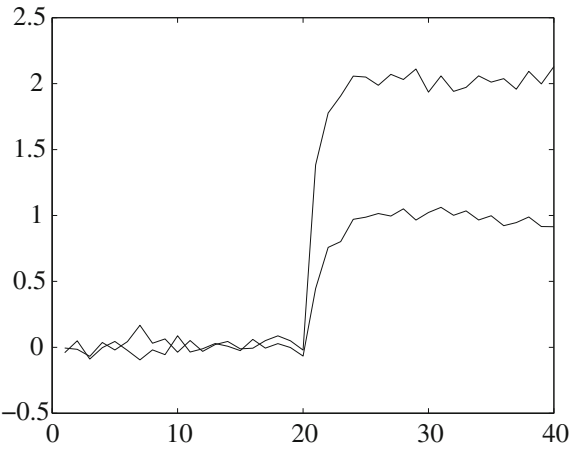


Fig. 7.9 Realisation of the vector sequence (7.62)

Table 7.1 First 10 values of the dynamical profile of the change

ℓ	$\rho_{s,1}$	$\rho_{s,2}$
0	0	0
1	0.5000	1.4000
2	0.7500	1.8200
3	0.8750	1.9460
4	0.9375	1.9838
5	0.9688	1.9951
6	0.9844	1.9985
7	0.9922	1.9996
8	0.9961	1.9999
9	0.9980	2.0000

Known dynamical profile of change up to an unknown constant - GLR algorithm. Yet a more general situation occurs when the form of the dynamical profile of change is known (Table 7.1), but its magnitude is not known.

Problem 7.4 (*Change detection, known dynamical profile of change up to an unknown constant*) Consider a sequence of n_z-dimensional random vectors $\boldsymbol{z}(1), \ldots, \boldsymbol{z}(k)$ that are independent and distributed as $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{Q})$, where $\boldsymbol{Q}$ is known, as well as the nominal value for $\boldsymbol{\mu}$, $\boldsymbol{\mu}_0$. Choose between the following two hypotheses:

- $\mathcal{H}_0 : \mathcal{L}(\boldsymbol{z}(i)) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{Q}), (i = 1, \ldots, k)$
- $\mathcal{H}_1$: From time instant 1 up to an unknown time instant k_0, $\boldsymbol{z}(i), (i = 1, \ldots, k_0 - 1)$ is distributed as

$$\mathcal{L}(\boldsymbol{z}(i)) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{Q}) \tag{7.63}$$

while for time instants $i \geq k_0$

$$\mathcal{L}(\boldsymbol{z}(i)) = \mathcal{N}(\boldsymbol{\mu}_0 + \tilde{\boldsymbol{\rho}}\,(i - k_0)\nu, \boldsymbol{Q}), \tag{7.64}$$

where $\tilde{\boldsymbol{\rho}}\,(i - k_0)$ is a known vector profile which is non-zero only for $i \geq k_0$, k_0 is an unknown time instant, and ν is an unknown scalar.

Besides detecting the possible change in the mean, one should also estimate its time of occurrence and its magnitude ν.

The cumulative sum for this problem setting, with j as an hypothetical value for k_0, is given as

$$S_j^k(\nu) = \sum_{i=j}^{k} \ln \frac{p_{\boldsymbol{\mu}_0 + \tilde{\boldsymbol{\rho}}\,(i-j)\nu}(\boldsymbol{z}(i))}{p_{\boldsymbol{\mu}_0}(\boldsymbol{z}(i))}$$
$$= \nu \sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)^{\mathrm{T}}\, \boldsymbol{Q}^{-1} \left(\boldsymbol{z}(i) - \boldsymbol{\mu}_0\right) - \frac{\nu^2}{2} \sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)^{\mathrm{T}}\, \boldsymbol{Q}^{-1} \tilde{\boldsymbol{\rho}}\,(i-j). \tag{7.65}$$

Similar computations as for the case of a constant direction of the parameter change yield the following maximum likelihood estimate of ν at time k, assuming the fault occurred at time j,

$$\hat{\nu}(k,\ j) = \frac{\displaystyle\sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)^{\mathrm{T}}\, \boldsymbol{Q}^{-1} \left(\boldsymbol{z}(i) - \boldsymbol{\mu}_0\right)}{\displaystyle\sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)^{\mathrm{T}}\, \boldsymbol{Q}^{-1} \tilde{\boldsymbol{\rho}}\,(i-j)} \tag{7.66}$$

The GLR decision function is directly deduced from (7.65) and (7.66) as

$$g(k) = \max_{k-M+1 \le j \le k} \max_{\nu}\ S_j^k(\nu)$$
$$= \max_{k-M+1 \le j \le k} \{\hat{\nu}(k,\ j) \sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)^{\mathrm{T}}\, \boldsymbol{Q}^{-1} \left(\boldsymbol{z}(i) - \boldsymbol{\mu}_0\right)$$
$$- \frac{\hat{\nu}(k,\ j)^2}{2} \sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)^{\mathrm{T}}\, \boldsymbol{Q}^{-1} \tilde{\boldsymbol{\rho}}\,(i-j)\}. \tag{7.67}$$

The estimated fault occurrence time upon acceptance of hypothesis $\mathcal{H}_1$ at time instant k_a is given as

$$\hat{k}_0 = \arg \left\{ \max_{k_\mathrm{a}-M+1 \le j \le k_\mathrm{a}} \hat{\nu}(k_\mathrm{a},\ j) \sum_{i=j}^{k_\mathrm{a}} \tilde{\boldsymbol{\rho}}\,(i-j)^{\mathrm{T}}\, \boldsymbol{Q}^{-1} \left(\boldsymbol{z}(i) - \boldsymbol{\mu}_0\right) - \right.$$
$$\left. - \frac{\hat{\nu}(k_\mathrm{a},\ j)^2}{2} \sum_{i=j}^{k_\mathrm{a}} \tilde{\boldsymbol{\rho}}\,(i-j)^{\mathrm{T}}\, \boldsymbol{Q}^{-1} \tilde{\boldsymbol{\rho}}\,(i-j) \right\}. \tag{7.68}$$

Parameter setting for the CUSUM algorithm. For the CUSUM algorithm associated to a known vector $\boldsymbol{\mu}_1$, the expressions of the ARL function (7.20) remains valid in the vector case. It suffices to replace μ_s and σ_s by:

$$\mu_s = \pm \frac{(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^{\mathrm{T}}\, \boldsymbol{Q}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)}{2}$$
$$\sigma_s^2 = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^{\mathrm{T}}\, \boldsymbol{Q}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0),$$

where the plus or the minus sign are chosen according to the value of $\boldsymbol{\mu}$, the expected value of $\boldsymbol{z}(i)$ (cf. Remark 7.1). The mean time for detection and the mean time between false alarms can, respectively, be estimated as

$$\hat{\bar{\tau}} = \hat{L} \left(\frac{(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^{\mathrm{T}}\, \boldsymbol{Q}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)}{2} \right) = \hat{L} \left(\frac{1}{2} \boldsymbol{\beta}^{\mathrm{T}}\, \boldsymbol{Q}^{-1} \boldsymbol{\beta} \right), \tag{7.69}$$

where $\boldsymbol{\beta} = \boldsymbol{\mu}_1 - \boldsymbol{\mu}_0$

$$\hat{\bar{T}} = \hat{L} \left(-\frac{(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^{\mathrm{T}}\, \boldsymbol{Q}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)}{2} \right). \tag{7.70}$$

In order to evaluate the above expressions, given a set of experimental data recorded in fault free condition, $\mathcal{Z}_0 = \{z_0(1), \ldots, z_0(N_0)\}$, and a change magnitude $\boldsymbol{\beta}$, the empirical mean and variances $\hat{\boldsymbol{\mu}}_0$ and $\hat{\boldsymbol{Q}}$ should be substituted for $\boldsymbol{\mu}_0$ and $\boldsymbol{Q}$. These empirical estimates should also be used for the implementation of (7.51)–(7.52).

The algorithm for the vector parameter case is thus similar to the scalar case, namely,

Algorithm 7.5 *CUSUM algorithm for Gaussian vector sequence with step-like change ($\boldsymbol{\mu}_1$ known)*

Given: A set of experimental data $\mathcal{Z}_0$, a change magnitude β, and a specified mean time for detection $\bar{\tau}_{acc}$ or a specified mean time between false alarms $\bar{T}_{acc}$.

Initialisation:

1. Determine $\hat{\boldsymbol{\mu}}_0$ and $\hat{\boldsymbol{Q}}$ from $\mathcal{Z}_0$
2. Choose h to meet either $\bar{\tau}_{acc}$ or $\bar{T}_{acc}$ using (7.69) or (7.70).

At each sample time:

1. Acquire the new data vector $\boldsymbol{z}(k)$.
2. Compute $g(k)$ by (7.52).
3. If $g(k) > h$, issue an alarm and reinitialise the decision function to 0.

Result: A sequence of alarm time instants k_{a} and estimated change occurrence times $\hat{k}_0$, for increasing time horizon k.

When the dynamical profile of the change is accounted for (as in (7.57)), the study of the properties of the algorithm such as mean delay for detection and mean time between false alarms becomes much more difficult. The difficulty stems from the fact that the increments in the cumulative sum of log-likelihood ratios are not identically distributed. Therefore, an experimental approach for setting the design parameters is proposed in the algorithm below. Data sets $\mathcal{Z}_0 = \{z_0(1), \ldots, z_0(N_0)\}$ and $\mathcal{Z}_1 = \{z_1(1), \ldots, z_1(N_1)\}$, respectively, recorded under hypothesis $\mathcal{H}_0$ and $\mathcal{H}_1$ are considered.

Algorithm 7.6 *CUSUM algorithm for Gaussian vector sequence with known dynamical profile of change*

Given: Data sets $\mathcal{Z}_0$ and $\mathcal{Z}_1$ and a dynamical profile of change $\boldsymbol{\rho}(i) \neq 0,\ i \geq 0$.

Initialisation:

1. Determine $\hat{\boldsymbol{\mu}}_0$ and $\hat{\boldsymbol{Q}}$ from $\mathcal{Z}_0$.

2. Compute the decision function $g(i), i = 1, \ldots, N_0$ for the data set $\mathcal{Z}_0$ by (7.59)–(7.61) and choose the threshold h so that $g(i) < h, i = 1, \ldots, N_0$.

3. Compute the decision function $g(i), i = 1, \ldots, N_1$ for the data set $\mathcal{Z}_1$ by (7.59)–(7.61); check that the fault is detected and that the delay for detection is acceptable.

4. Possibly iterate on the choice of h.

At each sampling time:

R1. Acquire the new data $\boldsymbol{z}(k)$.

R2. Compute $g(k)$ from (7.59)–(7.61).

R3. If $g(k) > h$, generate an alarm by setting $k_\mathrm{a} = k$, and provide an estimate of the change occurrence time as $k_\mathrm{a} - N(k_\mathrm{a})$ by (7.60).

Reinitialisation:

1. Reset $g(k_\mathrm{a})$, $N(k_\mathrm{a})$, and $S(k_\mathrm{a})$ to zero in (7.59)–(7.61).

2. Restart the recursive algorithm with (step R1).

Result: A sequence of alarm time instants k_a and estimated fault occurrence times $\hat{k}_0$, for increasing time horizon k.

Remark 7.5 (*Reinitialisation procedure*) The reinitialisation may depend on what one wishes to detect. Here it is assumed that one wishes to check whether the observed change remains present. By reinitialising the algorithm as proposed, repeated alarms will occur as long as the change is present in the signal. Another option for reinitialisation is to change the sign of the log-likelihood ratio which amounts to changing the sign of the last two terms in (7.61) and to reset all variables to zero as indicated in step 1 of the reinitialisation. In this way a return to normal will generate an alarm.

The proposed reinitilisation policies require that the dynamical profile of change does not asymptotically vanish. Should this not hold, one should resort to a GLR algorithm as illustrated in the ship example in Sect. 7.3.4. □

An example of application of this algorithm in the framework of a fault detection system is given below in Sect. 7.3.2.

Parameter setting for the GLR algorithm. Here also an experimental approach is used to set the design parameters. The algorithm is only presented for a change characterised by a dynamical profile with unknown magnitude.

The initialisation algorithm is similar to Algorithm 7.3 except that, besides $\mathcal{Z}_0$ and $\mathcal{Z}_1$ and an acceptable detection delay, a dynamical profile of change $\tilde{\rho}(i)$, $i \geq 0$, is also given. Furthermore, all scalar values should be replaced by their vector or matrix counterpart. In particular, in steps 3 and 4, the decision function should be computed from (7.66), (7.67) instead of (7.30), (7.32).

Algorithm 7.7 *GLR algorithm with known dynamical profile but unknown change magnitude*

Initialisation:

1. Run the initialisation procedure (Algorithm 7.3) with the above mentioned changes.

At each sampling time:

R1. Acquire the new data $z(k)$.

R2. Compute $g(k)$ from (7.66), (7.67)

R3. If $g(k) > h$, generate an alarm, provide the estimate of the change occurrence time, $\hat{k}_0$, by (7.68), and the estimated change magnitude $\hat{\nu}(k, \hat{k}_0)$ computed by (7.66).

Reinitialisation:

1. Collect a set of M data from time $\hat{k}_0$ to $\hat{k}_0 + M - 1$.

2. Compute the estimated change magnitude $\hat{\boldsymbol{\nu}}(\hat{k}_0 + M - 1, \hat{k}_0)$ for these data by (7.66).

3. Restart the recursive algorithm from $k = \hat{k}_0 + M$ onwards with step R1 while accounting for the remark below.

Result: A sequence of alarm time instants k_a, estimated change occurrence times $\hat{k}_0$ and change magnitudes $\hat{\nu}(k, \hat{k}_0)$.

Remark 7.6 (*Reinitialisation for GLR algorithm*)

- The reason for collecting a set of M data to estimate $\hat{\nu}(\hat{k}_0 + M - 1,\ \hat{k}_0)$ is to assure a sufficient precision of the estimated change magnitude so that updating the mean of the signal is performed properly.
- After reinitialisation, (7.66) and (7.67) should be replaced in step R2 by the following expression which accounts for the estimated mean of the signal after change

$$\hat{\nu}(k,\ j) = \frac{\sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)^{\mathrm{T}}\ \boldsymbol{Q}^{-1}\left(\boldsymbol{z}(i) - \hat{\boldsymbol{\mu}}_0 - \hat{\nu}\tilde{\rho}(i-\hat{k}_0)\right)}{\sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)^{\mathrm{T}}\ \boldsymbol{Q}^{-1}\tilde{\boldsymbol{\rho}}\,(i-j)} \tag{7.71}$$

$$g(k) = \max_{k-M+1\le j\le k} \left\{ \hat{\nu}(k,\ j) \sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)^{\mathrm{T}}\ \boldsymbol{Q}^{-1}\left(\boldsymbol{z}(i) - \hat{\boldsymbol{\mu}}_0 - \hat{\nu}\tilde{\rho}(i-\hat{k}_0)\right) - \right.$$
$$\left. - \frac{\hat{\nu}(k,\ j)^2}{2} \sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)^{\mathrm{T}}\ \boldsymbol{Q}^{-1}\tilde{\boldsymbol{\rho}}\,(i-j) \right\}, \tag{7.72}$$

where $\hat{\nu}$ stands for $\hat{\nu}(\hat{k}_0 + M - 1,\ \hat{k}_0)$. □

The method will be illustrated in Sect. 7.3.3, as a part of a fault detection and estimation system.

Note that, in this section only two possible distributions are considered for the vector data sample $\boldsymbol{z}(i)$. However, when dealing with a residual vector, a different distribution will typically be associated to each faulty mode. Fault detection and isolation then consists of detecting a change from the fault free situation, and fault isolation aims at determining the most likely fault hypothesis. The corresponding multi-hypotheses decision-making problem is addressed in the next subsection.

7.2.6 Sequential Change Detection and Isolation: The Vector Case

Introduction. An offline statistical approach will first be presented to explain the reasoning for solving the problem. Next, a recursive algorithm inspired by this solution will be described. We refer the reader to the bibliographical note for the theoretical properties of the offline and the recursive algorithms.

Problem statement. Consider a sequence of n_z-dimensional independent random vectors $\boldsymbol{z}(1), \ldots, \boldsymbol{z}(k)$ with probability density function $p_\theta(\boldsymbol{z})$ depending on the vector parameter θ. Choose at time instant k between the hypotheses:

- $\mathcal{H}_0$: $\theta = \theta_0$, $(1 \le i \le k)$

- $\mathcal{H}_\ell$: From time instant 1 up to an unknown time instant $k_0 - 1$, $\theta = \theta_0$, and for $k_0 \leq i \leq k$, $\theta = \theta_\ell$, where $\ell = 1, \ldots, n_f$.

An alarm should be issued as soon as a transition to one of the hypotheses $\mathcal{H}_\ell$, $\ell = 1, \ldots, n_f$ is detected. The known vectors θ_0 and θ_ℓ, $\ell = 1, \ldots, n_f$ are all distinct.

Note that the problem of deciding for $\mathcal{H}_\ell$ can be seen as a hypothesis testing problem between hypothesis $\mathcal{H}_\ell$ and the composite hypothesis $\bar{\mathcal{H}}_\ell$ made of all other alternative hypotheses, namely: $\bar{\mathcal{H}}_\ell = \bigcup_{0 \leq q \neq \ell \leq n_f} \mathcal{H}_q$. In the same spirit as the GLR algorithm presented previously, the most likely hypothesis is chosen among the set $\{\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_{\ell-1}, \mathcal{H}_{\ell+1}, \ldots, \mathcal{H}_{n_f}\}$. Hence the generalised log-likelihood ratio for testing between $\mathcal{H}_\ell$ and $\bar{\mathcal{H}}_\ell$ can be written:

$$g_\ell(k) = \max_{1 \leq j \leq k} \sum_{i=j}^{k} \ln \frac{p_{\theta_\ell}(z(i))}{\max_{0 \leq q \neq \ell \leq n_f} p_{\theta_q}(z(i))} \tag{7.73}$$

$$= \max_{1 \leq j \leq k} \sum_{i=j}^{k} \ln \min_{0 \leq q \neq \ell \leq n_f} \frac{p_{\theta_\ell}(z(i))}{p_{\theta_q}(z(i))}$$

$$= \max_{1 \leq j \leq k} \min_{0 \leq q \neq \ell \leq n_f} \sum_{i=j}^{k} \ln \frac{p_{\theta_\ell}(z(i))}{p_{\theta_q}(z(i))} \tag{7.74}$$

The maximisation with respect to j aims at determining the most likely fault occurrence time, while the maximisation with respect to q in (7.73) aims at selecting the most likely hypothesis in the indicated set.

Let us introduce the following notation for the log-likelihood ratio between hypothesis $\mathcal{H}_\ell$ and $\mathcal{H}_q$ assuming the change time is equal to j.

$$S_j^k(\ell, q) = \sum_{i=j}^{k} \ln \frac{p_{\theta_\ell}(z(i))}{p_{\theta_q}(z(i))} \tag{7.75}$$

$$= \sum_{i=j}^{k} s_i(\ell, q) \tag{7.76}$$

Note that, this is a change of notation from the previous sections. Indeed, the arguments (ℓ, q) are added to indicate the considered pair of hypotheses and the index i is substituted for the argument $(z(i))$ in the log-likelihood associated to the ith data sample.

Expression (7.74) can be rewritten as:

$$g_\ell(k) = \max_{1 \leq j \leq k} \min_{0 \leq q \neq \ell \leq n_f} S_j^k(\ell, q) \tag{7.77}$$

The decision rule is then the following:

$$\begin{array}{l}\text{if} \quad g_\ell(k) > h \quad \text{for some} \quad \ell \in \{1, \ldots, n_f\} \quad \text{accept} \quad \mathcal{H}_\ell \\ \text{and generate an alarm,} \\ \text{else} \quad \text{accept} \quad \mathcal{H}_0.\end{array} \tag{7.78}$$

In other words, an alarm is generated for a change from θ_0 to θ_ℓ at the time instant

$$k_{a\ell} = \inf\{k \geq 1 : g_\ell(k) > h\} \tag{7.79}$$

In the above algorithm, the maximisation over j is performed over a window of increasing size as time elapses. One approach to alleviate this pitfall is to resort to a recursive algorithm that solves the same problem.

Recursive algorithm. The derivation of this algorithm is based on the following observation. Note that the log-likelihood ratio $s_i(\ell, q)$ can be deduced from $s_i(\ell, 0)$ and $s_i(q, 0)$ since

$$s_i(\ell, q) = s_i(\ell, 0) - s_i(q, 0) \tag{7.80}$$

Similarly,

$$S_j^k(\ell, q) = S_j^k(\ell, 0) - S_j^k(q, 0) \tag{7.81}$$

Hence the decision function (7.77) can be written alternatively

$$g_\ell(k) = \max_{1 \leq j \leq k} \min_{0 \leq q \neq \ell \leq n_f} \left(S_j^k(\ell, 0) - S_j^k(q, 0) \right) \tag{7.82}$$

To deduce a recursive algorithm, let us introduce the following n_f CUSUM decision functions that correspond to testing between hypotheses $\mathcal{H}_0$ and $\mathcal{H}_q$:

$$\begin{array}{c}\bar{g}_{q0}(k) = \max(0, \bar{g}_{q0}(k-1) + s_k(q, 0)) \quad q = 1, \ldots, n_f \\ \text{with} \quad \bar{g}_{q0}(0) = 0\end{array} \tag{7.83}$$

By similarity with the equivalence between (7.77) and (7.82), the decision functions to decide on the occurrence of a change from θ_0 to θ_ℓ, $\ell = 1, \ldots, n_f$ are defined as

$$g_\ell(k) = \min_{0 \leq q \neq \ell \leq n_f} (\bar{g}_{\ell 0}(k) - \bar{g}_{q0}(k)) \tag{7.84}$$

where $\bar{g}_{00}(k) = 0$. An alarm for a change to $\theta = \theta_\ell$ is issued when $g_\ell(k) > h$, where h is a user defined threshold, indicating that hypothesis $\mathcal{H}_\ell$ holds true. Otherwise $\mathcal{H}_0$ is decided.

The algorithm is now particularised to normally distributed random vectors as this situation will be encountered when processing residual vectors.

Problem 7.5 (*Change detection and isolation in the mean of a Gaussian sequence*) Consider a data sequence $\{z(1), \ldots, z(k)\}$ of independent samples with Gaussian

probability density that depends on the mean vector μ. Choose at time instant k between the following hypotheses:

- $\mathcal{H}_0$: $\mathcal{L}(\boldsymbol{z}(i)) = \mathcal{N}(0, \boldsymbol{Q})$, $(1 \leq i \leq k)$
- $\mathcal{H}_\ell$: From time instant 1 up to an unknown time instant $k_0 - 1$, $\mathcal{L}(\boldsymbol{z}(i)) = \mathcal{N}(0, \boldsymbol{Q})$, and for $k_0 \leq i \leq k$, $\mathcal{L}(z(i)) = \mathcal{N}(\boldsymbol{\mu}_\ell, \boldsymbol{Q})$, $(\ell = 1, \ldots, n_f)$.

An alarm should be issued as soon as a transition to one of the hypotheses $\mathcal{H}_\ell$, $(\ell = 1, \ldots, n_f)$ is detected. The known vectors μ_ℓ, $(\ell = 1, \ldots, n_f)$ are all distinct.

In this case, $s_k(q, 0)$ takes the following form:

$$s_k(q, 0) = \boldsymbol{\mu}_q^{\mathrm{T}} \boldsymbol{Q}^{-1} (\boldsymbol{z}(k) - \frac{1}{2} \boldsymbol{\mu}_q) \tag{7.85}$$

and the change detection/isolation problem can be directly solved by substituting this expression in (7.83) and using the decision functions (7.84).

The reinitialisation of the algorithm after the occurrence of an alarm is now addressed, as well as the tuning of the threshold h.

Practical issues. Reinitialisation of the algorithm upon occurrence of an alarm should be performed by resetting all CUSUM decision functions to zero in (7.83). Indeed, the approach based on inverting the likelihood ratio used in the case of fault detection would be quite cumbersome to extend to this multi-hypotheses situation.

Let us now turn to the tuning of the threshold h. To this end, the notion of Kullabck Leibler information between two probability density functions say $p_{\boldsymbol{\mu}_q}(\boldsymbol{z})$ and $p_{\boldsymbol{\mu}_\ell}(\boldsymbol{z})$ is useful. It is defined as

$$\begin{aligned} \kappa_{q,\ell} &= \int p_{\boldsymbol{\mu}_q}(\boldsymbol{z}) \ln \frac{p_{\boldsymbol{\mu}_q}(\boldsymbol{z})}{p_{\boldsymbol{\mu}_\ell}(\boldsymbol{z})} d\boldsymbol{z} \\ &= E_q(s_i(q, \ell)) \qquad q \neq \ell, \quad q, \ell \in \{0, \ldots, n_f\} \end{aligned}$$

Notice that for the particular case where the log-likelihood ratio is associated to a change in the mean of Gaussian distributions, $\kappa_{q,\ell}$ takes a very simple form, namely:

$$\kappa_{q,\ell} = \frac{1}{2} (\boldsymbol{\mu}_q - \boldsymbol{\mu}_\ell)^{\mathrm{T}} \boldsymbol{Q}^{-1} (\boldsymbol{\mu}_q - \boldsymbol{\mu}_\ell) \tag{7.86}$$

Assuming that $\kappa_{q,\ell}$ is bounded and non-zero for $q \neq \ell$ with $q, \ell \in \{0, \ldots, n_f\}$, an asymptotic upper bound for the mean detection/isolation delay, $\bar{\tau}_\ell$, for a change from $\boldsymbol{\mu} = \boldsymbol{\mu}_0$ to $\boldsymbol{\mu} = \boldsymbol{\mu}_\ell$ has been computed. More precisely, it has been proved that, for a sufficiently large threshold (theoretically as h tends to infinity),

$$\bar{\tau}_\ell \leq \gamma_\ell \qquad \text{with} \quad \gamma_\ell = \frac{h}{\min_{q \neq \ell} \kappa_{q,\ell}} \tag{7.87}$$

where the minimum is taken over all $q, \ell \in \{0, \ldots, n_f\}$ with $q \neq \ell$.

Note that, in the original work of I. Nikiforov (see reference section), separate detection and an isolation thresholds are considered. They are assumed to be identical here for the sake of simplicity, but this limits the flexibility in setting the detection/isolation delay.

We can now summarise the algorithm to solve Problem 7.5. To this end, let us introduce the notations $\mathcal{Z}_0 = \{z_0(1), \ldots, z_0(N_0)\}$ a set of zero mean data corresponding to hypothesis $\mathcal{H}_0$, and $\boldsymbol{\mu}_\ell$, $\ell = 1, \ldots, n_f$, the mean of the vector signal under hypothesis $\mathcal{H}_\ell$. The latter corresponds to twice the minimum magnitude of the change to be detected and isolated, or to the most likely magnitude of this change.

Algorithm 7.8 *Multi-CUSUM algorithm for change detection and isolation in a Gaussian vector sequence*

Given: A data set $\mathcal{Z}_0$, changes in the mean $\boldsymbol{\mu}_\ell$, $\ell = 1, \ldots, n_f$ and a specified maximum mean detection/isolation delay $\bar{\tau}_{d/i}$.

Initialisation:

1. Compute an estimate of the covariance matrix $\boldsymbol{Q}$, $\hat{\boldsymbol{Q}}$, from $\mathcal{Z}_0$.

2. Choose h such that $\max_{\ell=1,\ldots,n_f} \gamma_\ell < \bar{\tau}_{d/i}$ where γ_ℓ is defined in (7.87)

3. Set $g_{q0}(0) = 0, \quad q = 1, \ldots, n_f$

At each sampling time:

R1. Acquire the new data $z(k)$.

R2. Compute $\bar{g}_{q0}(k)$, $q = 1, \ldots, n_f$ from (7.83) and $g_\ell(k)$, $\ell = 1, \ldots, n_f$ from (7.84)

R3. If $g_\ell(k) > h$, generate an alarm for hypothesis $\mathcal{H}_\ell$ by setting $k_{a\ell} = k$

Reinitialisation:

1. Reset $\bar{g}_{q0}(k_{a\ell})$, $q = 1, \ldots, n_f$ to zero.

2. Restart the recursive algorithm with step R1

Result: A sequence of alarm time instants $\{k_{a\ell_1}, k_{a\ell_2}, \ldots, k_{a\ell_i}, \ldots\}$ with $\ell_i \in \{1, \ldots, n_f\}$ for the increasing time horizon k.

Remark 7.7 (*Accounting for a dynamical profile of change*) The method can be extended to account for a known dynamical profile of change as considered in Problem 7.3. The interested reader is referred to the reference section for more information. □

7.3 Kalman Filter Approach to Diagnosis

After a presentation of the model of the supervised process, the problems of detection, isolation and estimation of additive faults in a stochastic system will be successively considered in this section.

7.3.1 Model

Let us consider a system described by a linear discrete-time model of the form

$$\begin{array}{ll} \boldsymbol{x}(k+1) & = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{F}_{\mathrm{d}}\,\boldsymbol{d}(k) + \boldsymbol{F}_f\,\boldsymbol{f}(k) + \boldsymbol{w}(k) \\ \boldsymbol{x}(0) & = \boldsymbol{x}_0 \\ \boldsymbol{y}(k) & = \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) + \boldsymbol{E}_{\mathrm{d}}\,\boldsymbol{d}(k) + \boldsymbol{E}_f\,\boldsymbol{f}(k) + \boldsymbol{v}(k), \end{array} \tag{7.88}$$

where $\boldsymbol{x} \in |\mathcal{R}^n$, $\boldsymbol{u} \in |\mathcal{R}^m$, $\boldsymbol{y} \in |\mathcal{R}^p$ are respectively the state vector, the vector of known input signals and the vector of measured output signals, $\boldsymbol{w}$ is the vector of state noise, $\boldsymbol{v}$ denotes the measurement noise. $\boldsymbol{w}(k)$ and $\boldsymbol{v}(k)$ are samples of vector white noise sequences with zero mean and covariance matrix:

$$E\left[\begin{pmatrix} \boldsymbol{w}(k) \\ \boldsymbol{v}(k) \end{pmatrix}\left(\boldsymbol{w}(\ell)^{\mathrm{T}}\boldsymbol{v}(\ell)^{\mathrm{T}}\right)\right] = \begin{pmatrix} \boldsymbol{Q}_w & \boldsymbol{Q}_{wv} \\ \boldsymbol{Q}_{wv}^{\mathrm{T}} & \boldsymbol{Q}_v \end{pmatrix}\delta_{k\ell}.$$

$\boldsymbol{x}_0$ is a stochastic vector with mean $\boldsymbol{m}_0$ and variance $\boldsymbol{\Pi}_0$ uncorrelated with the state and measurement noise sequences. Finally, $\boldsymbol{d} \in |\mathcal{R}^{n_{\mathrm{d}}}$ is a vector of unknown input signals or disturbances (deterministic or stochastic with non-zero mean), and $\boldsymbol{f} \in |\mathcal{R}^{n_f}$ is a vector of unknown input signals representing the faults to be detected. The faults are said to be additive, since they enter linearly in the model as additional input.

Such a model can also be written in terms of a single vector white noise sequence, with variance equal to the identity matrix by considering the factorisation

$$\begin{pmatrix} \boldsymbol{Q}_w & \boldsymbol{Q}_{wv} \\ \boldsymbol{Q}_{wv}^{\mathrm{T}} & \boldsymbol{Q}_v \end{pmatrix} = \begin{pmatrix} \boldsymbol{B}_\epsilon \\ \boldsymbol{D}_\epsilon \end{pmatrix}\begin{pmatrix} \boldsymbol{B}_\epsilon^{\mathrm{T}} & \boldsymbol{D}_\epsilon^{\mathrm{T}} \end{pmatrix}.$$

A sample of this sequence will be denoted $\boldsymbol{\epsilon}(k)$, hence the index in $\boldsymbol{B}_\epsilon$ and $\boldsymbol{D}_\epsilon$. It is a n_ϵ-dimensional random vector, where n_ϵ is the rank of the variance of $(\boldsymbol{w}(k)^{\mathrm{T}} \quad \boldsymbol{v}(k)^{\mathrm{T}})^{\mathrm{T}}$, generally equal to $n + p$. The state-space model (7.88) can thus be rewritten as

$$\begin{array}{ll} \boldsymbol{x}(k+1) & = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{F}_{\mathrm{d}}\,\boldsymbol{d}(k) + \boldsymbol{F}_f\,\boldsymbol{f}(k) + \boldsymbol{B}_\epsilon\boldsymbol{\epsilon}(k) \\ \boldsymbol{x}(0) & = \boldsymbol{x}_0 \\ \boldsymbol{y}(k) & = \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) + \boldsymbol{E}_{\mathrm{d}}\,\boldsymbol{d}(k) + \boldsymbol{E}_f\,\boldsymbol{f}(k) + \boldsymbol{D}_\epsilon\boldsymbol{\epsilon}(k). \end{array} \tag{7.89}$$

7.3.2 Fault Detection

Problem statement. Fault detection amounts to determining whether the supervised process is working in a normal (or healthy) operating mode. The problem can be stated as follows:

Problem 7.6 (*Fault detection*) Given

- a model of the process of the form (7.88) or (7.89)
- a sequence of measured process input and output $(\boldsymbol{y}(i), \boldsymbol{u}(i))_{1 \le i \le k}$, where k denotes the current time instant.

Choose between the following two hypotheses:

$\mathcal{H}_0$: healthy operating condition,
$\mathcal{H}_1$: faulty operating condition.

The quality of a fault detection system is measured in terms of detection delay and time between false alarms. A typical objective is to minimise the mean delay for detection of a change subject to a fixed mean time between false alarms before the change time.

To achieve this goal, the task is usually divided into two parts: residual generation and residual evaluation. Each of them is addressed successively in the following subsections:

Residual generation. As in the deterministic case, the residual generators are filters with input signals $\boldsymbol{u}$ and $\boldsymbol{y}$, belonging to the following class of linear time-invariant systems

$$\begin{aligned} z(k+1) &= \boldsymbol{A}_z\, z(k) + \boldsymbol{B}_{zu}\, \boldsymbol{u}(k) + \boldsymbol{B}_{zy}\, \boldsymbol{y}(k), \quad z(0) = z_0 \\ \boldsymbol{r}(k) &= \boldsymbol{C}_{rz}\, z(k) + \boldsymbol{D}_{ru}\, \boldsymbol{u}(k) + \boldsymbol{D}_{ry}\, \boldsymbol{y}(k) \end{aligned} \tag{7.90}$$

or, in transfer function form, after taking the z-transform of the above equations and assuming zero initial conditions:

$$\begin{aligned} \boldsymbol{r}(z) &= \boldsymbol{V}_{ru}(z)\boldsymbol{u}(z) + \boldsymbol{V}_{ry}(z)\boldsymbol{y}(z) \\ &= \begin{pmatrix} \boldsymbol{V}_{ru}(z) & \boldsymbol{V}_{ry}(z) \end{pmatrix} \begin{pmatrix} \boldsymbol{u}(z) \\ \boldsymbol{y}(z) \end{pmatrix}. \end{aligned} \tag{7.91}$$

The problem of designing a residual generator can be stated as follows:

Problem 7.7 (*Residual generator design for fault detection*) Determine a stable linear time-invariant filter (7.90) or (7.91) such that:

1. The sequence of output values $\boldsymbol{r}(k), k = 1, 2, \ldots$ is a zero mean white noise vector sequence (which is not affected by $\boldsymbol{u}$ and $\boldsymbol{d}$), once the transient due to initial conditions has vanished.

2. In the presence of a fault ($\boldsymbol{f}(k) \neq 0$ for all $k \geq k_0$), the mean of $\boldsymbol{r}(k)$ is different from zero for at least some $k \geq k_0$.

As $\boldsymbol{u}$ and $\boldsymbol{d}$ are arbitrary signals, they cannot affect $\boldsymbol{r}$ for the latter to be a white noise sequence. To define rigorously what is meant by this statement, notice that the global system made of the supervised process and the residual generator, obtained by combining Eqs. (7.89) and (7.90), is seen to have as input signals $\boldsymbol{u}, \boldsymbol{d}, \boldsymbol{f}, \boldsymbol{\epsilon}$, and state $(\boldsymbol{x}\ \ \boldsymbol{z})^{\mathrm{T}}$. Hence the residual at time k can be considered as a function of the above input and the initial state, namely:

$$\boldsymbol{r}(k) = \boldsymbol{r}(k;\ \boldsymbol{u},\ \boldsymbol{d},\ \boldsymbol{f},\ \boldsymbol{\epsilon};\ \boldsymbol{x}_0,\ z_0).$$

Saying that the residual is not affected by $\boldsymbol{u}$ and $\boldsymbol{d}$ means that, for any two distinct input sequences $\boldsymbol{u}^1(k), \boldsymbol{u}^2(k)$ and $\boldsymbol{d}^1(k), \boldsymbol{d}^2(k), k = 1, 2, \ldots,$

$$\boldsymbol{r}\left(k;\ \boldsymbol{u}^1,\ \boldsymbol{d},\ \boldsymbol{f},\ \boldsymbol{\epsilon};\ \boldsymbol{x}_0,\ z_0\right) = \boldsymbol{r}\left(k;\ \boldsymbol{u}^2,\ \boldsymbol{d},\ \boldsymbol{f},\ \boldsymbol{\epsilon};\ \boldsymbol{x}_0,\ z_0\right)$$

and

$$\boldsymbol{r}\left(k;\ \boldsymbol{u},\ \boldsymbol{d}^1,\ \boldsymbol{f},\ \boldsymbol{\epsilon};\ \boldsymbol{x}_0,\ z_0\right) = \boldsymbol{r}\left(k;\ \boldsymbol{u},\ \boldsymbol{d}^2,\ \boldsymbol{f},\ \boldsymbol{\epsilon};\ \boldsymbol{x}_0,\ z_0\right),$$

whatever the initial state and the input sequences.

One way to solve the problem is to design a filter which meets the first condition in Problem 7.7, and then to check whether the second requirement is fulfilled. In order to maximise the chances for this second condition to hold, one should make sure that, when imposing condition 1, no useful information contained in $\boldsymbol{y}$ is lost. Only the information corrupted by an unknown input should be cancelled. A filter that meets the latter condition, together with the first condition of Problem 7.7 is called an innovation filter for reasons that will be clarified in the next subsection.

To construct a residual generator, one will first solve the innovation filter design problem below. Next fault sensitivity of the filter output will be checked to see whether condition 2 is met in Problem 7.7. In the affirmative, the innovation filter is a residual generator. The issue of fault sensitivity is the object of a specific subsection.

Problem 7.8 (*Innovation filter design*) Determine a stable linear time-invariant filter (7.91) with the least number of output signals such that:

1. In the absence of fault (i.e. $\boldsymbol{f}(k) = 0$ for all k), the sequence of output values $\boldsymbol{r}(k), k = 1, 2, \ldots$ is a zero mean white noise vector sequence which is not affected by $\boldsymbol{u}$ and $\boldsymbol{d}$, once the transient due to initial conditions has vanished.
2. No information on the fault contained in $\boldsymbol{y}$ is lost, except if it is affected by the unknown input vector $\boldsymbol{d}$.

An observer-based approach will be used to solve the Problem 7.8. Two situations have to be distinguished, namely the absence of unknown input ($\boldsymbol{E}_{\mathrm{d}} = \boldsymbol{O}$ and $\boldsymbol{F}_{\mathrm{d}} = \boldsymbol{O}$) and the presence of unknown input.

No unknown input. In this situation the design of an innovation filter amounts to the design of a steady state Kalman filter. Such a filter provides a prediction of the output $\boldsymbol{y}(k)$, $\hat{\boldsymbol{y}}(k)$, given the data up to time $k-1$, namely, given $\boldsymbol{u}(i)$, $\boldsymbol{y}(i)$, $i = 1, 2, \ldots, k-1$. The output prediction error, $\boldsymbol{r}(k) = \boldsymbol{y}(k) - \hat{\boldsymbol{y}}(k)$ is called the innovation in standard Kalman filter literature, because it consists of the new information contained in $\boldsymbol{y}(k)$, which was not available in $\boldsymbol{y}(1), \ldots, \boldsymbol{y}(k-1)$. The innovation sequence is known to be a white noise sequence not affected by $\boldsymbol{u}$ (once the transient due to initial conditions has decayed to zero). Hence it fulfils condition 1 of Problem 7.8. Besides, the information about $\boldsymbol{f}$ contained in the sequence of data is also contained in the innovation sequence. For this reason, the innovation is said to be a sufficient statistics for the fault vector $\boldsymbol{f}$. Thus condition 2 of Problem 7.8 is also fulfilled by the innovation sequence, and hence the latter is a suitable candidate as a residual sequence. It is the fact that the innovation sequence meets conditions 1 and 2 of Problem 7.8 that justifies the name innovation filter.

To state the design procedure precisely, let us introduce the notion of a regular quadruple $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$.

Definition 7.1 (*Regular quadruple*) The quadruple $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ is regular if the matrix

$$\begin{pmatrix} -z\boldsymbol{I} + \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix}$$

has full row rank for all z on the unit circle ($z = \exp(\mathrm{j}\omega)$, $\omega \in |\mathcal{R}$).

It is assumed below that the pair $(\boldsymbol{C}, \boldsymbol{A})$ is detectable, and $(\boldsymbol{A}, \boldsymbol{B}_\epsilon, \boldsymbol{C}, \boldsymbol{D}_\epsilon)$ is regular.

Remark 7.8 (*Uncorrelated state and measurement noise sequences*) In the particular case where the sequence $\boldsymbol{w}(k)$ and $\boldsymbol{v}(k)$ are uncorrelated (which amounts to $\boldsymbol{B}_\epsilon \boldsymbol{D}_\epsilon^{\mathrm{T}} = \boldsymbol{O}$), the above regularity assumption can be replaced by the classical requirement that the pair $(\boldsymbol{A}, \boldsymbol{B}_\epsilon)$ has no uncontrollable mode on the unit circle. □

Under such hypotheses, the equations for the steady state Kalman filter can be written as

$$\begin{aligned} \hat{\boldsymbol{x}}(k+1) &= \boldsymbol{A}\hat{\boldsymbol{x}}(k) + \boldsymbol{B}\boldsymbol{u}(k) - \boldsymbol{K}(\boldsymbol{y}(k) - \boldsymbol{C}\hat{\boldsymbol{x}}(k) - \boldsymbol{D}\boldsymbol{u}(k)), \\ \hat{\boldsymbol{x}}(0) &= \hat{\boldsymbol{x}}_0 \\ \boldsymbol{r}(k) &= \boldsymbol{y}(k) - \boldsymbol{C}\hat{\boldsymbol{x}}(k) - \boldsymbol{D}\boldsymbol{u}(k), \end{aligned} \tag{7.92}$$

where the filter gain $\boldsymbol{K}$ is given by

$$\boldsymbol{K} = -(\boldsymbol{A}\boldsymbol{P}\boldsymbol{C}^{\mathrm{T}} + \boldsymbol{Q}_{wv})(\boldsymbol{Q}_v + \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^{\mathrm{T}})^{-1}, \tag{7.93}$$

$\boldsymbol{P}$ being the symmetric semi-positive definite solution of the following discrete algebraic Riccati equation

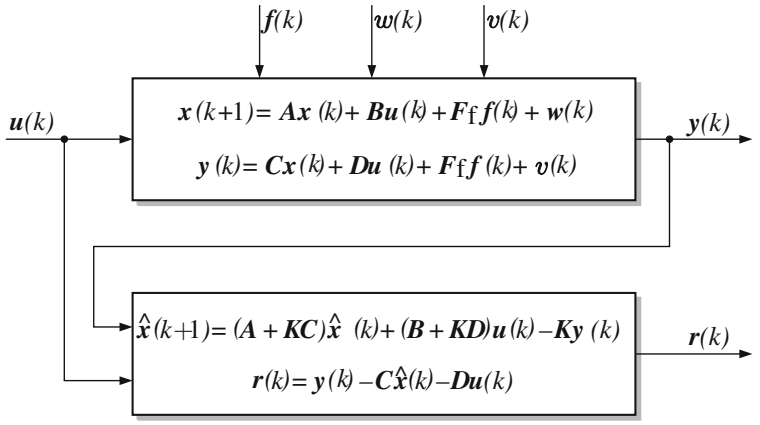


Fig. 7.10 Block diagram of the supervised system together with the innovation filter

$$\begin{aligned} \boldsymbol{P} = \boldsymbol{A}\boldsymbol{P}\boldsymbol{A}^{\mathrm{T}} - (\boldsymbol{A}\boldsymbol{P}\boldsymbol{C}^{\mathrm{T}} + \boldsymbol{Q}_{wv})(\boldsymbol{Q}_v + \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^{\mathrm{T}})^{-1} \cdot \\ \cdot (\boldsymbol{C}\boldsymbol{P}\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{Q}_{wv}^{\mathrm{T}}) + \boldsymbol{Q}_w. \end{aligned} \tag{7.94}$$

Figure 7.10 illustrates the state-space implementation of the innovation filter.

In a transfer function form, this filter is described by

$$\begin{aligned} \boldsymbol{r}(z) &= \boldsymbol{V}_{\mathrm{ru}}(z)\boldsymbol{u}(z) + \boldsymbol{V}_{\mathrm{ry}}(z)\boldsymbol{y}(z) \\ &= (-\boldsymbol{C}(z\boldsymbol{I} - \boldsymbol{A} - \boldsymbol{K}\boldsymbol{C})^{-1}(\boldsymbol{B} + \boldsymbol{K}\boldsymbol{D}) - \boldsymbol{D})\boldsymbol{u}(z) + \\ &\quad + (\boldsymbol{C}(z\boldsymbol{I} - \boldsymbol{A} - \boldsymbol{K}\boldsymbol{C})^{-1}\boldsymbol{K} + \boldsymbol{I})\boldsymbol{y}(z). \end{aligned} \tag{7.95}$$

If the pair $(\boldsymbol{C},\ \boldsymbol{A})$ is not detectable, it is still possible to design an innovation filter by extracting the observable part of system (7.89) and designing a Kalman filter for this observable subsystem. Note that this approach can also be considered when (7.89) is detectable but not observable, if one wishes to obtain a residual generator with the lowest possible order.

The following two remarks present other forms of Kalman filter that may prove useful for residual generation.

Remark 7.9 (*Time-varying Kalman filter*) To assure coherency with Sect. 7.2.5 and to ease the study of the sensitivity to the fault, a steady state Kalman filter is considered above. This implies that whiteness of the sequence $\boldsymbol{r}(k)$ is only reached after the transient has vanished. A white noise sequence can be generated from time $k = 0$, if a (time-varying) Kalman filter is used instead of a steady state Kalman filter. Equations (7.92), (7.93) and (7.94) are then replaced by

$$\begin{aligned} \hat{\boldsymbol{x}}(k+1) &= \boldsymbol{A}\hat{\boldsymbol{x}}(k) + \boldsymbol{B}\boldsymbol{u}(k) - \boldsymbol{K}(k)(\boldsymbol{y}(k) - \boldsymbol{C}\hat{\boldsymbol{x}}(k) - \boldsymbol{D}\boldsymbol{u}(k)) \\ \hat{\boldsymbol{x}}(0) &= \boldsymbol{m}_0 \\ \boldsymbol{r}(k) &= \boldsymbol{y}(k) - \boldsymbol{C}\hat{\boldsymbol{x}}(k) - \boldsymbol{D}\boldsymbol{u}(k), \end{aligned} \tag{7.96}$$

where the filter gain $\boldsymbol{K}(k)$ is given by

$$\boldsymbol{K}(k) = -(\boldsymbol{A}\boldsymbol{P}(k)\boldsymbol{C}^{\mathrm{T}} + \boldsymbol{Q}_{wv})(\boldsymbol{Q}_v + \boldsymbol{C}\boldsymbol{P}(k)\boldsymbol{C}^{\mathrm{T}})^{-1}, \tag{7.97}$$

$\boldsymbol{P}(k)$ being the solution of the following discrete Riccati equation

$$\begin{aligned}\boldsymbol{P}(k+1) = {} & \boldsymbol{A}\boldsymbol{P}(k)\boldsymbol{A}^{\mathrm{T}} - (\boldsymbol{A}\boldsymbol{P}(k)\boldsymbol{C}^{\mathrm{T}} + \boldsymbol{Q}_{wv})(\boldsymbol{Q}_v + \boldsymbol{C}\boldsymbol{P}(k)\boldsymbol{C}^{\mathrm{T}})^{-1} \cdot \\ & \cdot (\boldsymbol{C}\boldsymbol{P}(k)\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{Q}_{wv}^{\mathrm{T}}) + \boldsymbol{Q}_w, \qquad \boldsymbol{P}(0) = \boldsymbol{\Pi}_0. \ \square\end{aligned} \tag{7.98}$$

For implementation purpose, it is interesting to separate the equations of the Kalman filter in a two-stage update procedure at each sampling time: a measurement update and a time update. The approach is the object of the following remark. This implementation of the Kalman filter allows one to handle missing measurements in a straightforward way.

Remark 7.10 (*Handling missing measurements*) The measurement update consists of taking into account the new information brought by an additional measurement, say $\boldsymbol{y}(k)$, in order to compute $\hat{\boldsymbol{x}}(k|k)$, the best estimate (in the least square sense) of $\boldsymbol{x}(k)$ given $\boldsymbol{u}(i)$, $\boldsymbol{y}(i)$, $i = 1, 2, \ldots, k$. The latter is deduced from $\boldsymbol{u}(k)$, $\boldsymbol{y}(k)$ and from $\hat{\boldsymbol{x}}(k)$ the best prediction of $\boldsymbol{x}(k)$ given $\boldsymbol{u}(i)$, $\boldsymbol{y}(i)$, $i = 1, 2, \ldots, k-1$. The time update then uses the plant model in order to predict the state evolution one step ahead.

An additional hypothesis is needed to use the algorithm below: the variance of the measurement noise, $\boldsymbol{Q}_v$ should be positive definite and diagonal. Thus, $\boldsymbol{Q}_v = \text{diag}\,(q_{v,1} \ldots q_{v,p})$, where $q_{vi} > 0$, $i = 1, \ldots, p$. The diagonal form can be enforced by a suitable change of output variable when $\boldsymbol{Q}_v$ is positive definite. It suffices to set $\boldsymbol{y}_{\mathrm{d}}(k) = \boldsymbol{Q}_v^{-1/2}\boldsymbol{y}(k)$, so that the variance of $\boldsymbol{y}_{\mathrm{d}}(k)$ is the $p \times p$ identity matrix.

The following notations are introduced in the algorithm below: $\boldsymbol{c}_i$ and $\boldsymbol{d}_i$ denote, respectively, the ith row of $\boldsymbol{C}$ and $\boldsymbol{D}$.

Algorithm 7.9 *Measurement and time update for the innovation filter*

Initialisation: Set $\boldsymbol{P}(0) = \boldsymbol{\Pi}_0$, $\hat{\boldsymbol{x}}(0) = \boldsymbol{m}_0$.

At each sampling time:

1 Measurement update.

Set $\boldsymbol{P}_0(k) = \boldsymbol{P}(k)$, $\hat{\boldsymbol{x}}_0(k|k) = \hat{\boldsymbol{x}}(k)$.

For $i = 1$ up to p, compute $\boldsymbol{P}_i(k|k)^{-1} = \boldsymbol{P}_{i-1}(k)^{-1} + \boldsymbol{c}_i^{\mathrm{T}}\boldsymbol{c}_i/q_{v,i}$.

Set $\boldsymbol{P}(k|k) = \boldsymbol{P}_p(k|k)$.

For $i = 1$ up to p, compute

$$\boldsymbol{K}_{f,i}(k) = \boldsymbol{P}(k|k)\boldsymbol{c}_i^{\mathrm{T}}/q_{v,i}$$
$$\hat{\boldsymbol{x}}_i(k|k) = \hat{\boldsymbol{x}}_{i-1}(k|k) + \boldsymbol{K}_{f,i}(k)(y_i(k) - \boldsymbol{c}_i\hat{\boldsymbol{x}}(k) - \boldsymbol{d}_i\boldsymbol{u}(k)).$$

Set $\hat{\boldsymbol{x}}(k|k) = \hat{\boldsymbol{x}}_p(k|k)$.

2 Time update.
Compute successively

$$\begin{aligned}
\boldsymbol{K}_f(k) &= \boldsymbol{P}(k|k)\boldsymbol{C}^{\mathrm{T}}\boldsymbol{Q}_v^{-1} \\
\boldsymbol{P}(k+1) &= \boldsymbol{A}\boldsymbol{P}(k|k)\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{Q}_w - \boldsymbol{Q}_{wv}(\boldsymbol{Q}_v + \boldsymbol{C}\boldsymbol{P}(k)\boldsymbol{C}^{\mathrm{T}})^{-1}\boldsymbol{Q}_{wv}^{\mathrm{T}} \\
&\quad -\boldsymbol{A}\boldsymbol{K}_f(k)\boldsymbol{Q}_{wv}^{\mathrm{T}} - \boldsymbol{Q}_{wv}\boldsymbol{K}_f(k)^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}} \\
\hat{\boldsymbol{x}}(k+1) &= \boldsymbol{A}\hat{\boldsymbol{x}}(k|k) + \boldsymbol{Q}_{wv}(\boldsymbol{Q}_v + \boldsymbol{C}\boldsymbol{P}(k)\boldsymbol{C}^{\mathrm{T}})^{-1} \cdot \\
&\quad \cdot(\boldsymbol{y}(k) - \boldsymbol{C}\hat{\boldsymbol{x}}(k) - \boldsymbol{D}\boldsymbol{u}(k)).
\end{aligned}$$

3 Computation of the residual.
For $i = 1$ up to p, compute the components of the residual vector

$$r_i(k) = y_i(k) - \boldsymbol{c}_i\hat{\boldsymbol{x}}(k) - \boldsymbol{d}_i\boldsymbol{u}(k).$$

Result: Residual vector $\boldsymbol{r}(k)$ for increasing time horizon k.

When a measurement is missing, it suffices to skip the corresponding measurement update, namely to skip the corresponding value of index i in the "for" loops. □

With unknown input. In this case, the design of an innovation filter consists of a two-step procedure. First, a reduced system having no unknown input is extracted from the original system. Then a steady state Kalman filter is designed for this subsystem and the candidate residual signal is nothing but the innovation associated to this filter. As above, to check whether the innovation sequence is a residual, its sensitivity to the fault vector $\boldsymbol{f}$ has to be verified, which is the object of the next subsection.

The idea behind the extraction of the subsystem will first be sketched in the case, where $\boldsymbol{E}_{\mathrm{d}} = \boldsymbol{O}$ (no unknown input affecting $\boldsymbol{y}$). Next a complete algorithm will be provided to solve Problem 7.8. The justification of this algorithm is relatively involved, and the interested reader is invited to consult the bibliography for the proofs.

To extract a subsystem which has not $\boldsymbol{d}$ as input, let $\boldsymbol{x}_{\mathrm{sub}}$ denote the state of this subsystem and set

$$\boldsymbol{x}_{\mathrm{sub}}(k) = \boldsymbol{\Pi}\boldsymbol{x}(k), \tag{7.99}$$

where $\boldsymbol{\Pi}$ is an $n_{\mathrm{sub}} \times n$ matrix (with $n_{\mathrm{sub}} \leq n$) to be determined. By multiplying the first Eq. (7.89) by $\boldsymbol{\Pi}$ on the left, and by taking (7.99) into account, one gets

$$\begin{aligned}
\boldsymbol{x}_{\mathrm{sub}}(k+1) &= \boldsymbol{\Pi}\boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{\Pi}\boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{\Pi}\boldsymbol{F}_{\mathrm{d}}\boldsymbol{d}(k) \\
&\quad +\boldsymbol{\Pi}\boldsymbol{F}_f\boldsymbol{f}(k) + \boldsymbol{\Pi}\boldsymbol{B}_\epsilon\boldsymbol{\epsilon}(k).
\end{aligned} \tag{7.100}$$

If the following relations are imposed

$$\boldsymbol{\Pi A} = \bar{\boldsymbol{A}}\boldsymbol{\Pi} + \bar{\boldsymbol{B}}\boldsymbol{C} \tag{7.101}$$

$$\boldsymbol{\Pi F}_{\mathrm{d}} = \boldsymbol{O}, \tag{7.102}$$

where $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{B}}$ are unknown matrices to be determined, then (7.100) can be written as

$$\begin{aligned}\boldsymbol{x}_{\mathrm{sub}}(k+1) = \bar{\boldsymbol{A}}\boldsymbol{x}_{\mathrm{sub}}(k) + \bar{\boldsymbol{B}}\left(\boldsymbol{y}(k) - \boldsymbol{D}\boldsymbol{u}(k) - \boldsymbol{E}_f\,\boldsymbol{f}(k) - \boldsymbol{D}_\epsilon\,\boldsymbol{\epsilon}(k)\right)\\ +\boldsymbol{\Pi B}\boldsymbol{u}(k) + \boldsymbol{\Pi F}_f\,\boldsymbol{f}(k) + \boldsymbol{\Pi B}_\epsilon\boldsymbol{\epsilon}(k)\end{aligned} \tag{7.103}$$

by using (7.99) and the output of Eq. (7.89) (in which $\boldsymbol{E}_{\mathrm{d}}$ is assumed to be null). Introducing the abbreviations

$$\begin{aligned}\tilde{\boldsymbol{B}} &= \boldsymbol{\Pi B} - \bar{\boldsymbol{B}}\boldsymbol{D}\\ \tilde{\boldsymbol{F}}_f &= \boldsymbol{\Pi F}_f - \bar{\boldsymbol{B}}\boldsymbol{E}_f\\ \tilde{\boldsymbol{B}}_\epsilon &= \boldsymbol{\Pi B}_\epsilon - \bar{\boldsymbol{B}}\boldsymbol{D}_\epsilon\end{aligned}$$

into (7.103) yields

$$\boldsymbol{x}_{\mathrm{sub}}(k+1) = \bar{\boldsymbol{A}}\boldsymbol{x}_{\mathrm{sub}}(k) + \tilde{\boldsymbol{B}}\boldsymbol{u}(k) + \bar{\boldsymbol{B}}\boldsymbol{y}(k) + \tilde{\boldsymbol{F}}_f\,\boldsymbol{f}(k) + \tilde{\boldsymbol{B}}_\epsilon\,\boldsymbol{\epsilon}(k). \tag{7.104}$$

This system has no unknown input $\boldsymbol{d}$ as could be expected by imposing (7.102). To design a Kalman filter based on the state Eq. (7.104) when $\boldsymbol{f} = 0$, the part of the measurement $\boldsymbol{y}$ which depends on $\boldsymbol{x}_{\mathrm{sub}}$, $\boldsymbol{u}$ and $\boldsymbol{\epsilon}$ only should be determined. This is achieved by defining the signal $\boldsymbol{y}_{\mathrm{sub}}$ as

$$\boldsymbol{y}_{\mathrm{sub}}(k) = \boldsymbol{M}\boldsymbol{y}(k) = \boldsymbol{MC}\boldsymbol{x}(k) + \boldsymbol{MD}\boldsymbol{u}(k) + \boldsymbol{MD}_\epsilon\,\boldsymbol{\epsilon}(k), \tag{7.105}$$

where $\boldsymbol{M}$ is unknown. Imposing

$$\boldsymbol{MC} = \boldsymbol{L\Pi}, \tag{7.106}$$

Equation (7.105) becomes

$$\boldsymbol{y}_{\mathrm{sub}}(k) = \boldsymbol{L}\boldsymbol{x}_{\mathrm{sub}}(k) + \boldsymbol{MD}\boldsymbol{u}(k) + \boldsymbol{MD}_\epsilon\,\boldsymbol{\epsilon}(k), \tag{7.107}$$

which has the required form.

Now, provided $(\boldsymbol{L},\ \bar{\boldsymbol{A}})$ is detectable, and $(\bar{\boldsymbol{A}},\ \tilde{\boldsymbol{B}}_\epsilon,\ \boldsymbol{L},\ \boldsymbol{MD}_\epsilon)$ is regular, a Kalman filter can be designed for the subsystem (7.104), (7.107), when $\boldsymbol{f} = 0$

$$\begin{aligned}\hat{\boldsymbol{x}}_{\mathrm{sub}}(k+1) = \bar{\boldsymbol{A}}\hat{\boldsymbol{x}}_{\mathrm{sub}}(k) + \tilde{\boldsymbol{B}}\boldsymbol{u}(k) + \bar{\boldsymbol{B}}\boldsymbol{y}(k) - \boldsymbol{K}_{\mathrm{sub}}\left(\boldsymbol{y}_{\mathrm{sub}}\right.\\ \left. - \boldsymbol{L}\hat{\boldsymbol{x}}_{\mathrm{sub}}(k) - \boldsymbol{MD}\boldsymbol{u}(k)\right),\end{aligned} \tag{7.108}$$

where

$$\boldsymbol{K}_{\text{sub}} = -\left(\bar{\boldsymbol{A}}\boldsymbol{P}_{\text{sub}}\boldsymbol{L}^{\text{T}} + \tilde{\boldsymbol{B}}_{\epsilon}\boldsymbol{D}_{\epsilon}^{\text{T}}\boldsymbol{M}^{\text{T}}\right)\left(\boldsymbol{M}\boldsymbol{D}_{\epsilon}\boldsymbol{D}_{\epsilon}^{\text{T}}\boldsymbol{M}^{\text{T}} + \boldsymbol{L}\boldsymbol{P}_{\text{sub}}\boldsymbol{L}^{\text{T}}\right)^{-1}$$

with $\boldsymbol{P}_{\text{sub}}$ the symmetric semi-positive definite solution of

$$\begin{aligned}\boldsymbol{P}_{\text{sub}} = \bar{\boldsymbol{A}}\boldsymbol{P}_{\text{sub}}\bar{\boldsymbol{A}}^{\text{T}} - \left(\bar{\boldsymbol{A}}\boldsymbol{P}_{\text{sub}}\boldsymbol{L}^{\text{T}} + \tilde{\boldsymbol{B}}_{\epsilon}\boldsymbol{D}_{\epsilon}^{\text{T}}\boldsymbol{M}^{\text{T}}\right)\left(\boldsymbol{M}\boldsymbol{D}_{\epsilon}\boldsymbol{D}_{\epsilon}^{\text{T}}\boldsymbol{M}^{\text{T}}\right.\\ \left.+\boldsymbol{L}\boldsymbol{P}_{\text{sub}}\boldsymbol{L}^{\text{T}}\right)^{-1}\left(\boldsymbol{L}\boldsymbol{P}_{\text{sub}}\bar{\boldsymbol{A}}^{\text{T}} + \boldsymbol{M}\boldsymbol{D}_{\epsilon}\tilde{\boldsymbol{B}}_{\epsilon}^{\text{T}}\right) + \tilde{\boldsymbol{B}}_{\epsilon}\tilde{\boldsymbol{B}}_{\epsilon}^{\text{T}}.\end{aligned}$$

The associated output reconstruction error is given by

$$\boldsymbol{r}(k) = \boldsymbol{y}_{\text{sub}}(k) - \boldsymbol{L}\hat{\boldsymbol{x}}_{\text{sub}}(k) - \boldsymbol{M}\boldsymbol{D}\boldsymbol{u}(k). \tag{7.109}$$

which can be evaluated from the available data. It can be checked that it fulfils the conditions for an innovation sequence. Indeed, the state estimation error, $\tilde{\boldsymbol{x}}_{\text{sub}}(k) = \boldsymbol{x}_{\text{sub}}(k) - \hat{\boldsymbol{x}}_{\text{sub}}(k)$, verifies the following equation obtained by subtracting (7.108) from (7.104):

$$\begin{aligned}\tilde{\boldsymbol{x}}_{\text{sub}}(k+1) = \left(\bar{\boldsymbol{A}} + \boldsymbol{K}_{\text{sub}}\boldsymbol{L}\right)\tilde{\boldsymbol{x}}_{\text{sub}}(k) + \tilde{\boldsymbol{F}}_{f}\boldsymbol{f}(k)\\ + \left(\tilde{\boldsymbol{B}}_{\epsilon} + \boldsymbol{K}_{\text{sub}}\boldsymbol{M}\boldsymbol{D}_{\epsilon}\right)\boldsymbol{\epsilon}(k).\end{aligned} \tag{7.110}$$

This error is clearly not affected by $\boldsymbol{d}$ and $\boldsymbol{u}$, and so is the associated innovation $\boldsymbol{r}(k) = \boldsymbol{L}\tilde{\boldsymbol{x}}_{\text{sub}}(k) + \boldsymbol{M}\boldsymbol{D}_{\epsilon}\boldsymbol{\epsilon}(k)$. Condition 1 of Problem 7.8 is thus fulfilled. To assure that the maximum amount of information on the fault has been kept (condition 2 of Problem 7.8), $\boldsymbol{x}_{\text{sub}}$ should have the largest possible dimension ($\boldsymbol{\Pi}$ should have the largest possible number of rows). The implementation of the innovation filter is summarised in the block diagram of Fig. 7.11.

The design of an innovation filter essentially amounts to solving the set of nonlinear algebraic Eqs. (7.101), (7.102), (7.106). Despite the nonlinearity, an algorithm based only on linear algebraic operations can be derived. It is presented below in the general situation, where $\boldsymbol{E}_{\text{d}} \neq \boldsymbol{O}$.

The algorithm relies on the following matrix equation:

$$\begin{aligned}\begin{pmatrix} -z\boldsymbol{I}_{n_{\text{sub}}} + \boldsymbol{A}_{\text{sub}} & \boldsymbol{B}_{\text{sub}} \\ \boldsymbol{C}_{\text{sub}} & \boldsymbol{D}_{\text{sub}} \end{pmatrix} = \boldsymbol{\Gamma}\begin{pmatrix} -z\boldsymbol{I}_{n} + \boldsymbol{A} & \boldsymbol{F}_{\text{d}} & \boldsymbol{B}_{\epsilon} \\ \boldsymbol{C} & \boldsymbol{E}_{\text{d}} & \boldsymbol{D}_{\epsilon} \end{pmatrix}\\ \cdot\begin{pmatrix} \boldsymbol{\Phi} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I}_{n_{\epsilon}} \end{pmatrix}.\end{aligned} \tag{7.111}$$

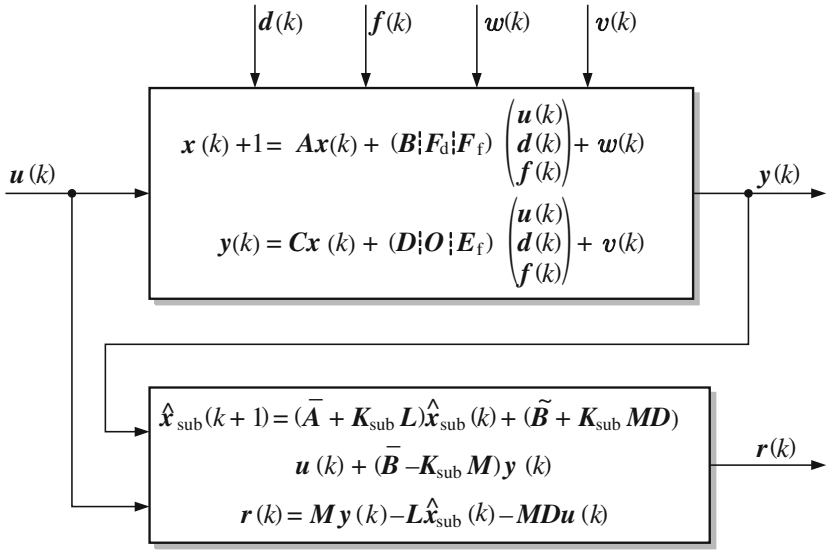


Fig. 7.11 Block diagram of the supervised system together with the innovation filter in the presence of unknown inputs

where $\boldsymbol{A}_{\text{sub}}$ and $\boldsymbol{C}_{\text{sub}}$ are respectively $n_{\text{sub}} \times n_{\text{sub}}$ and $p_{\text{sub}} \times n_{\text{sub}}$ matrices. It also involves the reduced system described by

$$\begin{aligned} \boldsymbol{x}_{\text{sub}}(k+1) &= \boldsymbol{A}_{\text{sub}}\, \boldsymbol{x}_{\text{sub}}(k) + \tilde{\boldsymbol{B}}_{\text{sub}}\, \boldsymbol{u}(k) - \boldsymbol{\Gamma}_{12}\, \boldsymbol{y}(k) \\ &\quad + \boldsymbol{B}_{\text{sub}}\, \boldsymbol{\epsilon}_{\text{sub}}(k) \qquad (7.112) \\ \boldsymbol{x}_{\text{sub}}(0) &= \boldsymbol{x}_{sub,0} \\ \boldsymbol{y}_{\text{sub}}(k) &= \boldsymbol{\Gamma}_{22}\, \boldsymbol{y}(k) = \boldsymbol{C}_{\text{sub}}\, \boldsymbol{x}_{\text{sub}}(k) + \tilde{\boldsymbol{D}}_{\text{sub}}\, \boldsymbol{u}(k) \\ &\quad + \boldsymbol{D}_{\text{sub}}\, \boldsymbol{\epsilon}_{\text{sub}}(k), \qquad (7.113) \end{aligned}$$

where $\boldsymbol{\epsilon}_{\text{sub}}(k)$ is a sample of a white noise sequence with variance equal to the identity matrix,

$$\boldsymbol{\Gamma} = \begin{pmatrix} \boldsymbol{\Gamma}_{11} & \boldsymbol{\Gamma}_{12} \\ \boldsymbol{\Gamma}_{21} & \boldsymbol{\Gamma}_{22} \end{pmatrix}$$

with $\boldsymbol{\Gamma}_{11}$, $\boldsymbol{\Gamma}_{12}$, $\boldsymbol{\Gamma}_{21}$, $\boldsymbol{\Gamma}_{22}$ respectively $n_{\text{sub}} \times n$, $n_{\text{sub}} \times p$, $p_{\text{sub}} \times n$ and $p_{\text{sub}} \times p$-dimensional matrices,

$$\begin{aligned} \tilde{\boldsymbol{B}}_{\text{sub}} &= \boldsymbol{\Gamma}_{11}\, \boldsymbol{B} + \boldsymbol{\Gamma}_{12}\, \boldsymbol{D}, \\ \tilde{\boldsymbol{D}}_{\text{sub}} &= \boldsymbol{\Gamma}_{21}\, \boldsymbol{B} + \boldsymbol{\Gamma}_{22}\, \boldsymbol{D}. \end{aligned}$$

Algorithm 7.10 *Innovation filter design for a system with unknown input*
Given: A system of the form (7.89).
Compute:

1. Determine the integer n_{sub} together with full row rank and full column rank matrices $\boldsymbol{\Gamma}$ and $\boldsymbol{\Phi}$ respectively such that Eq. (7.111) is fulfilled. The Algorithm 7.11 presented below can be used to compute n_{sub}, $\boldsymbol{\Gamma}$ and $\boldsymbol{\Phi}$. The n_{sub}-dimensional subsystem $(\boldsymbol{A}_{\text{sub}}, \boldsymbol{B}_{\text{sub}}, \boldsymbol{C}_{\text{sub}}, \boldsymbol{D}_{\text{sub}})$ has no unknown input.
2. Design a Kalman filter for the reduced system (7.112)–(7.113) The resulting innovation qualifies as a residual.

Result: An innovation filter for system (7.89).

The algorithm to be used in step 1 is presented below. It involves the singular value decomposition of several matrices which is denoted as follows for an arbitrary matrix $\boldsymbol{X}$:

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{U}_X^1 & \boldsymbol{U}_X^2 \end{pmatrix} \begin{pmatrix} \boldsymbol{\Sigma}_X & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \boldsymbol{V}_X^1 \\ \boldsymbol{V}_X^2 \end{pmatrix} \tag{7.114}$$

For a matrix with full column rank $\boldsymbol{V}_X^2 = 0$ and we write $\boldsymbol{V}_X^1 = \boldsymbol{V}_X$

Algorithm 7.11 *Computation of* n_{sub}, $\boldsymbol{\Gamma}$, $\boldsymbol{\Phi}$
Initialisation: Let

$$\boldsymbol{Z} = \begin{pmatrix} -\boldsymbol{I}_n & \boldsymbol{O}_{n\times n_d} \\ \boldsymbol{O}_{p\times n} & \boldsymbol{O}_{p\times n_{\text{d}}} \end{pmatrix}, \boldsymbol{W} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{F}_{\text{d}} \\ \boldsymbol{C} & \boldsymbol{E}_{\text{d}} \end{pmatrix}.$$

Set

$$\boldsymbol{Z}^* = \boldsymbol{Z}, \boldsymbol{W}^* = \boldsymbol{W}, \boldsymbol{M} = \boldsymbol{I}_{n+p} \text{ and } \boldsymbol{N} = \boldsymbol{I}_{n+n_{\text{d}}}.$$

Compute:

a. While $\boldsymbol{Z}^*$ is not full column rank, do

1. perform a singular value decomposition of $\boldsymbol{Z}^*$, and compress the columns of $\boldsymbol{Z}^*$ by computing the right-hand side of the first equality below:
$\begin{pmatrix} \boldsymbol{Z}_1^* & \boldsymbol{O} \end{pmatrix} = \boldsymbol{Z}^* \begin{pmatrix} \boldsymbol{V}_{Z^*}^{1\text{T}} & \boldsymbol{V}_{Z^*}^{2\text{T}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{U}_{Z^*}^1 \boldsymbol{\Sigma}_{Z^*} & \boldsymbol{O} \end{pmatrix}$.
2. Let $\begin{pmatrix} \boldsymbol{W}_1^* & \boldsymbol{W}_2^* \end{pmatrix} = \boldsymbol{W}^* \begin{pmatrix} \boldsymbol{V}_{Z^*}^{1\text{T}} & \boldsymbol{V}_{Z^*}^{2\text{T}} \end{pmatrix}$.
3. Find the highest rank full row rank matrix $\boldsymbol{L}$ satisfying $\boldsymbol{L}\boldsymbol{W}_2^* = 0$ by using the singular value decomposition of $\boldsymbol{W}_2^*$, namely $\boldsymbol{L} = \boldsymbol{U}_{W_2^*}^{2\text{T}}$.

4. Let $\boldsymbol{Z}^* = \boldsymbol{L}\boldsymbol{Z}_1^*$, $\boldsymbol{W}^* = \boldsymbol{L}\boldsymbol{W}_1^*$, $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{M}$, $\boldsymbol{N} = \boldsymbol{N}\boldsymbol{V}_{Z^*}^{1^{\mathrm{T}}}$, end do.

b. Determine an invertible matrix $\boldsymbol{T}$ such that

$$\boldsymbol{T}\boldsymbol{Z}^* = \begin{pmatrix} -\boldsymbol{I} \\ \boldsymbol{O} \end{pmatrix},$$

where the dimension of $\boldsymbol{I}$ is obviously equal to $\operatorname{rank}\boldsymbol{Z}^*$. Such a matrix can be computed as follows:

$$\boldsymbol{T} = \begin{pmatrix} (\boldsymbol{\Sigma}_{Z^*}\boldsymbol{V}_{Z^*})^{-1} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{U}_{Z^*}^{1^{\mathrm{T}}} \\ \boldsymbol{U}_{Z^*}^{2^{\mathrm{T}}} \end{pmatrix},$$

c. Set $\boldsymbol{\Gamma} = \boldsymbol{T} - \boldsymbol{M}$, $\boldsymbol{\Phi} = \boldsymbol{N}$,

$$\begin{pmatrix} \boldsymbol{A}_{\mathrm{sub}} \\ \boldsymbol{C}_{\mathrm{sub}} \end{pmatrix} = -\boldsymbol{\Gamma}\boldsymbol{W}\boldsymbol{\Phi}, \quad \begin{pmatrix} \boldsymbol{B}_{\mathrm{sub}} \\ \boldsymbol{D}_{\mathrm{sub}} \end{pmatrix} = \boldsymbol{\Gamma} \begin{pmatrix} \boldsymbol{B}_{\epsilon} \\ \boldsymbol{D}_{\epsilon} \end{pmatrix}.$$

The above design procedure may fail in different ways:

- When the dimensions of $\boldsymbol{\Gamma}$ and $\boldsymbol{\Phi}$ are such that $\begin{pmatrix} \boldsymbol{A}_{\mathrm{sub}} \\ \boldsymbol{C}_{\mathrm{sub}} \end{pmatrix}$ is a square matrix, the obtained subsystem has no output, and hence no Kalman filter can be designed and no residual generator can be obtained. This typically occurs when $n_{\mathrm{d}} \geq p$.
- When

$$\begin{pmatrix} z\boldsymbol{I} + \boldsymbol{A}_{\mathrm{sub}} & \boldsymbol{B}_{\mathrm{sub}} \\ \boldsymbol{C}_{\mathrm{sub}} & \boldsymbol{D}_{\mathrm{sub}} \end{pmatrix}$$

has full generic rank, but it loses rank for $z = \exp(-\mathrm{j}\omega)$, $\omega \in |\mathcal{R}$, then it is not possible to design a residual generator as the regularity assumption needed for the design of the Kalman filter is not fulfilled.[1]
- When the regularity assumption ceases to be met due to $\boldsymbol{B}_{\mathrm{sub}} = \boldsymbol{O}$, $\boldsymbol{D}_{\mathrm{sub}} = \boldsymbol{O}$ or due to

$$\begin{pmatrix} -z\boldsymbol{I} + \boldsymbol{A}_{\mathrm{sub}} & \boldsymbol{B}_{\mathrm{sub}} \\ \boldsymbol{C}_{\mathrm{sub}} & \boldsymbol{D}_{\mathrm{sub}} \end{pmatrix}$$

having not full generic rank, the design is more involved and the reader is referred to the bibliography for this case.

Example 7.4 Innovation filter design for the ship example
Let us consider the linearised augmented ship-steering model described by combining the wave model and the ship-steering system

[1] Fulfilment of this condition can be checked by computing the zeros of system $(\boldsymbol{A}_{\mathrm{sub}}, \boldsymbol{B}_{\mathrm{sub}}, \boldsymbol{C}_{\mathrm{sub}}, \boldsymbol{D}_{\mathrm{sub}})$ and by verifying that none of them lies on the unit circle.

$$\begin{pmatrix} \dot{x}_{w1} \\ \dot{x}_{w2} \\ \dot{\omega}_3 \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} -2\eta_\omega \sigma_0 & -\sigma_0^2 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & b\eta_1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_{w1} \\ x_{w2} \\ \omega_3 \\ \psi \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ b \\ 0 \end{pmatrix} \delta + \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} w_\omega$$

$$\begin{pmatrix} \omega_{3m} \\ \psi_m \end{pmatrix} = \begin{pmatrix} 1\ 0\ 1\ 0 \\ 0\ 0\ 0\ 1 \end{pmatrix} \begin{pmatrix} x_{w1} \\ x_{w2} \\ \omega_3 \\ \psi \end{pmatrix} + \begin{pmatrix} f_\omega \\ f_\psi \end{pmatrix} + \begin{pmatrix} \nu_\omega \\ \nu_\psi \end{pmatrix}.$$

A sampled-data model of this system has been obtained at a sampling rate of 0.5 Hz. The resulting equations are:

$$\begin{pmatrix} x_{w1}(k+1) \\ x_{w2}(k+1) \\ \omega_3(k+1) \\ \psi(k+1) \end{pmatrix} = \begin{pmatrix} -0.1281 & -0.6365 & 0 & 0 \\ 0.9945 & 0.1106 & 0 & 0 \\ 0 & 0 & 0.0000 & 0 \\ 0.9945 & -0.8894 & 0.0500 & 1.0000 \end{pmatrix} \begin{pmatrix} x_{w1}(k) \\ x_{w2}(k) \\ \omega_3(k) \\ \psi(k) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0.0500 \\ 0.0975 \end{pmatrix} \delta(k) + \boldsymbol{w}(k) \tag{7.115}$$

$$\begin{pmatrix} \omega_{3m}(k) \\ \psi_m(k) \end{pmatrix} = \begin{pmatrix} 1\ 0\ 1\ 0 \\ 0\ 0\ 0\ 1 \end{pmatrix} \begin{pmatrix} x_{w1}(k) \\ x_{w2}(k) \\ \omega_3(k) \\ \psi(k) \end{pmatrix} + \begin{pmatrix} f_\omega(k) \\ f_\psi(k) \end{pmatrix} + \begin{pmatrix} v_\omega(k) \\ v_\psi(k) \end{pmatrix}. \tag{7.116}$$

The covariance matrix of the state noise $\boldsymbol{w}(k)$ can be evaluated by the sampling procedure described in Appendix B. It yields

$$E(\boldsymbol{w}(k)\boldsymbol{w}(k)^{\mathrm{T}}) = \boldsymbol{Q}_w = \begin{pmatrix} 0.0015 & 0.0056 & 0.0019 & 0.0056 \\ 0.0056 & 0.0322 & 0.0077 & 0.0322 \\ 0.0019 & 0.0077 & 0.0024 & 0.0077 \\ 0.0056 & 0.0322 & 0.0077 & 0.0322 \end{pmatrix}.$$

The measurement noise sequence is characterised by a covariance matrix given as

$$\boldsymbol{Q}_v = \begin{pmatrix} 0.0001 & 0 \\ 0 & 0.005 \end{pmatrix}.$$

State and measurement noises are supposed to be uncorrelated, hence $\boldsymbol{Q}_{wv} = \boldsymbol{O}$.

The considered input signal $\delta(t)$ is a sine wave with period 20π s.

Figure 7.12 gives the evolution of the sampled output signals in healthy working mode (first 300 samples), when a 0.1 deg/s bias on the turn rate $\omega_3(k)$ is added (from sample 301 to sample 600), and when this bias disappears bringing the system back to healthy working

mode (sample 601 to 900) In other words, a step-like fault f_ω occurs between sample 301 and 600.

From the above model, the following Kalman filter is deduced.[2]

$$\begin{pmatrix} \hat{x}_{w1}(k+1) \\ \hat{x}_{w2}(k+1) \\ \hat{\omega}_3(k+1) \\ \hat{\psi}(k+1) \end{pmatrix} = \begin{pmatrix} -0.1281 & -0.6365 & 0 & 0 \\ 0.9945 & 0.1106 & 0 & 0 \\ 0 & 0 & 0.0000 & 0 \\ 0.9945 & -0.8894 & 0.0500 & 1.0000 \end{pmatrix} \begin{pmatrix} \hat{x}_{w1}(k) \\ \hat{x}_{w2}(k) \\ \hat{\omega}_3(k) \\ \hat{\psi}(k) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0.0500 \\ 0.0975 \end{pmatrix} \delta(k) + \begin{pmatrix} -0.3265 & -0.4544 \\ 0.6710 & 0.0067 \\ 0.0000 & 0.0000 \\ 0.6880 & 0.0119 \end{pmatrix} \cdot \left(\begin{pmatrix} \omega_{3m}(k) \\ \psi_m(k) \end{pmatrix} - \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{x}_{w1}(k) \\ \hat{x}_{w2}(k) \\ \hat{\omega}_3(k) \\ \hat{\psi}(k) \end{pmatrix} \right). \tag{7.117}$$

The innovation is computed from

$$r(k) = \begin{pmatrix} \omega_{3m}(k) \\ \psi_m(k) \end{pmatrix} - \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{x}_{w1}(k) \\ \hat{x}_{w2}(k) \\ \hat{\omega}_3(k) \\ \hat{\psi}(k) \end{pmatrix}. \tag{7.118}$$

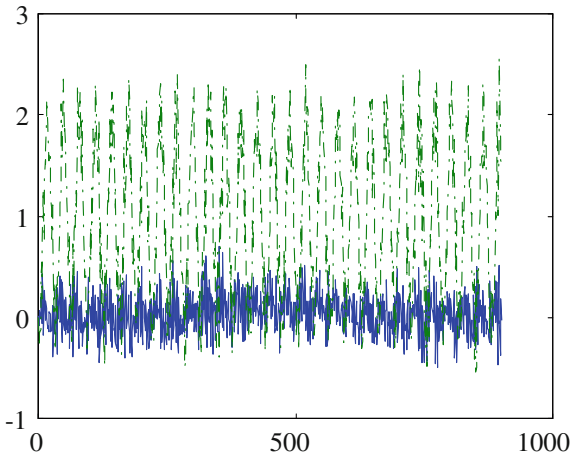


Fig. 7.12 Sampled output sequence of ship model in healthy and faulty working modes; ω_{3m} as a function of sample number (*continuous line*), ψ_m as a function of sample number (*dash-dotted line*)

[2]The gain of this filter can be computed by MATLAB function *dlqe* for instance.

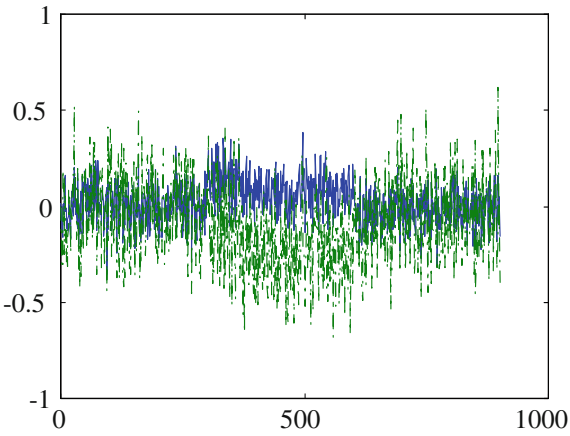


Fig. 7.13 Innovation sequences computed by (7.117), (7.118) from the data of Fig. 7.12; first component (*continuous line*); second component (*dash-dotted line*)

The innovation sequences for the data of Fig. 7.12 is plotted in Fig. 7.13. The change in the mean of the innovation sequence due to the fault is visible. However, such a change cannot be detected by comparing the signals to a simple threshold. □

The existence of a filter that meets the conditions in Problem 7.8 does not guarantee that the filter output (namely, the innovation) is useful for fault detection. It should be affected by $\boldsymbol{f}$ in order to meet the second condition of Problem 7.7, and thus to be suitable as a residual. This issue is addressed in the next subsection.

Sensitivity to faults or fault detectability. There are several ways to define the sensitivity of an innovation signal to a fault $\boldsymbol{f}$ or, equivalently, the detectability of a fault by a given innovation signal.

In a similar way as for the deterministic case, an innovation filter for system (7.89) is said to be *fault sensitive* if its output is affected by $\boldsymbol{f}$. Equivalently, the fault is said to be detectable in this case.

If the transfer function from $\boldsymbol{f}$ to $\boldsymbol{r}$ is left invertible, then the innovation filter is *strictly fault sensitive*.

It can be shown that system (7.89) has a (strict) fault sensitive innovation filter which solves Problem 7.8 if and only if every innovation filter solution of Problem 7.8 is (strictly) fault sensitive. Thus (strict) fault sensitivity is a property of the supervised system (7.89), and it does not depend upon the choice of innovation filter. Therefore, (strict) fault sensitivity of (7.89) will be referred to in the sequel.

Assuming the pair $(\boldsymbol{C}_{\text{sub}}, \boldsymbol{A}_{\text{sub}})$ resulting from the design procedure is observable, the following necessary and sufficient conditions for sensitivity can be exploited:

System (7.89) is fault sensitive if and only if [3]:

[3]The image (space) $\Im(\boldsymbol{X})$ of a linear transformation associated to the $n \times m$ matrix $\boldsymbol{X}$ is the set of all vectors $\boldsymbol{y}$ in $|\mathcal{R}^n$ that equal $\boldsymbol{Xu}$ for some $\boldsymbol{u}$ in $|\mathcal{R}^m$. The kernel (or null space) $Ker(\boldsymbol{X})$ of a linear transformation associated to the $n \times m$ matrix $\boldsymbol{X}$ is the set of all vectors $\boldsymbol{u}$ in $|\mathcal{R}^m$ that fulfil $\boldsymbol{Xu} = 0$.

$$\Im \begin{pmatrix} \boldsymbol{F}_f \\ \boldsymbol{E}_f \end{pmatrix} \not\subset Ker(\boldsymbol{\Gamma}). \tag{7.119}$$

System (7.89) is strictly fault sensitive if and only if system ($\boldsymbol{A}_{\text{sub}}$, $\boldsymbol{F}_{f,sub}$, $\boldsymbol{C}_{\text{sub}}$, $\boldsymbol{E}_{f,sub}$), where

$$\begin{pmatrix} \boldsymbol{F}_{f,sub} \\ \boldsymbol{E}_{f,sub} \end{pmatrix} = \boldsymbol{\Gamma} \begin{pmatrix} \boldsymbol{F}_f \\ \boldsymbol{E}_f \end{pmatrix} \tag{7.120}$$

is left invertible.

Yet another notion is strong fault sensitivity, which is typically considered for scalar faults. As for a deterministic residual, the innovation signal is strongly fault sensitive when it reaches a non-zero steady state value for a step-like fault, $f(k) = \bar{f} 1_{\{k>k_0\}}$, for any constant non-zero $\bar{f}$. This property can be checked a posteriori by computing the steady state gain of the transfer function between fault f and innovation $\boldsymbol{r}$ and verifying that it has at least one non-zero entry.

Remark 7.11 (*Comments on strong fault detectabilty*) In a deterministic framework, necessary and sufficient conditions for the existence of a residual generator which is strongly fault sensitive for a given system have been developed [255]. The corresponding fault is said to be strongly detectable. It is unclear whether the innovation signal computed as the output of the filter (7.92) (or as the innovation of a Kalman filter for the subsystem in step 2 of the Algorithm 7.10) is necessarily strongly fault sensitive, when a strongly detectable fault is considered. □

Distribution of the residual vector and residual evaluation. For proper choice of the residual evaluation method, it is necessary to analyse the statistical distribution of $\boldsymbol{r}(k)$. For the sake of simplicity, the situation, where $\boldsymbol{x}_0$, $\boldsymbol{v}(k)$, $\boldsymbol{w}(k)$, $k = 0, 1, \ldots$, are normally distributed is considered. Then, the residual has asymptotically (as k tends to infinity) a Gaussian distribution with known variance and with zero mean or non-zero mean, depending on whether $\boldsymbol{f}(k)$ asymptotically vanishes or not assuming the fault is strongly detectable. The normal distribution results from the linearity of the filter and the supervised process.

In order to characterise this distribution, let us consider the situation, where there is no unknown input, and hence the residual generator is given by (7.92). The reasoning below also applies to the Kalman filter designed for the system given in step 2 of the Algorithm 7.10, but the notations are more cumbersome. The first two moments of the distribution of $\boldsymbol{r}(k)$ can be computed as follows. Let $\tilde{\boldsymbol{x}}(k) = \boldsymbol{x}(k) - \hat{\boldsymbol{x}}(k)$. Then classical results on steady state Kalman filters provide the following expression for the mean and the variance of $\tilde{\boldsymbol{x}}(k)$ in the absence of fault:

$$\lim_{k\to\infty} E(\tilde{\boldsymbol{x}}(k)) = \boldsymbol{0}$$

$$\lim_{k\to\infty} E\left(\tilde{\boldsymbol{x}}(k)\,\tilde{\boldsymbol{x}}(k)^{\mathrm{T}}\right) = \boldsymbol{P},$$

with $\boldsymbol{P}$ given as the semi-positive definite solution of (7.94). By substituting the second equation of (7.89) (with $\boldsymbol{E}_{\mathrm{d}} = \boldsymbol{O}$) for $\boldsymbol{y}(k)$ in the expression (7.92) for $\boldsymbol{r}(k)$, the residual can be written as

$$\boldsymbol{r}(k) = \boldsymbol{C}\tilde{\boldsymbol{x}}(k) + \boldsymbol{E}_f\, \boldsymbol{f}(k) + \boldsymbol{D}_\epsilon\, \boldsymbol{\epsilon}(k). \tag{7.121}$$

When $\boldsymbol{f}(k)$ vanishes as k tends to infinity, one deduces from (7.121) with $\boldsymbol{f}(k) = 0$:

$$\begin{aligned} \boldsymbol{r}_m &= \lim_{k\to\infty} E\left(\boldsymbol{r}(k)\right) = \boldsymbol{0} \\ \boldsymbol{Q}_r &= \lim_{k\to\infty} E\left(\left(\boldsymbol{r}(k) - \boldsymbol{r}_m\right)\left(\boldsymbol{r}(k) - \boldsymbol{r}_m\right)^{\mathrm{T}}\right) = \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^{\mathrm{T}} + \boldsymbol{D}_\epsilon\, \boldsymbol{D}_\epsilon^{\mathrm{T}}. \end{aligned}$$

If, on the contrary $\lim_{k\to\infty} \boldsymbol{f}(k) = \bar{\boldsymbol{f}} \neq 0$, the residual mean is non-zero. It can be obtained from the transfer function between $\boldsymbol{f}(z)$ and $\boldsymbol{r}(z)$ deduced from (7.89) and (7.95), namely, $\boldsymbol{V}_{\mathrm{ry}}(z)\left(\boldsymbol{C}\left(z\boldsymbol{I} - \boldsymbol{A}\right)^{-1}\boldsymbol{F}_f + \boldsymbol{E}_f\right)$. Indeed, the mean of the residual is nothing but the steady state value of the residual for $\boldsymbol{f}(k) = \bar{\boldsymbol{f}}$. Thus,

$$\boldsymbol{r}_m = \lim_{k\to\infty} E\left(\boldsymbol{r}(k)\right) = \boldsymbol{V}_{\mathrm{ry}}(1)\left(\boldsymbol{C}(\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{F}_f + \boldsymbol{E}_f\right)\bar{\boldsymbol{f}}.$$

Stability of the supervised system is implicitly assumed when writing this expression. The variance of the residual is unchanged, since the fault signal is considered as deterministic.

The problem of fault detection thus amounts to deciding between the following two hypotheses:

$$\mathcal{H}_0 : \mathcal{L}\left(\boldsymbol{r}(k)\right) = As\,\mathcal{N}(0,\ \boldsymbol{Q}_r) \tag{7.122}$$

$$\mathcal{H}_1 : \mathcal{L}\left(\boldsymbol{r}(k)\right) = As\,\mathcal{N}\left(\boldsymbol{V}_{\mathrm{ry}}(1)\left(\boldsymbol{C}\left(\boldsymbol{I} - \boldsymbol{A}\right)^{-1}\boldsymbol{F}_f + \boldsymbol{E}_f\right)\bar{\boldsymbol{f}},\ \boldsymbol{Q}_r\right), \tag{7.123}$$

where the notation $\mathcal{L}\left(\boldsymbol{r}(k)\right)$ denotes the distribution of $\boldsymbol{r}(k)$, and

$$\mathcal{L}\left(\boldsymbol{r}(k)\right) = As\,\mathcal{N}(\boldsymbol{a},\ \boldsymbol{X})$$

indicates that this distribution is normal with mean $\boldsymbol{a}$ and variance $\boldsymbol{X}$ as k tends to infinity. Note that the residual must be strongly sensitive to fault $\boldsymbol{f}$ for the distributions under $\mathcal{H}_0$ and $\mathcal{H}_1$ to be different.

The asymptotic character of (7.122), (7.123) is due to the effects of initial conditions and filter transient upon occurrence of a fault. Neglecting this transient, and assuming that $\bar{\boldsymbol{f}}$ is known, one can recast the above problem as the following test of hypotheses:

Problem 7.9 (*Test of hypotheses: transient not accounted for*) Given a sequence of residual vectors $\boldsymbol{r}(1), \ldots, \boldsymbol{r}(k)$, obtained as the output of filter (7.92), choose between the following two hypotheses at the current time instant k:

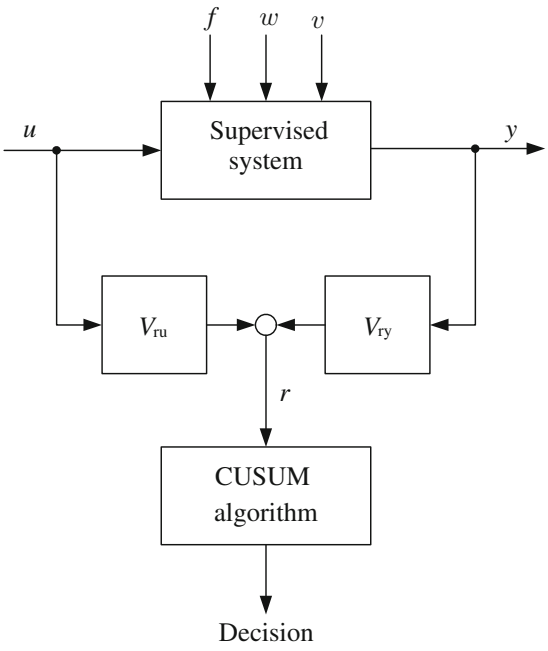


Fig. 7.14 Fault detection system

$\mathcal{H}_0$: $\mathcal{L}(\boldsymbol{r}(i)) = \mathcal{N}(0,\ \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^{\mathrm{T}} + \boldsymbol{D}_\epsilon \boldsymbol{D}_\epsilon^{\mathrm{T}})$ for $1 \le i \le k$,
$\mathcal{H}_1$: From time instant 1 up to an unknown time instant k_0, $\boldsymbol{r}(i)$, $i = 1, \ldots, k_0 - 1$ is distributed as

$$\mathcal{L}(\boldsymbol{r}(i)) = \mathcal{N}\left(0,\ \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^{\mathrm{T}} + \boldsymbol{D}_\epsilon \boldsymbol{D}_\epsilon^{\mathrm{T}}\right)$$

while for time instant $i \ge k_0$

$$\mathcal{L}(\boldsymbol{r}(i)) = \mathcal{N}\left((\boldsymbol{V}_{\mathrm{ry}}(1)\,\boldsymbol{C}\,(\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{F}_f + \boldsymbol{E}_f)\bar{\boldsymbol{f}},\ \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^{\mathrm{T}} + \boldsymbol{D}_\epsilon \boldsymbol{D}_\epsilon^{\mathrm{T}}\right).$$

This problem is of the form of a change detection in the mean of a Gaussian vector sequence (Problem 7.2) with $\boldsymbol{z}(i)$ replaced by $\boldsymbol{r}(i)$, $\boldsymbol{\mu}_0 = 0$, $\boldsymbol{Q} = \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^{\mathrm{T}} + \boldsymbol{D}_\epsilon \boldsymbol{D}_\epsilon^{\mathrm{T}}$ and $\boldsymbol{\mu}_1 = \left(\boldsymbol{V}_{\mathrm{ry}}(1)\,\boldsymbol{C}(\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{F}_f + \boldsymbol{E}_f\right)\bar{\boldsymbol{f}}$. Hence, the CUSUM algorithm based on a step-like change can be used for residual evaluation, with $\bar{f}$ taken as twice the minimum magnitude of the fault to be detected or as the most likely magnitude of this fault. The complete fault detection system is depicted in Fig. 7.14.

Remark 7.12 (χ^2 *-test*) In some applications, particularly in the area of predictive maintenance, the delay for detection may not be a crucial factor, and one may resort to an offline approach to solve a simplified version of the above hypotheses testing problem. The most recent data over a sliding window $[k - M + 1,\ k]$ are considered, and the time instant k_0 is set to 1, which amounts to considering that the change

has affected all elements of the batch of data. The method to solve this hypotheses testing problem relies on a χ^2-test. □

When stating the above hypotheses testing problem, the transient of the system and the residual generator upon occurrence of a fault are not taken into account. This may significantly affect the detection delay. If a priori knowledge on the fault sequence $\boldsymbol{f}(i),\ i = k_0, k_0+1, \ldots$ is available, the performance of the detection system can be improved by introducing a suitable dynamical profile of change in the CUSUM algorithm.

Most commonly, step-like changes in the fault sequence are considered, namely,

$$\begin{aligned} \boldsymbol{f}(i) &= \boldsymbol{0} \quad i = 1, 2, \ldots, k_0 - 1 \\ \boldsymbol{f}(i) &= \bar{\boldsymbol{f}} \quad i \geq k_0 \end{aligned} \tag{7.124}$$

or, in a compact way, $\boldsymbol{f}(i) = \bar{\boldsymbol{f}} 1_{\{i \geq k_0\}}$, where $\bar{\boldsymbol{f}}$ is a constant vector.

Due to the linearity of the system (7.89) and the filter (7.92), the residual sequence can be written as

$$\boldsymbol{r}(k) = \boldsymbol{r}_0(k) + \boldsymbol{\rho}(k,\ k_0), \tag{7.125}$$

where $\boldsymbol{r}_0(k)$ is the value of the residual in the absence of fault, and $\boldsymbol{\rho}(k,\ k_0)$ is the contribution to $\boldsymbol{r}(k)$ of a fault occurring at time $k_0 \leq k$. In the case of a step-like fault considered above, $\boldsymbol{\rho}(k,\ k_0)$ can be computed easily; it only depends on the difference $k - k_0$, and hence, is written with an abuse of notation $\boldsymbol{\rho}(k,\ k_0) = \boldsymbol{\rho}(k - k_0)$. For the sake of simplicity, only a scalar fault sequence is considered. Then, $\boldsymbol{\rho}(k - k_0) = \tilde{\boldsymbol{\rho}}(k - k_0)\bar{f}$, where $\tilde{\boldsymbol{\rho}}(k)$ is the response of system (7.89), (7.92) to a fault signal of the form (7.124) with $\bar{f} = 1$, for $\boldsymbol{u}(k) = 0,\ \boldsymbol{d}(k) = 0$ and $\boldsymbol{\epsilon}(k) = 0$ for all $k > 0$, and for zero initial conditions. It coincides with the step response of the system with transfer function $\boldsymbol{V}_{\mathrm{ry}}(z)\ \left(\boldsymbol{C}(z\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{E}_f + \boldsymbol{F}_f\right)$. The hypotheses testing problem when taking into account the dynamical profile of the change can be written as

Problem 7.10 (*Test of hypotheses: transient accounted for*) Given a sequence of residual vectors $\boldsymbol{r}(1), \ldots, \boldsymbol{r}(k)$, obtained as the output of filter (7.92), choose between the following two hypotheses at the current time instant k

$\mathcal{H}_0$: $\mathcal{L}(\boldsymbol{r}(i)) = \mathcal{N}(0,\ \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^{\mathrm{T}} + \boldsymbol{D}_\epsilon \boldsymbol{D}_\epsilon^{\mathrm{T}})$ for $1 \leq i \leq k$,
$\mathcal{H}_1$: From time instant 1 up to an unknown time instant k_0, $\boldsymbol{r}(i)$, $i = 1, \ldots, k_0 - 1$ is distributed as

$$\mathcal{L}(\boldsymbol{r}(i)) = \mathcal{N}(0,\ \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^{\mathrm{T}} + \boldsymbol{D}_\epsilon \boldsymbol{D}_\epsilon^{\mathrm{T}}) \tag{7.126}$$

while for time instant $i \geq k_0$,

$$\mathcal{L}(\boldsymbol{r}(i)) = \mathcal{N}(\tilde{\boldsymbol{\rho}}(i - k_0)\bar{f},\ \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^{\mathrm{T}} + \boldsymbol{D}_\epsilon \boldsymbol{D}_\epsilon^{\mathrm{T}}). \tag{7.127}$$

This problem is in the form of Problem 7.3. (7.126), (7.127) precisely have the form (7.56), (7.57) with $\boldsymbol{r}(i)$ replacing $z(i)$, $\boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^{\mathrm{T}}+\boldsymbol{D}_{\epsilon}\boldsymbol{D}_{\epsilon}^{\mathrm{T}}$ replacing $\boldsymbol{Q}$, $\tilde{\boldsymbol{\rho}}(i-k_0)\bar{f}$ replacing $\boldsymbol{\rho}(i-k_0)$ and $\boldsymbol{\mu}_0 = 0$. The CUSUM algorithm based on a known dynamical profile of change can thus be applied with $\boldsymbol{\rho}(k) = \tilde{\boldsymbol{\rho}}(k)\bar{f}$, where $\bar{f}$ is taken as twice the minimum magnitude of the change to be detected or as the most likely magnitude of this change.

Remark 7.13 (*Delay in dynamical profile*) In the statement of Problem 7.3, $\rho(j)$ is supposed to be different from zero for $j > 0$. This hypothesis is not verified when the transfer function $\boldsymbol{V}_{\mathrm{ry}}(z)\left(\boldsymbol{C}(z\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{E}_f + \boldsymbol{F}_f\right)$ has no direct feedthrough term. In this case, one should use $\rho(k) = \tilde{\rho}(k-\tau)\bar{f}$, where τ denotes the minimum delay in the n_r elements of the mentioned transfer function. □

Note that strong fault sensitivity is no more a required property of the residual in order to achieve fault detection, when the dynamical profile of the change is accounted for. Indeed, it suffices that the mean of the distributions (7.126), (7.127) be different for some time interval. Checking that the fault subsists by reinitialisation of the CUSUM algorithm is, however, impossible when the residual is not strongly fault sensitive.

Remark 7.14 (*Fault sequence*) The choice of a step-like fault sequence can be made without loss of generality. Indeed, other signal forms could possibly be represented as the step response of a linear system, and this linear model could be included in the state-space Eqs. (7.88). □

Example 7.4 (**cont.**) **Ship example**
The CUSUM algorithm based on the knowledge of the dynamical profile of change will be used to detect the occurrence of fault f_ω. In order to determine the dynamical profile of the change to be used in the algorithm, it suffices to consider the response of the system made of Eqs. (7.115)–(7.118) to a step-like fault f_ω, keeping all other input signals equal to zero and starting with zero initial conditions. This corresponds to the step response of the transfer function $\boldsymbol{V}_{\mathrm{ry}}(z)\left(\boldsymbol{C}(z\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{E}_f + \boldsymbol{F}_f\right)$ with respect to the first input.

Given the specifications, one decides that the smallest bias on ω_3 to be detected is 0.025 deg/s. $\bar{f}_\omega$ is set to twice this value, which yields 0.05 deg/s. A step of magnitude 0.05 deg/s is thus applied as signal for f_ω. The vector dynamical profile of change with respect to fault f_ω, $0.05\,\tilde{\boldsymbol{\rho}}_\omega$ is plotted in Fig. 7.15.

The evolution of the CUSUM decision functions for detection of f_ω, g_{ω_3} is plotted in Fig. 7.16. The indicated threshold (dashed line) has been set on the basis of the value of the decision function for the first 300 samples (healthy working mode). The reinitialisation policy is the reset procedure indicated in the description of the algorithm. One notices the repeated threshold crossing of the decision function g_ω while the fault is present (from sample 300 to 600). □

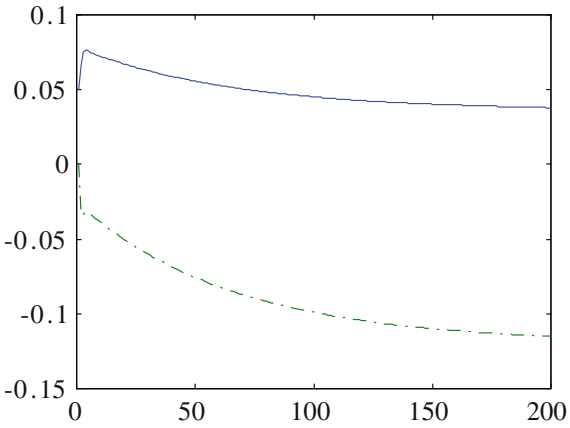


Fig. 7.15 Dynamic profile of change for fault f_ω; first component of $0.05\,\tilde{\rho}_\omega$ (*continuous line*); second component (*dash-dotted line*)

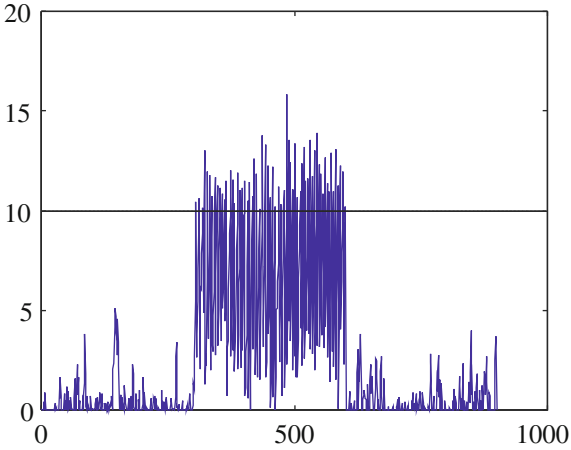


Fig. 7.16 CUSUM decision function resulting from application to the innovation sequence of Fig. 7.13 of the CUSUM algorithm based on the known dynamical profile of change (Fig. 7.15)

7.3.3 Fault Estimation

In this section, a model of the form (7.88) with $n_f = 1$ is considered. Besides, it is assumed that step-like faults of unknown magnitude occur. Thus a scalar sequence $f(i),\ i = 1, 2, \ldots$ of the form (7.124) with an unknown constant $\bar{f}$ is assumed. The problem can be stated as follows:

Problem 7.11 (*Fault estimation*) Given

1. a model of the process of the form (7.88) subject to a scalar step-like fault sequence $f(i) = \bar{f}\,1_{\{i \geq k_0\}}$ of unknown magnitude $\bar{f}$.
2. a sequence of measured process input and output values: $(\boldsymbol{y}(i),\ \boldsymbol{u}(i))_{1 \leq i \leq k}$, where k denotes the current time instant.

Choose between the following two hypotheses:

$\mathcal{H}_0$: healthy operating condition,
$\mathcal{H}_1$: faulty operating condition.

When $\mathcal{H}_1$ is selected, an estimate of the fault occurrence time, $\hat{k}_0$, and of the fault magnitude, $\hat{\bar{f}}$, should be provided.

As for the fault detection problem, a two-step procedure is used to solve this problem. The first step, namely the residual generation, is the same for both problems. For residual evaluation, a generalised likelihood ratio algorithm is used to obtain an estimate of the fault magnitude. Indeed, given the specific fault model, the residual evaluation reduces to Problem 7.10 in which $\bar{f}$ is unknown. Hence, it is of the form of Problem 7.4. Equations (7.126), (7.127) precisely have the form of Eqs. (7.63), (7.64) with $\boldsymbol{r}(i)$ replacing $\boldsymbol{z}(i)$, $\boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^{\mathrm{T}} + \boldsymbol{D}_\epsilon \boldsymbol{D}_\epsilon^{\mathrm{T}}$ replacing $\boldsymbol{Q}$, $\bar{f}$ replacing ν and $\boldsymbol{\mu}_0 = 0$. The GLR algorithm based on a known dynamical profile of change but an unknown fault magnitude can thus directly be used to process the residual vector in order to obtain an online solution to Problem 7.11.

Example 7.4 (cont.) Ship example
Let us again consider the innovation sequence depicted in Fig. 7.13. Instead of using a CUSUM algorithm, we now perform a GLR algorithm on this sequence. A dynamical profile of change has to be provided. It can be computed as for the CUSUM algorithm and one gets a profile similar to Fig. 7.15 except that the minimum fault magnitude is not accounted for. Thus to obtain the dynamical profile $\tilde{\rho}_\omega$, the signal f_ω which is used is a step function with unit magnitude instead of the magnitude of 0.05 deg/s used previously.

M is chosen as 50. This allows one to determine a quite precise estimate of the fault magnitude in the reinitialisation procedure. Given the values of the decision function obtained for the first 300 data, which correspond to the set $\{z_0(1), \ldots, z_0(N_0)\}$, and given its values upon occurrence of the fault, the threshold h is set to 30. The evolution of the GLR decision function is plotted in Fig. 7.17. Each time the threshold is crossed, an alarm is generated, and the decision function remains equal to zero until enough data are available for estimating the fault magnitude in a reliable way. The recursive algorithm restarts at $\hat{k}_0 + M$.

Note that successive changes separated by less than M samples cannot be handled properly. For the considered data, an alarm is generated at time instants 308 and 606. The estimated change times are 300 and 601, while the actual changes occur at 301 and 601. All numbers should be multiplied by the sampling period to obtain time in seconds. The estimates of the change magnitude used for reinitilisation are, respectively, 0.1020 for the positive change and -0.1073 for the negative change (disappearance of the fault). Remember that the actual change magnitude is 0.1 in both cases. Note that the estimate of the change magnitude plotted in Fig. 7.18 converges relatively fast after occurrence of a fault. Hence the horizon M could possibly be chosen smaller for this situation, yet this value is used to make the convergence of the estimate visible in the plot.

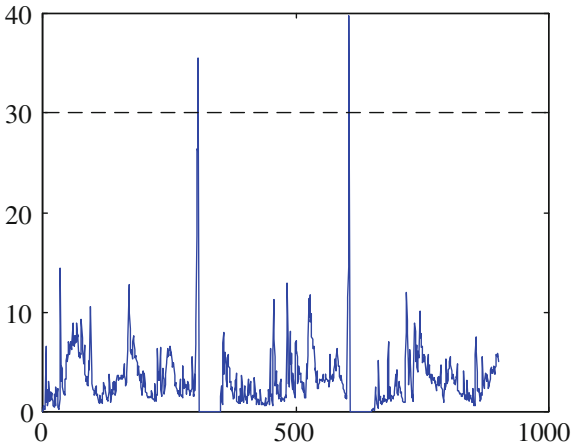


Fig. 7.17 GLR decision function resulting from application to the innovation sequence of Fig. 7.13 of the algorithm with known dynamical profile of change

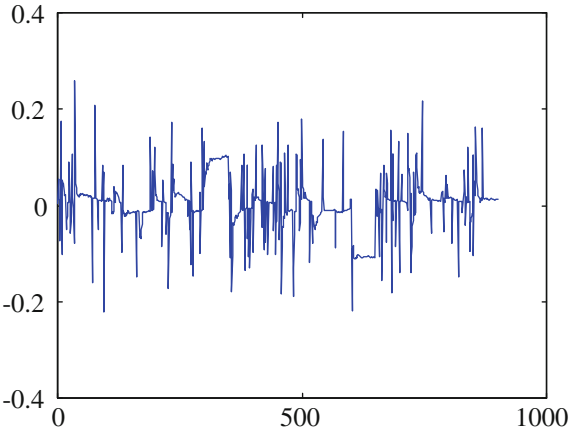


Fig. 7.18 Estimate of the change magnitude resulting from application to the innovation sequence of Fig. 7.13 of the GLR algorithm with known dynamical profile of change

7.3.4 Fault Isolation

Up to now the plant model used in the stochastic framework only contained one single (possibly vector) fault to be detected. However, most often several faults may affect the behaviour of the supervised process, and one should not only detect them, but also isolate the faulty components. An appropriate model to describe the process then takes the form

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) + \sum_{i=1}^{n_f} \boldsymbol{F}_i f_i(k) + \boldsymbol{B}_\epsilon \boldsymbol{\epsilon}(k) \\ \boldsymbol{x}(0) &= \boldsymbol{x}_0 \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) + \sum_{i=1}^{n_f} \boldsymbol{E}_i f_i(k) + \boldsymbol{D}_\epsilon \boldsymbol{\epsilon}(k), \end{aligned} \tag{7.128}$$

where, for the sake of simplicity, scalar faults f_i, $i = 1, \ldots, n_f$ are considered. Besides, in the residual evaluation stage, it will be assumed that f_i asymptotically tends to a constant value, say $\bar{f}_i, i = 1, \ldots, n_f$.

Two approaches can be distinguished to handle this situation. The first one is similar to the method presented in Sect. 6.4. It aims at ensuring fault isolation by enforcing a specific structure in the mapping from faults to residuals. The second one resorts to a multi-CUSUM approach to distinguish between different directions in the residual space, while using a single innovation filter for residual generation. Both methods are presented below and their respective advantages and drawbacks are discussed.

Method based on structured residuals. A set of residual generators are designed in such a way that each of them provides an output that is only affected by certain faults. This can be achieved by recasting the problem as a fault detection problem for a specific system with unknown input. An ideal situation is reached when the structure depicted in Table 7.2 can be enforced, assuming a situation with three faults.

An $\times$ in Table 7.2 indicates that the fault in the corresponding column affects the residual in the corresponding row. In this case, the fault detection problem corresponding to row α is based on model (7.88) with

$$\boldsymbol{d} = \left(f_1, \ldots, f_{\alpha-1}, \ f_{\alpha+1}, \ldots, f_{n_f}\right)^{\mathrm{T}}$$

and $f = f_\alpha$. n_f such problems must be solved to obtained the n_f residual generators.

From the conditions for fault detectability indicated in Sect. 6.4, the following necessary conditions can be deduced for the above scheme to work

$$\begin{aligned} \text{rank } \left(\boldsymbol{H}_{\mathrm{y},f_\ell}(s) \ \ \boldsymbol{H}_{\mathrm{y},f_j}(s)\right) > \text{rank } \boldsymbol{H}_{\mathrm{y},f_j}(s) \\ \text{for all } \ \ell, \ j = 1, \ldots, n_f, \ell \neq j, \end{aligned} \tag{7.129}$$

Table 7.2 Effects of the faults on the residuals

	f_1	f_2	f_3
r_1	$\times$	0	0
r_2	0	$\times$	0
r_3	0	0	$\times$

where $\boldsymbol{H}_{y,f_\ell}(z)$ is the transfer matrix between f_ℓ and y.[4] Roughly speaking, this indicates that two different faults have not the same effect on the measured output vector.

When it is not possible to design residual generators in such a way that each residual is only sensitive to a single fault, it is still possible to achieve fault isolation provided the zero entries in the table characterising the effect of the faults on the residual have a different pattern in each column. However, only a diagonal structure such as in Table 7.2 allows isolation of multiple simultaneous faults. An issue to keep in mind, however, is the effect of the modelling uncertainties on the decoupling of the residual vectors $\boldsymbol{r}_i$ with respect to the faults to which they should not be sensitive.

For processing the global residual vector generated according to Table 7.2, the first idea would be to handle each $\boldsymbol{r}_i$, $i = 1, \ldots, n_f$ individually by a GLR algorithm or a CUSUM algorithm according as an estimate of the fault magnitude is needed or not. However, this approach does not account for the correlation between the different vector residuals. A better solution, when no estimate of fault magnitude is needed is to resort to a multi-CUSUM algorithm (see Sect. 7.2.6) that processes at once the vector $\boldsymbol{r}(k) = \left(\boldsymbol{r}_1(k)^{\mathrm{T}}\ \boldsymbol{r}_2(k)^{\mathrm{T}}\ \ldots\ \boldsymbol{r}_{n_f}(k)^{\mathrm{T}}\right)^{\mathrm{T}}$. This algorithm is also the easiest alternative when a non-diagonal structure is used for residual generation. Its application is illustrated in the next section where the residual is generated from a single innovation filter exploiting all measurements at once.

Method based on a single innovation filter. The residual issued by a single innovation filter is determined from (7.92) which is repeated here for the sake of readability

$$\begin{array}{ll} \hat{\boldsymbol{x}}(k+1) & = \boldsymbol{A}\hat{\boldsymbol{x}}(k) + \boldsymbol{B}\boldsymbol{u}(k) - \boldsymbol{K}(\boldsymbol{y}(k) - \boldsymbol{C}\hat{\boldsymbol{x}}(k) - \boldsymbol{D}\boldsymbol{u}(k)), \\ \hat{\boldsymbol{x}}(0) & = \hat{\boldsymbol{x}}_0 \\ \boldsymbol{r}(k) & = \boldsymbol{y}(k) - \boldsymbol{C}\hat{\boldsymbol{x}}(k) - \boldsymbol{D}\boldsymbol{u}(k), \end{array} \tag{7.130}$$

with filter gain computed from (7.93). The concatenated system made of model (7.128) and (7.130) can be seen as a system with inputs $\boldsymbol{u}$, f_ℓ, $\ell = 1, \ldots, n_f$, $\boldsymbol{\epsilon}$ and output $\boldsymbol{r}$. Following the same reasoning that led to (7.122), (7.123), the distribution of the residual under fault free and faulty situation can be directly deduced, namely,

$$\begin{aligned} \mathcal{L}(\boldsymbol{r}(k)) &= As\,\mathcal{N}(0,\ \boldsymbol{Q}_r) \quad \text{in the absence of fault} \\ \mathcal{L}(\boldsymbol{r}(k)) &= As\,\mathcal{N}\left(\left[\boldsymbol{V}_{\mathrm{ry}}(1)\left(\boldsymbol{C}(\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{F}_f + \boldsymbol{E}_f\right)\right]_{.,\ell}\bar{\boldsymbol{f}}_\ell,\ \boldsymbol{Q}_r\right) \\ &\qquad \text{in the presence of fault } \ell,\ \ell = 1, \ldots, n_f \end{aligned}$$

where $[.]_{.,\ell}$ denotes the ℓth column of a matrix and $\boldsymbol{Q}_r = \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}^{\mathrm{T}} + \boldsymbol{D}_\epsilon\boldsymbol{D}_\epsilon^{\mathrm{T}}$.

The problem of fault detection and isolation can thus be formulated in the form of Problem 7.5 with $\boldsymbol{Q} = \boldsymbol{Q}_r$ and $\boldsymbol{\mu}_\ell = \left[\boldsymbol{V}_{\mathrm{ry}}(1)\left(\boldsymbol{C}(\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{F}_f + \boldsymbol{E}_f\right)\right]_{.,\ell}\bar{\boldsymbol{f}}_\ell$. It can be solved by the multi-CUSUM algorithm presented in Sect. 7.2.6. Notice that, upon occurrence of a fault, the residual signal will exhibit a transient which is not

[4] rank $\boldsymbol{H}(z)$ stands for the normal rank of matrix $\boldsymbol{H}(z)$; it can be computed as $\max_z$ rank $\boldsymbol{H}(z)$.

accounted for in the present approach. If a specific fault pattern is assumed, say a step-like fault for instance, the corresponding profile in the residual sequence can be accounted for in the change detection/isolation algorithm, as mentioned in the reference section.

Example 7.4 (cont.) Ship example
Faults on both the rate sensor and the angular position sensor are now considered. Figure 7.19 depicts the measurement signals obtained when step-like faults with magnitude 0.1 deg/s and 0.5 deg are, respectively, introduced on ω_{3m} between time instant 301 and 600 and on ψ_m between time instant 900 and 1200. All time data are expressed in number of sampling periods.

In order to achieve fault isolation, two residual signals are generated, one being sensitive to f_ω, the other to f_ψ. To this end, consider the sampled-data ship model (7.115), (7.116). If a Kalman filter is designed for this system using only the first measurement equation in (7.116), the resulting residual will only be affected by f_ω. Such a filter cannot be designed because the resulting system is not detectable. However, there is no need to estimate the whole state to generate a residual; it suffices to design a Kalman filter for the first 3 state equations in (7.115) and the first measurement equation. This filter takes the form:

$$\begin{pmatrix} \hat{x}_{w1}(k+1) \\ \hat{x}_{w2}(k+1) \\ \hat{\omega}_3(k+1) \end{pmatrix} = \begin{pmatrix} -0.1281 & -0.6365 & 0 \\ 0.9945 & 0.1106 & 0 \\ 0 & 0 & 0.0000 \end{pmatrix} \begin{pmatrix} \hat{x}_{w1}(k) \\ \hat{x}_{w2}(k) \\ \hat{\omega}_3(k) \end{pmatrix}$$
$$+ \begin{pmatrix} 0 \\ 0 \\ 0.0500 \end{pmatrix} \delta(k) + \begin{pmatrix} -0.6760 \\ 0.8104 \\ 0.0000 \end{pmatrix} \cdot \left(\omega_{3m}(k) - (1 \;\; 0 \;\; 1) \begin{pmatrix} \hat{x}_{w1}(k) \\ \hat{x}_{w2}(k) \\ \hat{\omega}_3(k) \end{pmatrix} \right) \tag{7.131}$$

The innovation is computed from

$$r_{\omega_3}(k) = \omega_{3m}(k) - (1 \;\; 0 \;\; 1) \begin{pmatrix} \hat{x}_{w1}(k) \\ \hat{x}_{w2}(k) \\ \hat{\omega}_3(k) \end{pmatrix} \tag{7.132}$$

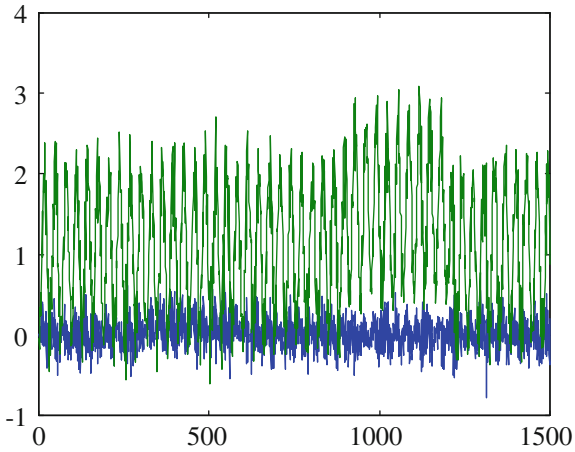


Fig. 7.19 Angular rate and heading measurements

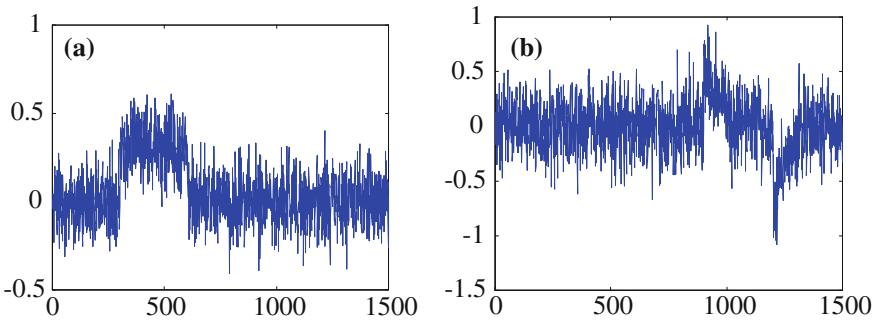


Fig. 7.20 Residual affected by f_ω (**a**) or f_ψ (**b**) only

It is plotted in Fig. 7.20a. A significant change in the mean of this signal is visible when the fault on ω_3 is present.

The design of a residual generator for detection and isolation of f_ψ is based on the model made of Eqs. (7.115) and (7.116). This system is detectable and the innovation, r_ψ, of the Kalman filter based on the above model is affected by fault f_ψ as can be seen in Fig. 7.20b. However, the latter fault is not strongly detectable.

Hence for evaluation of residual r_ψ, one has to resort to the GLR algorithm, since it is not possible to detect fault disappearance by successive reinitilisation of a CUSUM algorithm. The latter option is possible for evaluation of r_{ω_3} however. Figure 7.21 represent the CUSUM decision function obtained by processing the residual of Fig. 7.20a and the GLR decision function obtained by processing the residual sequence of Fig. 7.20b. Repeated alarms are issued by the CUSUM algorithm, the first occurring at time 300, the last one at time 597. In this time interval, the CUSUM decision function crosses its threshold every 5 samples on the average. Appearance and disappearance of the fault on the angular rate measurement can thus be detected and isolated. As far as the GLR decision function of Fig. 7.21b is concerned, it reaches its threshold at time 904, and the estimated fault occurrence time of f_ψ is 900 (actual value 901). The estimated fault magnitude based on the residual in the time window

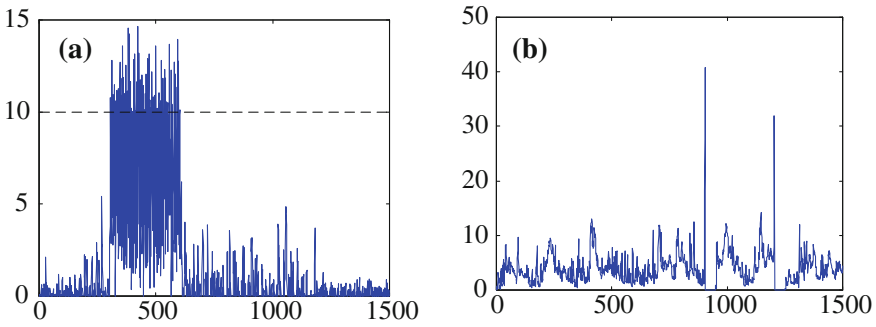


Fig. 7.21 CUSUM decision function and GLR decision function resulting from evaluation of r_ψ (**a**) and r_{ω_3} (**b**)

[900 949] is 0.449, which is in error by 10 %. After reinitialisation, the GLR algorithm detects fault disappearance at time 1203 and it provides instant 1201 as the estimate of the change occurrence time, namely the correct time instant. The estimated change magnitude is −0.618 which is in error by 23 %.

7.4 Exercises

Exercise 7.1 Covariance of high-pass filter output–input is band-limited noise
Given a high-pass filter with the state-space representation

$$\begin{aligned}\dot{x}(t) &= -\alpha x(t) + \alpha w(t)\\ y(t) &= w(t) - x(t)\end{aligned}$$

with input $w(t)$, a band-limited random signal generated by

$$\dot{w}(t) = -\beta w(t) + \sigma_w\sqrt{2\beta}v(t),$$

where $v(t)$ has the intensity $S_v = 1$.

1. Represent the filter in the form

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{v}(t).$$

2. Let the covariance matrix be

$$\boldsymbol{Q} = E\left\{\begin{pmatrix} w(t)\\ x(t)\end{pmatrix}\begin{pmatrix} w(t) & x(t)\end{pmatrix}\right\} = \begin{pmatrix} a & c\\ c & b\end{pmatrix}.$$

   Calculate the covariance $\boldsymbol{Q}$ as the solution to the Lyapunov equation

$$\boldsymbol{A}\boldsymbol{Q} + \boldsymbol{Q}\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{B}\boldsymbol{S}_{\mathrm{v}}\boldsymbol{B}^{\mathrm{T}} = \boldsymbol{O}.$$

3. Show that the variance on y , σ_y^2, is

$$\sigma_y^2 = \frac{\beta}{\alpha+\beta}\sigma_w^2 \tag{7.133}$$

   and determine the value of the pole α required to obtain a desired value of σ_y^2 given σ_w^2.
□

Exercise 7.2 Industrial actuator-fault detection using threshold and CUSUM tests
The task is to implement a residual generator in, e.g. MATLAB/Simulink and investigate how faults propagate in the residual generators in the presence of measurement noise and random disturbances. The system is the industrial actuator used in Exercise 3.4. The residual generator is described by the parity equations

$$\begin{aligned} r_1(s) &= n_{\mathrm{m}}(s) - \frac{1}{sI_{\mathrm{tot}}+\alpha} k_q \eta i_{\mathrm{m}}(s) \\ r_2(s) &= \frac{\tau_f s}{\tau_f s+1}\theta_{\mathrm{m}}(s) - \frac{1}{N(\tau_f s+1)}\tau_f n_{\mathrm{m}}(s) \end{aligned} \tag{7.134}$$

where $\tau_f = 1\,\mathrm{s}$. A noise specification for the measurements i_{m}, n_{m} and θ_{m} is given by the autocorrelation function

$$R_{ii}(\tau) = \sigma_i^2 e^{-\beta_i|\tau|} \tag{7.135}$$

where

$$\begin{aligned} n_{\mathrm{m}} &: \sigma_n = 3\ \mathrm{rad/s} \quad \beta = 10 \\ \theta_{\mathrm{m}} &: \sigma_\theta = 0.01\ \mathrm{rad} \quad \beta = 2 \\ i_{\mathrm{m}} &: \sigma_i = 1\ \mathrm{mA} \quad \beta = 10 \end{aligned} \tag{7.136}$$

The system is depicted in Fig. 3.8 where also the parameters are given.

1. Implement a simple threshold (level) detector on the two residuals from the parity equations (Eq. 7.134). Investigate whether you can detect
   (a) f_n: a step change of 3.5 rad/s in the speed feedback.
   (b) f_i: a step change of 0.15 A in the power drive current.
2. Design a scalar CUSUM detector of change in mean value. Test for the hypothesis that a fault is present and reflected in the mean value change of the values given above.
3. Design a detector for the speed sensor fault that has a time to detect of 2.5 s. Determine the average time between false alarms. Increase the specified time to detect to 10 s and determine the new average between false alarms.
4. Investigate experimentally (by simulation) whether the two faults can be detected.
5. Verify the time to detect and the false alarm rates using different seeds in your measurement noise generators.
   Note 1: The results on time to detect and mean time between false alarms assume white noise statistics of the log-likelihood test quantity $s(k)$. When $s(k)$ is not white, the statistical results are only approximate figures that can only be used as guidelines for design. □

Exercise 7.3 Industrial actuator change detection using the GLR test
Design a generalised likelihood ratio test (GLRT). Assume that faults are of unknown magnitude and a GLRT-based detector is hence needed. The simulation of the velocity controlled actuator from Exercise 7.2 is reused.

1. Design and implement a scalar GLRT-based detector for residual r_1. Implementation in Simulink is conveniently done using an *embedded MATLAB* block from the Simulink *user defined functions* part of the block set.
2. Simulate the system without any fault and with a fault on n_{m}. The no-fault case is denoted $\mathcal{H}_0$, the one with a fault is $\mathcal{H}_1$. Plot the histograms of $g(k)$ for the two cases: (1) no faults, (2) $f_n = 3.5\,\mathrm{rad/s}$. Make also a probability plot of the two distributions using the MATLAB *probplot* command.
3. Determine the threshold you should use for the GLRT to get a false alarm probability of $P_{FA} \leq 0.001$.
4. Investigate which influence the choice of GLRT window size M has on the choice of threshold for the test. □

Exercise 7.4 Single-axis satellite: Detection of a fault of known magnitude
Referring to Exercise 6.3, measurements on the satellite are subject to noise. A state-space model for the single-axis satellite is given by:

$$
\begin{aligned}
\dot{x_1} &= \tfrac{1}{I}(u_1 k_1 + u_2 k_2 + d) \\
\dot{x_2} &= x_1 \\
y_1 &= x_1 + f_1 + w_1 \\
y_2 &= x_2 + f_2 + w_2 \\
y_3 &= x_2 + f_3 + w_3 \\
y_4 &= u_1 k_1 + f_4 \\
y_5 &= u_2 k_2 + f_5 \\
\dot{x_1} &= \tfrac{\mathrm{d}x_1}{\mathrm{d}t} \\
\dot{x_2} &= \tfrac{\mathrm{d}x_2}{\mathrm{d}t}
\end{aligned}
\tag{7.137}
$$

The noise specification is given as

$$
R_{ii}(\tau) = \sigma_i^2 e^{-\beta_i |\tau|} \tag{7.138}
$$

where the standard deviations are

$$
\begin{aligned}
w_1 &: \sigma_1 = 2 \times 10^{-4}\,\text{rad/s} \quad \beta = 10 \\
w_2 &: \sigma_2 = 1 \times 10^{-5}\,\text{rad} \quad \beta = 10 \\
w_3 &: \sigma_3 = 2 \times 10^{-3}\,\text{rad} \quad \beta = 10
\end{aligned}
\tag{7.139}
$$

Consider the two residuals

$$
\begin{aligned}
r_{12}(s) &= \tfrac{1}{s+\alpha} y_1(s) - \tfrac{s}{s+\alpha} y_2(s) \\
r_{23}(s) &= y_2(s) - y_3(s)
\end{aligned}
\tag{7.140}
$$

where $\alpha = 0.01$. The system parameters are given in Exercise 6.3. For residual generation, measurements are sampled with a sampling interval of 1 s

1. Design a set of scalar-based change detection algorithms for the case where the fault on y_2 has a magnitude of $\mu_{21} = 2 \times 10^{-3}$ rad and the change is a step. Verify that you are able to detect the fault within 10 samples. It is desired to use r_{23} to isolate the fault but the variance on y_3 is too large to detect a change of this magnitude.
2. Determine which variance y_3 should have to isolate the fault on y_2 within 10 samples. Design a low-pass filter to obtain such reduced variance of y_3 and verify by simulation that isolation is possible. □

Exercise 7.5 Single-axis satellite: Vector-based detection of change signature
Consider again the system (7.137)–(7.139) and residual generator (7.140).

1. Determine the fault signature in the residual vector assuming a fault on y_2 appears as a step.
2. Design a vector-based change detection algorithm for the case where the fault on y_2 has a magnitude of 2×10^{-3} rad and appears as a step. Discuss the properties of the vector-based change detection compared with the set of scalar algorithms. □

Exercise 7.6 Hardware redundancy
Consider a set of three sensors measuring a single quantity denoted x. The measurement system can be modelled as:

$$\begin{bmatrix} y_1(k) \\ y_2(k) \\ y_3(k) \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} f_1(k) \\ f_2(k) \\ f_3(k) \end{bmatrix} + \begin{bmatrix} v_1(k) \\ v_2(k) \\ v_3(k) \end{bmatrix} \tag{7.141}$$

where $y_i(k)$, $v_i(k)$ and $f_i(k)$, $i = 1, 2, 3$, respectively, denote the sensor measurement, the measurement noise and a possible additive fault. $v_i(k)$, $i = 1, 2, 3$ is assumed to be a Gaussian white noise sequence with zero mean and variance $\sigma_1^2 = \sigma_2^2 = 1$ and $\sigma_3^2 = 2$. Besides, the noise sequences are mutually uncorrelated. The faults are modelled as unknown deterministic signals, and it is assumed that simultaneous faults do not occur. A fault on sensor i thus corresponds to a non-zero value for $f_i(k)$ from an unknown fault occurrence time, say k_0.

1. Let us rewrite Eq. (7.141) more compactly as

$$\boldsymbol{y}(k) = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^{\mathrm{T}} \boldsymbol{x}(k) + \boldsymbol{f}(k) + \boldsymbol{v}(k) \tag{7.142}$$

   where $\boldsymbol{y}(k) = \begin{bmatrix} y_1(k) & y_2(k) & y_3(k) \end{bmatrix}^{\mathrm{T}}$ and similarly for $\boldsymbol{f}(k)$ and $\boldsymbol{v}(k)$. Let Ω denote a basis for the left null space of $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^{\mathrm{T}}$. Show that $\boldsymbol{r}(k) \equiv \Omega \boldsymbol{y}(k)$ is independent of $x(k)$.
2. Prove that $\boldsymbol{r}(k)$ has zero mean in the absence of fault and that its mean is equal to $\Omega_{\cdot,i} f_i(k)$ in the presence of a fault on sensor i. Here $\Omega_{\cdot,i}$ denotes the ith column of matrix Ω. Besides, show that the variance of $\boldsymbol{r}(k)$ is equal to $\boldsymbol{Q}_r = \Omega \boldsymbol{Q} \Omega^{\mathrm{T}}$ where $\boldsymbol{Q} = \text{ diag } (\sigma_1^2, \sigma_2^2, \sigma_3^2)$
3. Let $\boldsymbol{r}_n = (\Omega \boldsymbol{Q} \Omega^{\mathrm{T}})^{-\frac{1}{2}} \boldsymbol{r}(k)$. Show that $\boldsymbol{r}_n(k)$ is distributed as $\mathcal{N}(O, \boldsymbol{I})$ when no faults are present and as $\mathcal{N}((\Omega \boldsymbol{Q} \Omega^{\mathrm{T}})^{-\frac{1}{2}} \Omega_{\cdot,i} f_i(k), \boldsymbol{I})$ upon occurrence of a fault on the ith sensor.
4. Assume that positive step-like faults with minimum magnitude equal to 0.5 can occur. Design a multi-CUSUM algorithm of the form presented in Sect. 7.2.6 to detect and isolate such faults.
5. Generate a data sequence according to model (7.141) in which $x(k) = \sin(0.1k)$, the noise properties are as described above, and $f_1(k) = 0.5 \ 1_{\{k \geq 20\}}$, $f_2(k) = f_3(k) = 0$ for all k.
6. Process the data generated in point (e) by the multi-CUSUM algorithm designed in point (d), and check its effectiveness.
7. Check how the magnitude of the fault affects the obtained results by repeating points (e) and (f) for different fault magnitudes. □

7.5 Bibliographical Notes

Change detection. The non-sequential and sequential algorithms for change detection in signals are described in details in the book [11]. The proof of the optimality of tests based on the likelihood ratio between two hypotheses can be found in Sect. 4.2.2 of [11]. The properties of sequential algorithms deduced from the ARL function are investigated in Chap. 4 of [11]. A heuristic approach for choosing the design parameters of the GLR algorithm for detection of changes in the mean is presented in [275]. The statistical properties of the multi-CUSUM algorithm for change detection and isolation are studied in [245]. The algorithm has been extended to account for a dynamical profile of change in [178], and the benefit of the extension has been demonstrated on a case study.

Kalman filters as residual generator. A numerically stable algorithm to extract the observable part of a given system can be found on p. 220 of [63]; it can be used as a first step to design a residual generator based on a Kalman filter for an unobservable system. The design of a residual generator based on a Kalman filter for a system subject to unknown input and additive faults is adapted from [246]. An alternative approach to compute an innovation sequence is to use parity relations and to filter the obtained residual by an appropriate whitening filter. This method has been considered in [275] for instance.

Robustness and alternative methods for FDI of additive faults. It was not possible to examine here the question of robustness with respect to modelling uncertainties of the statistical approach to fault detection. A valuable reference to study this question is [228]. An alternative to the combined use of an innovation filter and a statistical change detection/isolation algorithm for the diagnosis of additive faults in linear systems has recently been proposed in [257]. The formulation consists of a constrained least-squares problem with sum-of-norm regularisation. Quantifying fault diagnosability is an important issue that has been studied in [93] by resorting to a statistical measure, namely, the Kullback–Leibler divergence.

Fault diagnosis based on parameter estimation. System identification based methods for fault detection, estimation and isolation have not been considered in this chapter but they have also proved useful in many applications. They can be separated in two classes: methods based on explicit parameter estimation and methods based on statistics (such as the statistical local approach). An introduction to the first class of methods can be found in [153, 154]. For the second class, the reader is referred to [11, 408].

Appendix on random variables and stochastic processes. For more information on the material in the appendix on random variables and stochastic processes, the reader can consult [9, 166, 261] for instance. In particular the approach for sampling a linear stochastic differential equation is borrowed from pp. 147–151 of [9].

Chapter 8
Reconfigurability Analysis

Abstract Fault-tolerant control brings together several theoretic frameworks that are needed to treat the different problems it involves. This chapter addresses these problems from a global perspective that includes the *specification* and the development of *control solutions*, as well as the *implementation* and the *evaluation* of these solutions. Among many possible control problems, this chapter uses linear quadratic control theory to illustrate the above-mentioned problems, under the two possible fault-tolerance strategies, namely *fault accommodation*, where the controller parameters are adapted to the parameters of the faulty plant, and *system reconfiguration*, where the subset of system components in operation is changed (and so is of course the control law). A variety of other control approaches will be developed in the next chapter.

8.1 The Fault-Tolerant Control Problem

8.1.1 Standard Control Problem

In order to explain the fault-tolerant control problem in more detail, the standard control problem is first stated as a problem that is defined by a given objective, a set of constraints and a set of admissible control laws. Standard control aims at finding a control law in a given set of control laws U, such that the controlled system achieves the control objectives O, while its behaviour satisfies a set of constraints C. Thus, the solution of the problem is completely defined by the triple $< O,\ C,\ U >$.

Problem 8.1 (*The control problem*)
Solve the problem $< O, C, U >$.

The following remarks should explain this problem in more detail:

- The set U of admissible control laws defines the algorithms that can be implemented, e.g. open-loop control (a mapping from the time domain to the control space), closed-loop control (a mapping from the output $\times$ reference spaces to the control space), using continuous or discrete-valued arguments for the variables,

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8_8

allowing for continuous or discontinuous, differentiable or non-differentiable mappings, etc.

- The objective O defines what the system is expected to achieve, when controlled by the above-mentioned control law. It may range from very general statements (e.g. achieve closed-loop stability) to much more specific ones (e.g. reach a given point, on a given circular orbit around the earth, at a given time, for a space *rendez-vous*).
- Constraints C are functional relations that the behaviour of the controlled system must satisfy over time. They are expressed by algebraic and differential or difference equations, when continuous variables are considered, and by other models when discrete values are of interest (see Chap. 2). Inequality constraints express that some saturations act on the system admissible solutions (e.g. any trajectory which results from an admissible control law must end on a given point of a given circular orbit, at a given time, but the energy consumed all along the trajectory is limited by the capacity of the vessel's reservoirs).

Example 8.1 Control of the single-tank system
Consider the problem of filling an initially empty tank up to a certain mass m of some liquid, as fast as possible, so as to start a batch operation process in the food industry. Let $m(t)$ be the mass present in the tank at time t, and let $u(t)$ be the controlled input flow. The control problem is a very classical minimum time problem defined by the triple:

O: The objective is to change the mass $m(t)$ from its initial value $m(t_0) = 0$ to the given final value $m(t_\mathrm{f}) = M$, in minimum time, i.e. minimising the functional

$$\int_{t_0}^{t_\mathrm{f}} \mathrm{d}t.$$

C: The behaviour of the system is constrained by the state equation

$$\dot{m}(t) = u(t).$$

U: The control law belongs to the class of piecewise continuous open-loop controls with saturation

$$\begin{aligned} u:\ & |\mathcal{R}^+ \to |\mathcal{R} \\ & t \longmapsto u(t) \\ & u(t) \in [0,\ u_{\max}] \\ & u \in \mathcal{C}^0 \end{aligned}$$

Once the tank has been filled with the mass m of liquid, the production process of the batch will start. Assume that for the proper biological reaction to take place, the temperature $\eta(t)$ must be regulated around some given reference η_{ref}. The associated control problem is again well known. A PI-regulation example is given below:

O: The objective is to regulate the temperature $\eta(t)$ of the batch around the reference value η_{ref}, during the processing period $[0, T]$. It is expressed by

$$|\eta_{\text{ref}} - \eta(t)| \leq \lambda, \ \forall t \in [T_0, \ T],$$

where λ is some constant which defines the admissible temperature excursion around the reference on the time interval $[T_0, \ T]$, and $T_0 > 0$ is the time value after which the neighbourhood of the reference temperature must have been reached.

C: The behaviour of the system is described by the state and measurement equations

$$\begin{aligned}\dot{\eta}(t) &= f(\eta(t)) + \zeta(t) + v(t)\\ \dot{\zeta}(t) &= g(\eta(t), \ t)\\ y(t) &= \eta(t) + \varepsilon(t),\end{aligned}$$

which expresses that the temperature variation is the result of thermal losses $f(\eta(t))$, of the exothermic character of the biological reaction (the reaction energy is $\zeta(t)$, whose evolution is given by $g(\eta(t), t)$ in the second state equation), and of the control $v(t)$, and that the available measurement signal is the temperature, which is corrupted by some measurement error $\varepsilon(t)$.

U: The control law belongs to the class of closed-loop PI controls

$$\begin{aligned}v: \ & \Psi_{\text{ref}} \times Y \rightarrow |\mathcal{R}\\ & (\eta_{\text{ref}}, \ y(t)) \longmapsto v(t) = k_{\text{P}}(\eta_{\text{ref}} - y(t)) + k_{\text{I}} \int_0^t (\eta_{\text{ref}} - y(\tau)) \, \mathrm{d}\tau,\end{aligned}$$

where k_{P} and k_{I} are coefficients to be found as the solution of the control problem, Ψ_{ref} is the set of possible references and Y is the set of the sensor output values. □

8.1.2 Impacts of Faults on the Control Problem

Fault-tolerant control is concerned with the control of the faulty system. This can be done by changing the control law without changing the plant which is operated (this is the *fault accommodation* strategy), or by changing both the control and the system (this is the *reconfiguration* strategy). Since the control algorithm just implements the solution of a control problem for a given system, changing the control or the system means that the control problem has been changed as the result of faults. In order to understand the different strategies which can be applied to the design of fault-tolerant control, let us first consider the impact of faults on the control problem $< O, \ C(\theta), \ U >$, where $C(\theta)$ denotes the dependency of the constraint C upon the parameter θ, which in turn depends upon the fault. The different fault-tolerant control strategies will then be introduced, as a consequence of the available knowledge.

System objectives. The occurrence of faults should not change the system objectives. The objectives are associated with the users (they define what the users expect the

system to achieve), and the very nature of fault-tolerant control is to still try to achieve these objectives, *in spite* of the faults. However, this will be possible or not. Therefore, two cases have to be distinguished:

1. There is a mean of still achieving the system objectives in the presence of certain faults. The system is said to be fault tolerant, with respect to that objectives and to these faults. Equivalently, the faults are said to be recoverable. The control engineer's task is to design some control law which is able to do that.
2. The objectives cannot be achieved in the presence of the considered faults. The system is not fault tolerant with respect to that objectives and these faults, in other terms the faults are not recoverable. However, it is not enough to stand with this conclusion. The control engineer should provide, in this case, indications about what to do with the system. Since the current objectives cannot be achieved, the problem is transformed into finding new objectives that are of interest in the current situation, and to design the structure and the parameters of the new control law to achieve these new objectives.

System constraints. The occurrence of faults may obviously change the constraints $C(\theta)$ of the control problem.

- First, the constraints may remain the same but the parameters may change, thus transforming the control problem $< O, C(\theta_\mathrm{n}), U >$ into the problem $< O,\ C(\theta_\mathrm{f}),\ U >$, where θ_n (respectively θ_f) denotes the nominal (respectively the faulty) system parameters.
- Second, the constraints themselves might change, transforming the control problem $< O,\ C_\mathrm{n}(\theta_\mathrm{n}),\ U >$ into the problem $< O,\ C_\mathrm{f}(\theta_\mathrm{f}),\ U >$, where C_n is the set of nominal constraints, and $C_\mathrm{f}(\theta_\mathrm{f})$ is a set of new constraints with new associated parameters.

Both cases may be summarised by the change of $C_\mathrm{n}(\theta_\mathrm{n})$ into $C_\mathrm{f}(\theta_\mathrm{f})$ since the change of parameters only is a particular case, described by $C_\mathrm{f} = C_\mathrm{n}$.

Example 8.2 A tank with two exit pipes
Consider for example a tank with two exit pipes, respectively, situated at levels l_1 and l_2 metres. The system nominal constraints are the following:

$$\begin{aligned} x(t) \in [0,\ l_1[\ \ \dot{x}(t) &= q_i(t) \\ x(t) \in [l_1,\ l_2[\ \ \dot{x}(t) &= q_i(t) - q_1(v_1, t) \\ x(t) \geq l_2\ \ \dot{x}(t) &= q_i(t) - q_1(v_1, t) - q_2(v_2,\ t), \end{aligned}$$

where $x(t)$ is the level in the tank, $q_i(t)$ is the input flow, $q_1(v_1,\ t)$ (respectively $q_2(v_2, t)$) is the output flow through pipe 1 (respectively through pipe 2) which depends on some external variable or control signal v_1 (respectively v_2). This might be, for example, the level in another tank connected to the pipe, or the control signal of an output valve on the pipe. Suppose now that pipe 1 is clogged, then as the result of the fault, the system constraints become

$$\begin{aligned} x(t) &\in [0,\ l_1[\ \dot{x}(t) = q_i(t) \\ x(t) &\in [l_1,\ l_2[\ \dot{x}(t) = q_i(t) \quad \forall v_1(t) \\ x(t) &\geq l_2\ \dot{x}(t) = q_i(t) - q_2(v_2,\ t), \end{aligned}$$

which can also be represented by adding a fourth constraint to the three nominal ones

$$\begin{aligned} x(t) &\in [0,\ l_1[\ \dot{x}(t) = q_i(t) \\ x(t) &\in [l_1,\ l_2[\ \dot{x}(t) = q_i(t) - q_1(v_1,\ t) \\ x(t) &\geq l_2\ \dot{x}(t) = q_i(t) - q_1(v_1,\ t) - q_2(v_2,\ t) \\ & q_1(v_1,\ t) = 0, \forall v_1,\ t. \quad \square \end{aligned}$$

Admissible control laws. The occurrence of faults may also change the set of admissible control laws since faults may occur in the computing and communication devices in which they are implemented. As in the previous subsection, the new set of admissible control laws is noted U_{f} while the nominal one is U_{n}.

Example 8.1 (cont.) Control of a single-tank system
Consider the standard control problem of filling a tank in minimum time for processing a batch in food industry: find the control law in the set U_{n} of piecewise continuous functions satisfying

$$\begin{aligned} u:\ & |\mathcal{R}^+ \rightarrow |\mathcal{R} \\ & t \longmapsto u(t) \\ & u(t) \in [0,\ u_{\max}] \\ & u \in \mathcal{C}^0 \end{aligned}$$

such that the initial mass $m(t_0) = 0$ is changed into the final mass $m(t_{\mathrm{f}}) = M$, in minimum time, while satisfying the state equation $\dot{m}(t) = u(t)$. Suppose now that the pump is faulty and can only deliver a fraction of its nominal maximum output flow, namely $u_{\max}$ is changed into $u'_{\max} < u_{\max}$. The set U_{n} is changed into the set U_{f}, where the saturation level is lower (thus leading to a larger filling time for the optimal solution). $\square$

8.1.3 Passive Versus Active Fault-Tolerant Control

In the passive approach, the control algorithm is designed so that the system is able to achieve its given objectives, in healthy as well as in faulty situations, without any change in the control law. Therefore, passive fault-tolerant control sets the control problem in a context, where the ability of the system to achieve its given objective is preserved, using the same control law, whatever the system situation (healthy or faulty).

In active approaches, the control law is changed when faults occur, so that the ability of the system to achieve its given objective is preserved, using a control law adapted to each fault situation. Therefore, active fault-tolerant control algorithms

implement the solution of problems which are specifically set for each of the possible (healthy and faulty) situations.

As the result of faults, the control problem is transformed

$$\text{from} \quad < O,\ C_{\mathrm{n}}(\theta_{\mathrm{n}}),\ U_{\mathrm{n}} > \quad \text{into} \quad < O,\ C_{\mathrm{f}}(\theta_{\mathrm{f}}),\ U_{\mathrm{f}} >.$$

Suppose that both $C_{\mathrm{f}}(\theta_{\mathrm{f}})$ and U_{f} are perfectly known, then the fault-tolerant control law has to solve $< O,\ C_{\mathrm{f}}(\theta_{\mathrm{f}}),\ U_{\mathrm{f}} >$. If such a solution exists, the system is fault tolerant with respect to the objective O and the fault situation $C_{\mathrm{f}}(\theta_{\mathrm{f}}),\ U_{\mathrm{f}}$. If the problem $< O,\ C_{\mathrm{f}}(\theta_{\mathrm{f}}),\ U_{\mathrm{f}} >$ has no solution, then the system is not fault tolerant and objective reconfiguration has to be explored, as previously explained.

The difference between passive and active fault-tolerant control can now be very simply explained.

Passive fault tolerance. In passive fault tolerance, the control law is not changed when faults occur. This means that the system objectives can be obtained when the system is healthy (thus it solves $< O,\ C_{\mathrm{n}}(\theta_{\mathrm{n}}),\ U_{\mathrm{n}} >$), as well as when the system is faulty (thus it also solves $< O,\ C_{\mathrm{f}}(\theta_{\mathrm{f}}),\ U_{\mathrm{f}} >$). Implementing passive fault tolerance for a given set of faults means that there is a common solution to problem $< O,\ C_{\mathrm{n}}(\theta_{\mathrm{n}}),\ U_{\mathrm{n}} >$ and to all problems $< O,\ C_{\mathrm{f}}(\theta_{\mathrm{f}}),\ U_{\mathrm{f}} >$, $(f \in \mathcal{F})$, where $\mathcal{F}$ indexes the set of all the considered faults.

This is a very particular situation, which is fulfilled, in general, only for objectives associated with very low levels of performances (this is a so-called *conservative* approach). Note that since the control law is not changed, the passive fault-tolerance approach is similar to the robust approach when uncertain systems are considered (cf. Chap. 1). Indeed, faults can be considered as uncertainties which affect the system parameters. The difference lies not only in the size and interpretation of these changes, but also in the fact that the structure of the constraints may change as the result of faults.

Active fault tolerance. In active fault tolerance, each of the problems

$$< O,\ C_{\mathrm{n}}(\theta_{\mathrm{n}}),\ U_{\mathrm{n}} > \text{ and } < O,\ C_{\mathrm{f}}(\theta_{\mathrm{f}}),\ U_{\mathrm{f}} >,$$

$f \in \mathcal{F}$, has its own specific solution, thus allowing for much more demanding objectives. However, for each of these problems to be solved the knowledge about $C_{\mathrm{f}}(\theta_{\mathrm{f}})$ and U_{f} must be available. This is the role of fault detection and isolation algorithms. This chapter deals with active fault-tolerant control.

8.1.4 Available Knowledge

Providing information about the fault impact is the aim of the fault diagnosis algorithms. However, the power and efficiency of these algorithms are limited. Fault detection informs that the problem to solve is no longer $< O,\ C_{\mathrm{n}}(\theta_{\mathrm{n}}),\ U_{\mathrm{n}} >$. Fault

isolation informs about the subset of the constraints $C_n(\theta_n)$ which are unchanged (those associated with the still healthy components), and the subset $U_f \subseteq U_n$ of control laws which can still be used. The knowledge about the changed constraints calls for fault estimation, which is a new function to be considered for the design of fault-tolerant control. According to its performances, three cases must be considered:

1. The fault diagnosis algorithm is able to provide an estimate $\hat{C}_f(\hat{\theta}_f)$, $\hat{U}_f$ of the fault impact. Then, the problem to be solved is the standard control problem $< O, \hat{C}_f(\hat{\theta}_f), \hat{U}_f >$. Note that, when a solution exists, there is still a risk that the actual faulty system (described by $C_f(\theta_f)$ and U_f) fails to satisfy the objectives O, although the available model of the faulty system does satisfy them.
2. The fault diagnosis algorithm is able to provide an estimate $\hat{\Gamma}_f(\hat{\Theta}_f)$, U_f of the fault impact, where $\hat{\Gamma}_f$ is a set of possible constraints and $\hat{\Theta}_f$ is a set of associated parameters. Then the problem to be solved is the robust control problem $< O, \hat{\Gamma}_f(\hat{\Theta}_f), \hat{U}_f >$. When a solution exists, the actual faulty system will satisfy the objectives O provided the actual constraints $C_f(\theta_f) \in \hat{\Gamma}_f(\hat{\Theta}_f)$, otherwise, the same risk as above exists.
3. The fault diagnosis algorithm detects and isolates the faults, but it cannot provide any estimate of the fault impact. The control engineer is faced with the problem of designing the control of a completely unknown system, which is not possible. Obtaining knowledge about that system could be thought of, using e.g. learning approaches, but then an estimation of the fault impact could indeed be obtained, which would bring the problem back to case 2.

Other possible cases are those where the fault diagnosis system detects the fault, but it cannot isolate it nor is it able to provide any estimate, and the case, where the fault diagnosis system does not even detect the fault. In the first case, the only possibility to keep mastering the system is to move to a fall back mode, while in the second case, any catastrophic behaviour is possible. Active fault tolerance is only concerned with cases 1, 2 and 3.

8.1.5 Active Fault-Tolerant Control Strategies

Fault accommodation. Fault accommodation is the fault-tolerant control strategy which is associated with cases 1 and 2. It solves the problem $< O, \hat{C}_f(\hat{\theta}_f), \hat{U}_f >$ or $< O, \hat{\Gamma}_f(\hat{\Theta}_f), \hat{U}_f >$, which is associated with the control of the faulty system. The fault situation can be accommodated with respect to the objectives O when the problem has a solution.

Problem 8.2 (*Fault accommodation problem*)
Solve the control problem $< O, \hat{C}_f(\hat{\theta}_f), \hat{U}_f >$, where $\hat{C}_f(\hat{\theta}_f)$ is the estimate of the actual constraints provided by the fault diagnosis algorithms.

Note that the interpretation of fault accommodation is that it is a strategy by which the faulty system is controlled in a specific way, so as to still achieve the objectives which were (before the fault) achieved by the healthy system.

System reconfiguration. System reconfiguration is the fault-tolerant control strategy which is associated with case 3. Remind that in this case the faulty system is absolutely unknown, but the control engineer wishes to design a control that achieves the system objectives. The only means to set any control problem is to switch off the faulty components (which are known from the isolation function), and to try to achieve the objectives using only the remaining (healthy) ones. Let $C_{\mathrm{f}}(\theta_{\mathrm{f}}) = C'_{\mathrm{n}}(\theta_{\mathrm{n}}) \cup C''_{\mathrm{f}}(\theta_{\mathrm{f}})$, where $C'_{\mathrm{n}}(\theta_{\mathrm{n}})$ is the subset of the constraints which are associated with the healthy part of the system, and $C''_{\mathrm{f}}(\theta_{\mathrm{f}})$ is the subset of the constraints which are associated with the faulty part.

Problem 8.3 (*Reconfiguration problem*)
Find a new set of system constraints $C_{\mathrm{f}}(\theta_{\mathrm{f}})$ such that the control problem $< O, C_{\mathrm{f}}(\theta_{\mathrm{f}}), U >$ has a solution, find and activate this solution.

The choice of a new set of constraints will imply that the input–output relations between the controller and the plant are changed.

Note that the constraints $C'_{\mathrm{n}}(\theta_{\mathrm{n}})$ are known while $C''_{\mathrm{f}}(\theta_{\mathrm{f}})$ are unknown. Using similar notations, let $U_{\mathrm{f}} = U'_{\mathrm{n}} \cup U''_{\mathrm{f}}$. Then, the reconfiguration strategy solves the problem $< O, C'_{\mathrm{n}}(\theta_{\mathrm{n}}), U'_{\mathrm{n}} >$, i.e. it tries to achieve the system objectives by controlling only the healthy part of the system.

Fault accommodation and reconfiguration are distinguished according to whether the I/O signals between the controller and the plant are changed. Reconfiguration implies the use of different I/O relations between the controller and the system. Switching the system to a different internal structure, to change its mode of operation, is an example of such I/O switching. Accommodation does not use such means.

Both fault accommodation and system reconfiguration strategies may need new control laws in response to faults. They also have to manage transient behaviour, which result from the change of control law or change of the constraints' structure.

8.1.6 Supervision

Suppose that the accommodation and the reconfiguration strategies fail to provide a solution. This means that there is no possibility, using the faulty system (accommodation) or a subsystem of it (reconfiguration) to achieve the objective. In this case, another objective has to be provided to the system. This introduces the most general problem, defined by the 3-tuple $< \mathcal{O}, \mathcal{C}(\theta), \mathcal{U} >$, where $\mathcal{O}$ is a set of possible control objectives. This problem is called the supervision problem. It is a decision problem in which the system objective is not pre-defined, but has to be determined, according to the system possibilities at each time, taking into account the actual system possibilities.

A supervision problem is thus a fault-tolerant control problem associated with a decision problem: if faults are such that fault tolerance cannot be achieved, the system goal itself has to be changed. When far-reaching decisions with respect to the system goal have to be taken, human operators are generally involved.

It may happen that no achievable objective exists under the actual system possibilities. This can be a design error, or a deliberate choice to accept certain failure scenarios, e.g. for reasons of benefit or small likelihood of certain events. Note that fail-to-safe conditions are intended to avoid this case in some situations, since they express that for certain classes of faults, the objective of stopping the system must always be achievable.

8.2 Fault-Tolerant Control Architecture

The method to achieve fault tolerance, which is considered in this chapter, relies on employing fault diagnosis schemes on-line and on reacting to the results of the diagnosis. A discrete-event signal to a supervisor agent is generated by the diagnostic algorithm when a fault is detected, another when it is isolated. This activates an alternative control that is supposed to handle the fault. The control for the particular case could be pre-determined for each type of critical fault or obtained from real-time analysis and on-line re-design. In any event, the design process must run through a number of cases equal to the number of faults to be handled and the control system needs to be re-designed for each such case. Some types of faults in sensors and actuators are simple to handle, others require a detailed re-design. It is, thus, worthwhile to first consider the simplest possibilities, thereafter the more general and complicated case of re-design.

The architecture of a fault-tolerant control system is illustrated in Fig. 8.1. A fault is a discrete event that acts on a system and by that changes some of its properties. Having diagnosed a fault, a decision needs to be taken about a remedial action.

The goal of fault-tolerant control is to respond to the occurrence of a fault such that the faulty system still satisfies the given specifications. Due to the discrete nature of the fault occurrence and reconfiguration, fault-tolerant control systems are hybrid in nature (cf. Sect. 3.7). In the figure, σ_{f} denotes fault events, σ_{a} are control events reconfiguring the system and q_{c} the control mode, which selects a control law. The actual physical mode q_{p} of the plant may be viewed as the discrete state of an automaton which is driven by plant internal events σ_{p}, the fault events σ_{f} and the control events σ_{a}. While many different approaches can be used to solve the fault-tolerant control problem, it may not be possible to control the system to a desired performance for an arbitrary change of parameter or structure. A final remedial action is then to close down to a safe state should proper control not be possible. The key issue is to be able to obtain certain specified properties of the control of the faulty system, and this chapter, therefore, focusses on methods for re-design based on specifications.

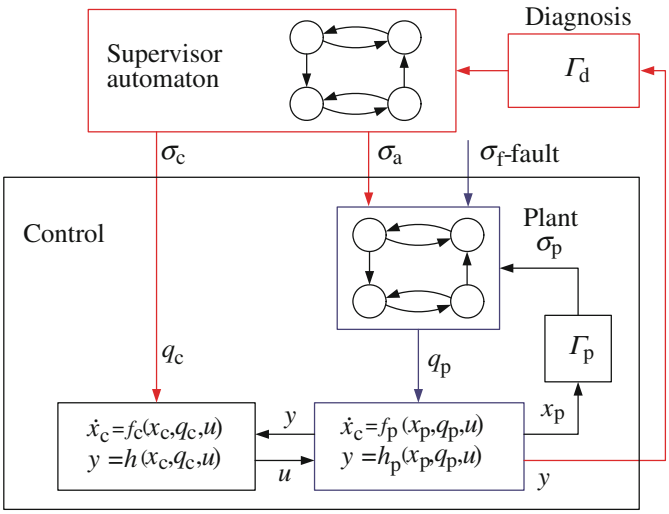


Fig. 8.1 The plant can change in a discrete way through change in states, a plant fault can cause a discrete event. Plant architecture can be changed by switch-over functions. Parameters or structure of the controller can be changed by logic in a supervisor automaton. The automaton gets its input from fault diagnosis

A fault in the plant can affect the structure and the parameters of a plant. The complexity of designing a controller for the faulty system is therefore immense, and there is no single, systematic way to design a control system with reconfiguration as depicted. Most research work deal either with diagnosis or controller reconfiguration, but not both. One approach is based on a bank of controllers, each one being associated to a healthy or a faulty plant working mode. The selection of the controller to be used for the present working mode must be assumed to be achieved with some delay and possibly false alarms.

The theory of logic-based switching control also relies on a bank of controllers (Fig. 8.2). It has recently been used for fault-tolerant control. The supervisor is made of a set of estimators, followed by performance evaluation, and a switching logic scheme (Fig. 8.3).

Each estimator reconstructs the plant output in either one of the healthy or faulty working modes. Its performance is evaluated by computing a norm of the output estimation error, and the estimator that yields the smallest performance index is assumed to correspond to the present working mode. The output of the switching logic η is the integer associated with that estimator, i.e. the estimator number. The corresponding controller is applied to the process using the switching logic.

This approach, however, presupposes that for each fault a reasonable controller has been designed before the plant is put into operation. From a practical point of view, this is not reasonable if a considerable number of faults has to be taken into account. To deal with this problem, two approaches are presented in this chapter, namely first methods that re-design the controller on-line after a fault has been identified to avoid

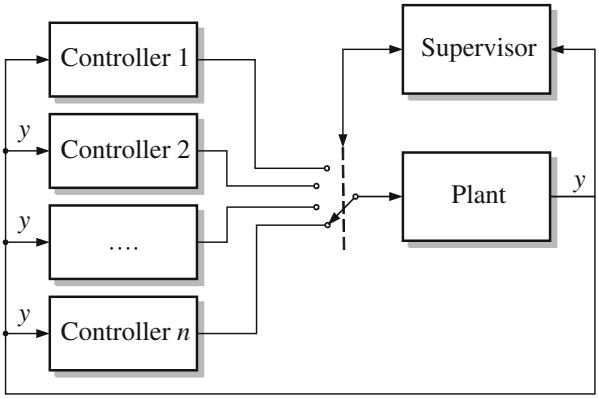


Fig. 8.2 Structure of logic-based switching controller

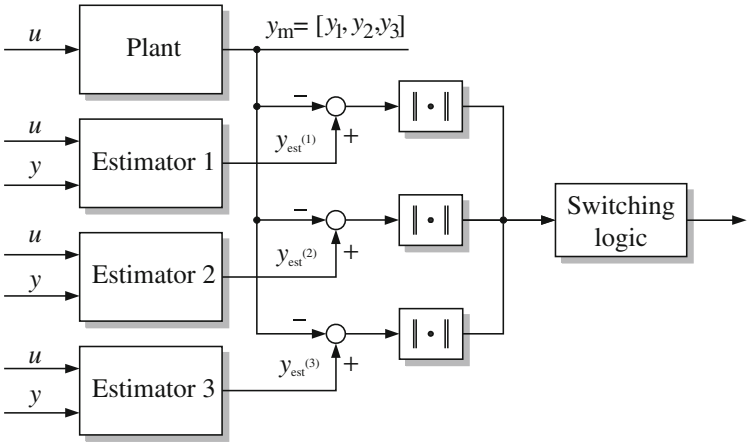


Fig. 8.3 Logic within a supervisor selects an output estimate from a bank of estimators

the use of a pre-determined bank of control laws, and second methods for reducing the size of pre-determined banks of control laws.

Note that a pre-determined bank of control laws needs possibly large memory space for their implementation but allow fast on-line reaction: once the fault has been isolated, the adequate control is just switched on, without any extra calculations. On the other hand, on-line re-design of the control law does not need extra storage but one has to wait for the design algorithm to be completed before the appropriate control law can be used.

In the sequel of this chapter, different approaches to the fault-tolerant control problem are presented, which refer to different objectives and faults. Although the presented approaches can also be used in different frames, this chapter builds on optimal control, actuator faults and the reconfiguration strategy. The linear quadratic

problem under actuator faults is considered in Sect. 8.3. Section 8.4 presents the lattice of actuator subsets whose properties are important since only the subset of healthy actuators controls the system under the reconfiguration strategy. Section 8.5 discusses the implementation problem associated with on-line re-design versus bank of control laws. The evaluation of fault tolerance is the subject of Sect. 8.6.

8.3 Fault-Tolerant Linear Quadratic Design

8.3.1 Control Problem

Linear quadratic (LQ) problems constitute a very popular frame for control design. In this section, the LQ problem is analysed with respect to the possible occurrence of actuator faults. It is shown that fault tolerance can only be achieved if admissible (rather than optimal) solutions exist. Conditions on an actuator fault to be possibly tolerated are given both for the fault accommodation and for the system reconfiguration strategies.

Consider the system whose nominal operation is modelled by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \sum_{i \in I} \boldsymbol{B}_i \, \boldsymbol{u}_i(t). \tag{8.1}$$

$\boldsymbol{x} \in \mathcal{X} \subset |\mathcal{R}^n$ is the state vector and $\boldsymbol{u} \in \mathcal{U} \subset |\mathcal{R}^m$ is the control vector. I is the set of the actuators, $\boldsymbol{u}_i(t) \in |\mathcal{R}^{m_i}$ is the input of actuator $i \in I$, and $m = \sum_{i \in I} m_i$. $\boldsymbol{A}$ and $\boldsymbol{B}$ are constant matrices of suitable dimensions, and it is assumed that the pair $(\boldsymbol{A}, \boldsymbol{B})$ is controllable. The following optimal control problem is considered:

Problem 8.4 (*Optimal control problem*)

1. **Objective** O**:** Transfer the system state from $\boldsymbol{x}(0) = \gamma$ towards $\boldsymbol{x}(\infty) = \boldsymbol{0}$, where $\gamma \in |\mathcal{R}^n$, and $\boldsymbol{x}(\infty)$ stands for $\lim_{t \longrightarrow \infty} \boldsymbol{x}(t)$ while minimising the functional

$$J(\boldsymbol{u}, \gamma) = \frac{1}{2} \int_0^\infty (\boldsymbol{u}^{\mathrm{T}}(t) \boldsymbol{R}\boldsymbol{u}(t) + \boldsymbol{x}^{\mathrm{T}}(t) \boldsymbol{Q}\boldsymbol{x}(t)) \, \mathrm{d}t, \tag{8.2}$$

   where $\boldsymbol{Q}$ and $\boldsymbol{R}$ are symmetric matrices, and $\boldsymbol{Q} \geq 0$, $\boldsymbol{R} > 0$.
2. **Constraints** C: Equation (8.1) is satisfied $\forall t \in [0, \infty)$, $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$ are continuous functions of time, and $\mathcal{X} = |\mathcal{R}^n, \mathcal{U} = |\mathcal{R}^m$ hold.

8.3.2 Control of the Nominal Plant

The solution of Problem 8.4 is well known from the classical theory of optimal control. Let $H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t)$ be the system Hamiltonian

$$H(\boldsymbol{x}(t),\ \boldsymbol{u}(t),\ \boldsymbol{p}(t),\ t) = -\frac{1}{2}\left(\boldsymbol{u}^{\mathrm{T}}(t)\boldsymbol{R}\boldsymbol{u}(t) + \boldsymbol{x}^{\mathrm{T}}(t)\boldsymbol{Q}\boldsymbol{x}(t)\right) + \boldsymbol{p}^{\mathrm{T}}(t)\left(\boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t)\right),$$

where $\boldsymbol{p}(t)$ is the adjoint state vector, then the necessary optimality condition is

$$\dot{\boldsymbol{x}}(t) = \frac{\partial H}{\partial \boldsymbol{p}}(\boldsymbol{x}(t),\ \boldsymbol{u}(t),\ \boldsymbol{p}(t),\ t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \tag{8.3}$$

$$\dot{\boldsymbol{p}}(t) = -\frac{\partial H}{\partial \boldsymbol{x}}(\boldsymbol{x}(t),\ \boldsymbol{u}(t),\ \boldsymbol{p}(t),\ t) = \boldsymbol{Q}\boldsymbol{x}(t) - \boldsymbol{A}^{\mathrm{T}}\boldsymbol{p}(t) \tag{8.4}$$

$$\boldsymbol{0} = \frac{\partial H}{\partial \boldsymbol{u}}(\boldsymbol{x}(t),\ \boldsymbol{u}(t),\ \boldsymbol{p}(t),\ t) = \boldsymbol{u}(t) - \boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{p}(t).$$

It is easy to show that the optimal solution is given by

$$\begin{aligned}\boldsymbol{p}(t) &= -\boldsymbol{P}\boldsymbol{x}(t)\\ \boldsymbol{u}(t) &= -\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}(t),\end{aligned}$$

where $\boldsymbol{P}$ is the (symmetric) solution of the algebraic Riccati equation

$$\boldsymbol{Q} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P} = \boldsymbol{0}$$

such that the closed-loop system

$$\dot{\boldsymbol{x}}(t) = \left(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\right)\boldsymbol{x}(t)$$

is stable. The solution exists since the pair $(\boldsymbol{A}, \boldsymbol{B})$ is controllable, and the optimal value of the criterion is given by

$$J(0,\ \emptyset,\ \gamma) = \frac{1}{2}\gamma^{\mathrm{T}}\boldsymbol{P}\gamma, \tag{8.5}$$

where the argument $0,\ \emptyset$ recalls that there is no faulty actuator on the time window $[0,\ \infty)$.

Nominal performances. Equation (8.5) shows that the nominal performance of the actuator set I depends on the value of γ.

$$\Gamma = \{\gamma \in |\mathcal{R}^n,\ \text{s.t.}\ \frac{1}{2}\gamma^{\mathrm{T}}\boldsymbol{P}\gamma \leq 1\}$$

represents the set of points in the state space from which the origin can be reached with a cost less than 1. The characterisation of the actuation scheme I independently of the control objective γ leads to consider the worst control problem from the quadratic criterion point of view: transfer the system state from $\boldsymbol{x}(0) = \gamma^*$ to $\boldsymbol{x}(\infty) = \boldsymbol{0}$,

where

$$\gamma^* = \arg\max_{|\gamma|=1} J\,(0, \emptyset, \gamma).$$

The set of actuators I is thus characterised by the maximum eigenvalue of $\boldsymbol{P}$ which is interpreted as the maximum cost which might be spent in transferring the system state from $\boldsymbol{x}(0) = \gamma$ to $\boldsymbol{x}(\infty) = \boldsymbol{0}$ for some $\gamma \in |\mathcal{R}^n$ such that $|\gamma| = 1$

$$J\left(0, \emptyset, \gamma^*\right) = \frac{1}{2}\lambda_{\max}(\boldsymbol{P}). \tag{8.6}$$

8.3.3 Fault Tolerance with Respect to Actuator Faults

This section considers the situation in which the system is faultless until the time instant t_f and has afterwards a fault in one or several actuators. Hence, the whole set of actuators I is healthy in the time interval $(0,\ t_\mathrm{f}[$ while there is a subset I_F of faulty actuators in the interval $[t_\mathrm{f},\ \infty)$. Let $I = I_\mathrm{N} \cup I_\mathrm{F}$, where I_N is the subset of the still normal actuators. After t_f the faulty system behaviour is described by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \sum_{i \in I_\mathrm{N}} \boldsymbol{B}_i\, \boldsymbol{u}_i\,(t) + \sum_{i \in I_\mathrm{F}} \boldsymbol{\beta}_i\ (\boldsymbol{u}_i(t),\ \theta_i), \tag{8.7}$$

where $\boldsymbol{\beta}_i\ (\boldsymbol{u}_i(t),\ \theta_i)$ describes the contribution of the faulty actuator i. This vector may be known, known with unknown parameters θ_i or completely unknown, depending on the faults which are considered, and of the capability of the fault diagnostic algorithm to estimate them. The objective, constraints and criterion of the fault-tolerant control problem are identical to those of the control problem, with the exception of constraint (8.1) being valid on $(0,\ t_\mathrm{f}[$ and being replaced by constraint (8.7) on $[t_\mathrm{f},\ \infty)$.

Problem constraints. Two cases can be considered as far as the status of constraint (8.7) is concerned.

1. In the first case, the fault tolerance analysis is done (off-line) for given faults, which are known to possibly occur in the considered system (from the failure-modes and effect analysis, for example). Therefore, constraint (8.7) *is known* and the fault-tolerant control can be designed beforehand (but it can be applied on-line only when the actual fault matches the fault for which it has been designed, which needs the actual fault to be identified).
2. In the second case, the analysis is done for any kind of fault which might occur during the system operation and, therefore, constraint (8.7) *being not known* has again to be identified (or replaced by another constraint if identification is impossible or not available). The identification of the subset I_F of faulty actuators is normally done by the fault diagnostic algorithm, which detects and isolates the faults. Defining the constraints resumes to identifying the functions

$\boldsymbol{\beta}_i(\boldsymbol{u}_i(t), \theta_i), i \in I_F$. This is not usually done by fault diagnostic algorithms, and could be referred to as a *diagnostic* (or fault estimation) possibility, which rests on fault modelling and on fault parameter identification, and it could be—or not—provided by the fault diagnostic system.

Therefore, the two approaches to fault-tolerant control can be applied in dependence upon the situation. Fault accommodation consists of controlling the faulty system after replacing Eq. (8.7) by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \sum_{i \in I_N} \boldsymbol{B}_i \, \boldsymbol{u}_i(t) + \sum_{i \in I_F} \hat{\boldsymbol{\beta}}_i \left(\boldsymbol{u}_i(t), \, \hat{\boldsymbol{\theta}}_i \right), \tag{8.8}$$

where the functions $\hat{\boldsymbol{\beta}}_i \, (\boldsymbol{u}_i(t), \, \hat{\boldsymbol{\theta}}_i)$ and parameters $\hat{\boldsymbol{\theta}}_i, i \in I_F$ are estimated. System reconfiguration consists of controlling only the healthy part of the system (thus switching off the faulty actuators), which means replacing Eq. (8.7) by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \sum_{i \in I_N} \boldsymbol{B}_i \, \boldsymbol{u}_i(t). \tag{8.9}$$

Admissible solutions. Whatever the selected strategy, a solution to the fault-tolerant control problem exists provided the objective $\boldsymbol{x}(\infty) = \boldsymbol{0}$ can still be reached from the initial state $\boldsymbol{x}(0) = \boldsymbol{\gamma}$. When solutions are needed to exist for any objective $\boldsymbol{\gamma}$, it is obviously necessary that the system (8.7) or (8.9) is still controllable.

Suppose that the fault-tolerant control problem has a solution, i.e. the system state can be transferred from $\boldsymbol{x}(0) = \boldsymbol{\gamma}$ to $\boldsymbol{x}(\infty) = \boldsymbol{0}$, and introduce the notation

$$J \, ((0, \, t_f), \, (\emptyset, \, I_F), \, \boldsymbol{\gamma}) \tag{8.10}$$

for the minimal cost associated with the two time instants $(0, \, t_f)$ at which the failed actuators are respectively $(\emptyset, \, I_F)$. Obviously, the fact that a solution exists does not mean that it is satisfactory. Two cases can be distinguished.

- The cost is of no importance provided the system objective is achieved in spite of the fault. In this case, the actuation scheme I is fault tolerant with respect to the situation I_F occurring at time t_f if and only if system (8.8)—when accommodation is used—or (8.9)—when reconfiguration is concerned—is controllable.
- Some cost limitation is considered. Although optimal, the cost might be too high, thus denying the actuation scheme I to deserve the "fault-tolerant" label with respect to the situation I_F.

Definition 8.1 (*Admissibility*) Let I_F be a fault situation occurring at time t_f. The solution of the fault-tolerant control problem is admissible with respect to the control objective $\boldsymbol{\gamma}$ if and only if

$$J\left((0,\ t_{\mathrm{f}}),\ (\emptyset,\ I_{\mathrm{F}}),\ \gamma\right) \leq \rho(\gamma) J\left(0,\ \emptyset,\ \gamma\right), \tag{8.11}$$

where $\rho(\gamma) \geq 1$ is some pre-defined function.

In Eq. (8.11), $\rho(\gamma)$ is the maximal loss of efficiency which is allowed when a control solution, which still achieves the objective γ but under the situation where the fault I_{F} occurs at time t_{f}, is used. Three special choices of $\rho(\gamma)$ may be of interest.

- $\rho(\gamma) = \infty,\ \forall \gamma \in |\mathcal{R}^n$.
  In this case, fault tolerance is only concerned with the existence of an optimal solution, whatever its cost, thus reducing the fault-tolerance property to the permanence of the controllability property: any fault such that the system remains controllable is recoverable,
- $\rho(\gamma) = \frac{\sigma}{\gamma^{\mathrm{T}} \boldsymbol{P}_\gamma},\ \forall \gamma \in |\mathcal{R}^n,\ |\gamma| \leq 1$
  defines a uniform bound σ for the cost of controlling the faulty system, whatever the initial state in the unit sphere: any fault such that there exists a stabilising control whose cost is less than σ is recoverable,
- $\rho(\gamma) = \rho^*,\ \forall \gamma \in |\mathcal{R}^n$
  defines a uniform bound for the loss of efficiency in the control of the faulty system, whatever the control objective: any fault such that there exists a stabilising control associated with a cost degradation factor less than ρ^* is recoverable.

Based on the definition of admissibility, fault tolerance can be defined as follows.

Definition 8.2 (*Fault tolerance of a system subject to actuator faults*) The actuation scheme I is fault tolerant with respect to the fault I_{F} occurring at time t_{f} for the control objective γ if the accommodation or the reconfiguration problem has an admissible solution (equivalently, fault I_{F} occurring at time t_{f} is said to be recoverable).

8.3.4 Fault Accommodation

The accommodation strategy is now analysed for the system described by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \sum_{i \in I} \boldsymbol{B}_i\, \boldsymbol{u}_i(t) \quad \text{for } t \in [0,\ t_{\mathrm{f}}[$$

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \sum_{i \in I_{\mathrm{N}}} \boldsymbol{B}_i\, \boldsymbol{u}_i(t) + \sum_{i \in I_{\mathrm{F}}} \boldsymbol{\beta}_i\ (\boldsymbol{u}_i(t),\ \theta_i), \quad \boldsymbol{x}(t_{\mathrm{f}}) = \boldsymbol{x}_{\mathrm{f}},$$

$$\text{for } t \in [t_{\mathrm{f}},\ \infty).$$

Identifying the faulty system. Since the functions $\boldsymbol{\beta}_i\ (\boldsymbol{u}_i(t),\ \theta_i)$ and parameters θ_i, $i \in I_{\mathrm{F}}$ are not known, they must be estimated, and therefore the LQ control problem is set for the model

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \sum_{i \in I} \boldsymbol{B}_i \, \boldsymbol{u}_i(t) \quad \text{for } t \in [0, \, t_\mathrm{f}[$$

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \sum_{i \in I_\mathrm{N}} \boldsymbol{B}_i \, \boldsymbol{u}_i(t) + \sum_{i \in I_\mathrm{F}} \hat{\beta}_i \, \left(\boldsymbol{u}_i(t), \, \hat{\theta}_i\right) \quad \text{for } t \in [t_\mathrm{f}, \, \infty),$$

where the functions $\hat{\beta}_i \, \left(\boldsymbol{u}_i(t), \, \hat{\boldsymbol{\theta}}_i\right)$ and parameters $\hat{\boldsymbol{\theta}}_i$, $i \in I_\mathrm{F}$ are known. This approach obviously needs some fault model to be defined, and its parameters to be identified.

Assume it is known that the faulty actuators can still be described by a linear model

$$\hat{\beta}_i \, \left(\boldsymbol{u}_i(t), \, \hat{\boldsymbol{\theta}}_i\right) = \hat{\boldsymbol{B}}_i \, \boldsymbol{u}_i(t), \, i \in I_\mathrm{F}$$

and, therefore, the model of the faulty system is

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \sum_{i \in I_\mathrm{N}} \boldsymbol{B}_i \boldsymbol{u}_i(t) + \sum_{j \in I_\mathrm{F}} \hat{\boldsymbol{B}}_j \boldsymbol{u}_j(t) \\ &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}_\mathrm{f} \boldsymbol{u}(t), \end{aligned} \tag{8.12}$$

where $\boldsymbol{B}_\mathrm{f} = (\boldsymbol{B}_\mathrm{N}, \hat{\boldsymbol{B}}_\mathrm{F})$ is the new actuator matrix, formed by the concatenation of the $\boldsymbol{B}_i$ matrices associated with the healthy actuators $\boldsymbol{B}_\mathrm{N} = (\boldsymbol{B}_i, i \in I_\mathrm{N})$ and the $\hat{\boldsymbol{B}}_j$ matrices associated with the faulty actuators $\hat{\boldsymbol{B}}_\mathrm{F} = (\hat{\boldsymbol{B}}_j, j \in I_\mathrm{F})$.

Accomodating the control to the faulty system. From Bellman's optimality principle, the accommodation strategy consists of applying the optimal control solution to system (8.12), with initial condition $\boldsymbol{x}_\mathrm{f} = \boldsymbol{x}(t_\mathrm{f})$, on the time interval $[t_\mathrm{f}, \, \infty)$, thus leading to compute the accommodated control and trajectories as the solution of

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}_\mathrm{f} \boldsymbol{u}(t) \\ \dot{\boldsymbol{p}}(t) &= \boldsymbol{Q}\boldsymbol{x}(t) - \boldsymbol{A}^\mathrm{T} \boldsymbol{p}(t) \\ \boldsymbol{u}(t) &= \boldsymbol{R}^{-1} \, \boldsymbol{B}_\mathrm{f}^\mathrm{T} \boldsymbol{P} \boldsymbol{x}(t) \end{aligned} \tag{8.13}$$

with the result that the value of the criterion is now

$$J\left((0, \, t_\mathrm{f}), \, (\emptyset, \, I_\mathrm{F}), \, \boldsymbol{\gamma}\right)\right) = J_{0f} + \frac{1}{2} \boldsymbol{x}_\mathrm{f}^\mathrm{T} \boldsymbol{P}_\mathrm{f} \boldsymbol{x}_\mathrm{f} \tag{8.14}$$

instead of

$$J(0, \, \emptyset, \, \boldsymbol{\gamma}) = \frac{1}{2} \boldsymbol{\gamma}^\mathrm{T} \boldsymbol{P} \boldsymbol{\gamma},$$

where J_{0f} is the cost already spent between $t = 0$ and $t = t_\mathrm{f}$ and $\boldsymbol{P}_\mathrm{f}$ is the solution of the algebraic Riccati equation in which $\boldsymbol{B}$ has been replaced by $\boldsymbol{B}_\mathrm{f}$, namely

$$\boldsymbol{Q} + \boldsymbol{A}^\mathrm{T} \boldsymbol{P}_\mathrm{f} + \boldsymbol{P}_\mathrm{f} \, \boldsymbol{A} - \boldsymbol{P}_\mathrm{f} \, \boldsymbol{B}_\mathrm{f} \, \boldsymbol{R}^{-1} \, \boldsymbol{B}_\mathrm{f}^\mathrm{T} \, \boldsymbol{P}_\mathrm{f} = \boldsymbol{0}. \tag{8.15}$$

Testing the admissibility of the accommodated control. From simple calculations, and taking into account that

$$J(0,\ \emptyset,\ \gamma) = J_{0f} + \frac{1}{2}\boldsymbol{x}_{\mathrm{f}}^{\mathrm{T}}\,\boldsymbol{P}\boldsymbol{x}_{\mathrm{f}}$$

one has

$$J_{0f} = \frac{1}{2}\gamma^{\mathrm{T}}\boldsymbol{P}\gamma - \frac{1}{2}\boldsymbol{x}_{\mathrm{f}}^{\mathrm{T}}\,\boldsymbol{P}\boldsymbol{x}_{\mathrm{f}}$$

and therefore

$$J\left((0,\ t_{\mathrm{f}}),\ (\emptyset,\ I_{\mathrm{F}}),\ \gamma\right)) = \frac{1}{2}\gamma^{\mathrm{T}}\boldsymbol{P}\gamma + \frac{1}{2}\boldsymbol{x}_{\mathrm{f}}^{\mathrm{T}}\,\left(\boldsymbol{P}_{\mathrm{f}} - \boldsymbol{P}\right)\boldsymbol{x}_{\mathrm{f}}. \tag{8.16}$$

From (8.14) and the different definitions of admissibility, the set of triples

$$(\boldsymbol{B}_{\mathrm{f}}, t_{\mathrm{f}}, \gamma)$$

which can be tolerated by an accommodation strategy are characterised as follows:

- $\rho(\gamma) = \infty,\ \forall\gamma \in |\mathcal{R}^n$

$$(\boldsymbol{A},\ \boldsymbol{B}_{\mathrm{f}})\ \text{controllable} \tag{8.17}$$

- $\rho(\gamma) = \frac{\sigma}{\gamma^{\mathrm{T}}\boldsymbol{P}\gamma},\ \forall\gamma \in |\mathcal{R}^n,\ |\gamma| \leq 1$

$$\begin{array}{c}(\boldsymbol{A},\ \boldsymbol{B}_{\mathrm{f}})\ \text{controllable}\\ \boldsymbol{x}_{\mathrm{f}}^{\mathrm{T}}\left(\boldsymbol{P}_{\mathrm{f}} - \boldsymbol{P}\right)\boldsymbol{x}_{\mathrm{f}} \leq \sigma - \gamma^{\mathrm{T}}\boldsymbol{P}\gamma\end{array} \tag{8.18}$$

- $\rho(\gamma) = \rho^*,\ \forall\gamma \in |\mathcal{R}^n$

$$\begin{array}{c}(\boldsymbol{A},\ \boldsymbol{B}_{\mathrm{f}})\ \text{controllable}\\ \boldsymbol{x}_{\mathrm{f}}^{\mathrm{T}}\left(\boldsymbol{P}_{\mathrm{f}} - \boldsymbol{P}\right)\boldsymbol{x}_{\mathrm{f}} \leq (\rho^* - 1)\,\gamma^{\mathrm{T}}K\gamma\end{array} \tag{8.19}$$

Note that these conditions depend on the value of the state $\boldsymbol{x}_{\mathrm{f}}$ at the time of the fault occurrence, which is computed by

$$\boldsymbol{x}_{\mathrm{f}}(t) = \mathrm{e}^{\,\boldsymbol{A}t_{\mathrm{f}}}\,\gamma + \int_0^{t_{\mathrm{f}}} e^{\boldsymbol{A}(t_{\mathrm{f}}-t)}\,\boldsymbol{B}\boldsymbol{u}(t)\,\mathrm{d}t,$$

where $\boldsymbol{u}(t)$ is the optimal control computed from (8.4), and can also be expressed as

$$\boldsymbol{x}_{\mathrm{f}} = \mathrm{e}^{\left(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{R}^{-1}\,\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\right)t_{\mathrm{f}}}\gamma.$$

Since t_{f} is unknown beforehand, these conditions can only be checked on-line, at time t_{f} when the fault is detected, isolated and diagnosed. Of course, it might be

unpleasant to discover on-line that the fault that just occurred cannot be accommodated. Therefore, it is interesting to look for sufficient conditions, which could be checked off-line. Such conditions can be found under the reasonable assumption that if the objective can be reached by an admissible control using the faulty system from the beginning, then it can also be reached by an admissible control when the nominal system is first used and replaced (at an unknown time) by the faulty one.

This assumption is satisfied as it can be seen by considering the worst case value of $\boldsymbol{x}_{\mathrm{f}}$ in the previous conditions. Under the assumption that $(\boldsymbol{P}_{\mathrm{f}} - \boldsymbol{P}) \geq 0$ (which is reasonable since it states that the faulty actuators are less efficient than the healthy ones), the worst case situation is that in which the fault occurs right at time $t_{\mathrm{f}} = 0$, and therefore one has $\boldsymbol{x}_{\mathrm{f}} = \boldsymbol{\gamma}$, which leads to the sufficient conditions (8.20)–(8.22) for the fault I_{F} to be tolerated using an accommodation strategy. Note that these conditions characterise all the pairs $(\boldsymbol{B}_{\mathrm{f}}, \boldsymbol{\gamma})$ for which the system is fault tolerant, whatever the time at which the fault $\boldsymbol{B}_{\mathrm{f}}$ occurs.

- $\rho(\boldsymbol{\gamma}) = \infty, \forall \boldsymbol{\gamma} \in |\mathcal{R}^n$

$$(\boldsymbol{A},\ \boldsymbol{B}_{\mathrm{f}})\ \text{controllable} \tag{8.20}$$

- $\rho(\boldsymbol{\gamma}) = \frac{\sigma}{\boldsymbol{\gamma}^{\mathrm{T}} \boldsymbol{P} \boldsymbol{\gamma}},\ \forall \boldsymbol{\gamma} \in |\mathcal{R}^n,\ |\boldsymbol{\gamma}| \leq 1$

$$\begin{array}{c} (\boldsymbol{A},\ \boldsymbol{B}_{\mathrm{f}})\ \text{controllable} \\ \boldsymbol{\gamma}^{\mathrm{T}} \boldsymbol{P}_{\mathrm{f}} \boldsymbol{\gamma} \leq \sigma \end{array} \tag{8.21}$$

- $\rho(\boldsymbol{\gamma}) = \rho^*,\ \forall \boldsymbol{\gamma} \in |\mathcal{R}^n$

$$\begin{array}{c} (\boldsymbol{A},\ \boldsymbol{B}_{\mathrm{f}})\ \text{controllable} \\ \boldsymbol{\gamma}^{\mathrm{T}} (\boldsymbol{P}_{\mathrm{f}} - \rho^* \cdot \boldsymbol{P}) \boldsymbol{\gamma} \leq 0 \end{array} \tag{8.22}$$

The conditions under which the fault $\boldsymbol{B}_{\mathrm{f}}$ can be tolerated for any objective $\boldsymbol{\gamma}$, whatever the time at which it occurs, are given by

- $\rho(\boldsymbol{\gamma}) = \infty,\ \forall \boldsymbol{\gamma} \in |\mathcal{R}^n$

$$(\boldsymbol{A},\ \boldsymbol{B}_{\mathrm{f}})\ \text{controllable}, \tag{8.23}$$

- $\rho(\boldsymbol{\gamma}) = \frac{\sigma}{\boldsymbol{\gamma}^{\mathrm{T}} \boldsymbol{P} \boldsymbol{\gamma}},\ \forall \boldsymbol{\gamma} \in |\mathcal{R}^n,\ |\boldsymbol{\gamma}| \leq 1$

$$\begin{array}{c} (\boldsymbol{A},\ \boldsymbol{B}_{\mathrm{f}})\ \text{controllable} \\ \lambda_{\max}\,(\boldsymbol{P}_{\mathrm{f}}) \leq \sigma, \end{array} \tag{8.24}$$

- $\rho(\boldsymbol{\gamma}) = \rho^*,\ \forall \boldsymbol{\gamma} \in |\mathcal{R}^n$

$$\begin{array}{c} (\boldsymbol{A},\ \boldsymbol{B}_{\mathrm{f}})\ \text{controllable} \\ \lambda_{\max}\,(\boldsymbol{P}_{\mathrm{f}} - \rho^* \cdot \boldsymbol{P}) \leq 0. \end{array} \tag{8.25}$$

8.3.5 Control Reconfiguration

Reconfiguration strategies set the control problem of a system in which the faulty part has been switched off. The choice of a reconfiguration strategy might follow from the impossibility of estimating the fault, or it can be deliberate, so as to implement fault-tolerant strategies which provide guaranteed results, and are as simple and as understandable as possible by operators. In many cases, reconfiguration is understood as the replacement of the faulty part by some non-faulty one. Considering the problem under investigation, this means that some actuators were not in service before the fault occurrence and that they can be switched on after the fault.

Let I_{off} be the set of those actuators, which are assumed without loss of generality to be non-faulty. It obviously follows that considering from the beginning the whole set of actuators $I \cup I_{\text{off}}$ reduces the problem to that of reconfiguring the system $I_{\text{off}} \cup I_{\text{N}} \cup I_{\text{F}}$ by simply removing the faulty part. Thus, including I_{off} within I (namely, into I_{N}), one can go on with unchanged notations. In this situation, the fault-tolerant control problem has to be analysed replacing Eq. (8.7) by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \sum_{i \in I_{\text{N}}} \boldsymbol{B}_i \, \boldsymbol{u}_i(t).$$

Therefore, all the previous results and statements also apply to the reconfiguration strategy, provided $\boldsymbol{B}_{\text{f}} = \left(\boldsymbol{B}_{\text{N}},\ \hat{\boldsymbol{B}}_{\text{F}}\right)$ is replaced by $\boldsymbol{B}_{\text{f}} = (\boldsymbol{B}_{\text{N}},\ \boldsymbol{O})$.

It is easily seen that $\boldsymbol{B}_{\text{f}}$ only depends on the subset of actuators I_{N} whatever the faults that act on the subset of actuators I_{F}. Therefore, neither is it needed to assume that the faulty actuators be described by a linear model $\hat{\boldsymbol{B}}_{\text{f}}$, nor is it needed to identify this model. Moreover, since there is only a finite number of actuator subsets, the reconfigured controls can be computed off-line for each possible subset I_{F}, and the solution can be switched on-line as soon as the FDI has provided the current subset of faulty actuators (this needs only fault detection and isolation). Note that for some subsets I_{F} an admissible solution will not exist, therefore it is of interest to analyse off-line all the possible subsets of faulty actuators.

8.4 The Lattice of Actuator Subsets

The accommodation and reconfiguration strategies have been presented in the previous section for the case of actuator faults in the Linear Quadratic problem. However, whatever the control objectives, the reconfiguration strategy always deals with controlling only the subset of the system's healthy components (the faulty ones are switched off) and therefore, the analysis of the system's component subsets is the general frame in which the reconfiguration problem is to be considered. This is the goal of this section.

8.4.1 Actuator Configurations

Since I is the set of all actuators in the system, the power set 2^I is the set of all possible actuator subsets, also named *actuator configurations*. According to the fact that the pair $(\boldsymbol{A},\ \boldsymbol{B}_{\mathrm{f}})$ associated with a given configuration I_{N} satisfies or not the admissibility conditions, 2^I can be partitioned into

$$2^I = \mathcal{R} \cup \overline{\mathcal{R}}$$

where

$$\mathcal{R} = \{I_{\mathrm{N}} \subseteq I :\ \text{the fault } I_{\mathrm{F}} = I \backslash I_{\mathrm{N}} \text{ can be tolerated}\}$$
$$\overline{\mathcal{R}} = \{J \subset I :\ \text{the fault } I_{\mathrm{F}} = I \backslash I_{\mathrm{N}} \text{ cannot be tolerated}\}.$$

Definition 8.3 (*Recoverable fault, recoverable configuration*) A fault I_{F} is said to be recoverable if configuration $I_{\mathrm{N}} \in \mathcal{R}$. It is non-recoverable if configuration $I_{\mathrm{N}} \in \overline{\mathcal{R}}$. In the sequel we also use the wording *recoverable/non-recoverable configuration*.

It is well known that power sets have a lattice structure. That means that 2^I can be represented by a hierarchical graph, where nodes are actuator configurations organised into levels as follows:

- level 0 contains only I,
- level 1 contains all configurations I_{N} such that I_{F} has only one element,
- level 2 contains all configurations I_{N} such that I_{F} has two elements,
- etc. ...
- the last level is the empty set (I_{F} contains all actuators I).

Each configuration at a given level belongs either to $\mathcal{R}$ or to $\overline{\mathcal{R}}$. Edges connect configurations which belong to adjacent levels and differ by only one actuator.

Definition 8.4 (*Successors, predecessors*) Let I_{N} be a configuration, $\mathcal{S}\,(I_{\mathrm{N}})$ the set of its successors and $\mathcal{P}\,(I_{\mathrm{N}})$ the set of its predecessors are defined as

$$\mathcal{S}(I_{\mathrm{N}}) = \{I' \in 2^I :\ I' \subseteq I_{\mathrm{N}}\}$$
$$\mathcal{P}(I_{\mathrm{N}}) = \{I'' \in 2^I :\ I'' \supseteq I_{\mathrm{N}}\}.$$

Note that from this definition, any configuration I_{N} belongs both to $\mathcal{S}(I_{\mathrm{N}})$ and $\mathcal{P}(I_{\mathrm{N}})$. Remark also that since a successor of I_{N} is included in I_{N} it represents a configuration with more faulty actuators while a predecessor of I_{N} represents a configuration with less faulty actuators.

Example 8.3 Reconfiguration after actuator faults
Consider a system with 7 states, and 4 actuators: $I = \{1,\ 2,\ 3,\ 4\}$. The matrices $\boldsymbol{A}$ and $\boldsymbol{B}^{\mathrm{T}}$ are as follows:

$$\boldsymbol{A} = \text{diag}\ \{-1,\ -0.5,\ -3,\ -4,\ -2,\ -1.5,\ -2.5\},$$
$$\boldsymbol{B}^{\mathrm{T}} = \begin{pmatrix} 0\ 1\ 1\ 1\ 1\ 1\ 0 \\ 1\ 0\ 1\ 1\ 1\ 0\ 1 \\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\ 1\ 0\ 1\ 0\ 1\ 1\ 1 \end{pmatrix}.$$

The considered criterion is

$$J(\boldsymbol{u},\ \gamma) = \frac{1}{2}\int_0^{\infty} \boldsymbol{u}^{\mathrm{T}}(t)\boldsymbol{u}(t)\,\mathrm{d}t,$$

which means that only the control energy is of interest and $\boldsymbol{R}$ is the identity matrix. In that case, it is known that

$$J(I,\ 0,\ \gamma) = \gamma^{\mathrm{T}}\boldsymbol{W}_{\mathrm{c}}^{-1}\gamma,$$

where $\boldsymbol{W}_{\mathrm{c}}$ is the Gramian associated with the pair $(\boldsymbol{A},\ \boldsymbol{B})$, i.e.

$$\boldsymbol{W}_{\mathrm{c}} = \int_0^{\infty} \mathrm{e}^{\boldsymbol{A}t}\,\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}\left(\mathrm{e}^{\boldsymbol{A}t}\right)^{\mathrm{T}}\,\mathrm{d}t.$$

The maximal eigenvalue is $\lambda_{\max}\left(\boldsymbol{W}_{\mathrm{c}}^{-1}\,(I)\right) = 0.4357$ energy units.

Assume that admissible solutions are defined such that the worst situation control cost should not exceed 1.125 energy units. Then, there are 10 fault situations in which the system is controllable by reconfiguration, namely when only actuators 1234, 234, 134, 124, 123, 34, 23, 14, 12, 13 remain available (using the short notations 1234 for {1, 2, 3, 4}, 24 for {2, 4}, etc.) but only 6 of them are admissible when energy limitation is considered, as shown by Table 8.1.

Table 8.1 Admissible actuators subsets and associated characteristics

Actuator subsets	$\lambda_{\max}$ (energy units)
1234	0.4357
234	1.1197
134	0.4676
124	0.8274
123	0.4778
34	**3.0201**
23	**1.3948**
14	**2.2576**
12	1.0612
13	**1.1452**

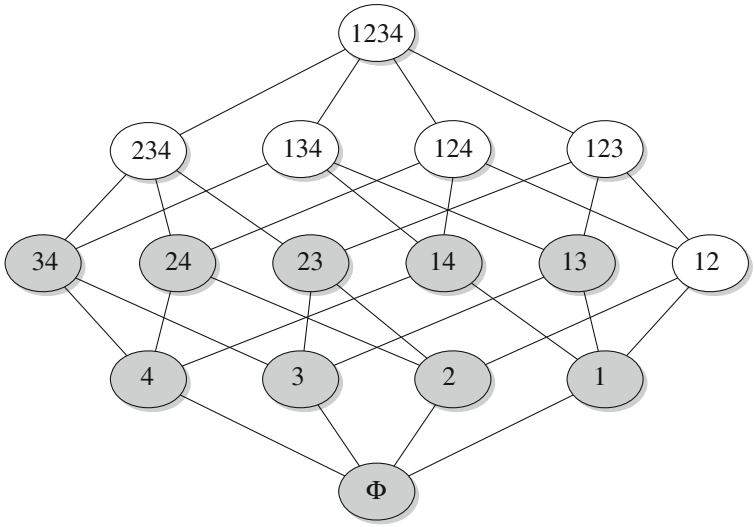


Fig. 8.4 The lattice of the actuators subsets in the example

Figure 8.4 shows the actuators lattice and its five levels. Dark grey nodes are configurations that cannot control the system (the corresponding pair $(\boldsymbol{A},\ \boldsymbol{B}_{\mathrm{f}})$ is not controllable), light grey nodes are configurations by which the system is controllable, but energy limitations are not met, white nodes are configurations which allow to control the system in an admissible way, i.e. they are recoverable. Grey nodes correspond to faults that cannot be tolerated, i.e. they represent non-recoverable configurations. □

Discrete state behaviour of the actuation system. Define the discrete state of the actuation system as the subset of actuators $I_{\mathrm{N}}(t)$, that are healthy at time t and assume, without loss of generality, that $I_{\mathrm{N}}(0) = I$. Assume that at time t_1 actuator σ_1 becomes faulty, then the reconfiguration mechanism, by discarding actuator σ_1, results in the discrete state $I_{\mathrm{N}}(t_1) = I \setminus \{\sigma_1\}$ which belongs to $\mathcal{S}(I)$. Further faults will result in discrete states $I_{\mathrm{N}}(t)$ moving to lower levels in the lattice, according to the dynamical dicrete state equation

$$I_{\mathrm{N}}(t^+) = I_{\mathrm{N}}(t^-) \setminus \Sigma_{\mathrm{f}}(t),$$

where $I_{\mathrm{N}}(t^-)$ is the discrete state before the fault, $\Sigma_{\mathrm{f}}(t) \subseteq I_{\mathrm{N}}(t^-)$ is the subset of actuators that become faulty at time t, and $I_{\mathrm{N}}(t^+)$ is the discrete state after the system reconfiguration. Symmetrically, repair operations move the discrete state to higher levels, according to

$$I_{\mathrm{N}}(t^{+}) = I_{\mathrm{N}}(t^{-}) \cup \Sigma_{\mathrm{r}}(t),$$

where $\Sigma_r(t) \subseteq I \setminus I_{\mathrm{N}}(t^{-})$ is the subset of actuators that have been repaired at time t (it can be checked that subsets of faulty or repaired actuators can be treated one by one, in an arbitrary order, resulting in the same post-fault or post-maintenance configuration in the lattice). Repair operations have an important impact on system reliability (by means of fault avoidance), but they are not considered here.

8.4.2 Critical Actuator Subsets and Minimal Recoverable Configurations

Consider a recoverable configuration $I_{\mathrm{N}} \in \mathcal{R}$. Loosing a subset of actuators $\Sigma \subset I_{\mathrm{N}}$ can be tolerated as long as the resulting configuration $I_{\mathrm{N}} \setminus \Sigma$ is still recoverable.

Definition 8.5 (*Critical actuator subsets*) A critical actuator subset associated with the recoverable configuration I_{N} is a minimal subset $\Sigma \subset I_{\mathrm{N}}$ such that $I_{\mathrm{N}} \setminus \Sigma \in \overline{\mathcal{R}}$.

Critical subsets are in general not unique. Let $C(I_{\mathrm{N}})$ be the ones associated with configuration I_{N}. Note that minimality is required in the definition because the loss of any superset of a critical actuator subset could obviously not be tolerated.

Definition 8.6 (*Minimal recoverable configuration*) A minimal recoverable configuration is a configuration that belongs to $\mathcal{R}$ while all its successors belong to $\overline{\mathcal{R}}$.

This is a very interesting property: in spite of those actuators already switched off, a minimal recoverable configuration is indeed recoverable, and so are all its predecessors, but loosing any extra actuator results in a non-recoverable configuration. As a result, the set of all recoverable configurations is completely known once the minimal recoverable ones have been found. Note also that the critical actuator subsets associated with a minimal recoverable configuration are the singletons formed with each actuator in the configuration.

Example 8.4 Critical subsets, Minimal recoverable configurations
It is easily seen on Fig. 8.4 that the set of recoverable configurations is

$$\mathcal{R} = \{1234,\ 123,\ 124,\ 134,\ 234,\ 12\}$$

The minimal recoverable configurations are therefore {12, 134, 234}. Indeed, loosing one more actuator in any of these configurations moves the system to a non-recoverable configuration, whatever the lost actuator. Note that any subset of a non-recoverable configuration is non-recoverable, while any superset of a recoverable configuration is recoverable. The critical actuator subsets associated with the nominal configuration 1234 are $C(1234) = \{24,\ 23,\ 14,\ 13,\ 12\}$, while the critical actuator subsets associated with configuration 234 are $C(234) = \{2,\ 3,\ 4\}$, which is not a surprise since 234 is a minimal recoverable configuration. □

8.5 Implementational Issues of Fault-Tolerant Control

8.5.1 On-Line Re-design Versus Bank of Control Laws

In the Fault-Tolerant Linear Quadratic problem, on-line re-design computes the control law $\boldsymbol{u}(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{P}_{\mathrm{f}}\boldsymbol{x}(t)$ adapted to the faulty system by solving the Riccati equation

$$\boldsymbol{Q} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{P}_{\mathrm{f}} + \boldsymbol{P}_{\mathrm{f}}\boldsymbol{A} - \boldsymbol{P}_{\mathrm{f}}\boldsymbol{B}_{\mathrm{f}}\boldsymbol{R}^{-1}\boldsymbol{B}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{P}_{\mathrm{f}} = \boldsymbol{O}$$

where the post-fault actuation matrix $\boldsymbol{B}_{\mathrm{f}}$ is known from a fault estimation procedure when accommodation is applied or from zeroing the columns associated with the faulty actuators when reconfiguration is used. Note that a solution exists under the condition that the fault is recoverable, but the delay between the occurrence of the fault and the availability of the re-designed control law may lead to possibly unpleasant transient behaviours, during the time when the faulty system is still controlled by the nominal control law (an approach to this problem will be presented in Chap. 9). While on-line re-design is compulsory in fault accommodation because $\boldsymbol{B}_{\mathrm{f}}$ is not known in advance, it is optional in system reconfiguration. Indeed, since there is a limited number of possible $\boldsymbol{B}_{\mathrm{f}}$ matrices (each one is associated with an actuator configuration), the control law associated with each of them can be designed off-line, and stored in a control bank from which the appropriate one is selected as soon as the faulty actuators have been isolated, i.e. the current configuration is known. The control bank contains as many control laws as the number of recoverable configurations, which may be unpractical if this number is large. A solution to this problem, the so-called Passive–Active approach, is presented now.

8.5.2 The Passive–Active Approach

The Passive–Active (PACT) approach is intended to decrease the number of control laws that allow to recover all the recoverable faults. It is based on a result known as the "Reliable Control Theorem".

Theorem 8.1 (Reliable Control) *Consider the Linear Quadratic problem associated with the nominal system*

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \quad \boldsymbol{x}(0) = \boldsymbol{\gamma}$$
$$J(\boldsymbol{u}, \boldsymbol{\gamma}) = \frac{1}{2}\int_0^{\infty} \left[\boldsymbol{x}^{\mathrm{T}}(t)\,\boldsymbol{Q}\boldsymbol{x}(t) + \boldsymbol{u}^{\mathrm{T}}(t)\boldsymbol{R}\boldsymbol{u}(t)\right] \mathrm{d}t$$

where matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{R}$, $\boldsymbol{Q}$ *are given (*$\boldsymbol{R}$ *is assumed to be diagonal) and such that the Riccati equation*

$$\boldsymbol{Q} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P} = \boldsymbol{O}$$

has a unique positive definite stabilising solution. The optimal control law is therefore $\boldsymbol{u}(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}(t)$. *Let* $\{I_{\mathrm{N}}, I_{\mathrm{F}}\}$ *be a partition of the set of actuators* I *and let* $\boldsymbol{B} = \boldsymbol{B}_{\mathrm{N}} + \boldsymbol{B}_{\mathrm{F}}$ *where* $\boldsymbol{B}_{\mathrm{N}}$ *(resp.* $\boldsymbol{B}_{\mathrm{F}}$*) is obtained by zeroing those columns in* $\boldsymbol{B}$ *that are associated with the actuators in* I_{F} *(resp.* I_{N}*). Assume that the Riccati equation*

$$\boldsymbol{Q} + \boldsymbol{A}^{\mathrm{T}}\boldsymbol{P}_{\mathrm{N}} + \boldsymbol{P}_{\mathrm{N}}\boldsymbol{A} - \boldsymbol{P}_{\mathrm{N}}\boldsymbol{B}_{\mathrm{N}}\boldsymbol{R}^{-1}\boldsymbol{B}_{\mathrm{N}}^{\mathrm{T}}\boldsymbol{P}_{\mathrm{N}} = \boldsymbol{O} \tag{8.26}$$

has a unique definite positive stabilising solution, then the control law $\boldsymbol{u}_{\mathrm{N}}(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}_{\mathrm{N}}\boldsymbol{x}(t)$ *has the following properties:*

1. *It stabilises the system when only controlled by the actuators in* I_{N}*, at the quadratic cost*
$$J(\boldsymbol{u}_{\mathrm{N}}, \gamma) = \frac{1}{2}\gamma^{\mathrm{T}}\boldsymbol{P}_{\mathrm{N}}\gamma$$
2. *It also stabilises the system when controlled by the actuators in* $I_{\mathrm{N}} \cup I_{\mathrm{f}}$ *where* I_{f} *is any subset of* I_{F}*, at a quadratic cost less than or equal to* $\frac{1}{2}\gamma^{\mathrm{T}}\boldsymbol{P}_{\mathrm{N}}\gamma$.

Discussion. Let $\{I_{\mathrm{N}}, I_{\mathrm{F}}\}$ be a partition of the set of actuators I such that I_{N} is a minimal recoverable configuration. It follows that the control law $\boldsymbol{u}_{\mathrm{N}}(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}_{\mathrm{N}}\boldsymbol{x}(t)$ where $\boldsymbol{P}_{\mathrm{N}}$ is the unique stabilising solution of Eq. (8.26) is admissible, i.e. the quadratic cost $\frac{1}{2}\gamma^{\mathrm{T}}\boldsymbol{P}_{\mathrm{N}}\gamma$ associated with the stable closed-loop system

$$\dot{\boldsymbol{x}}(t) = \left[\boldsymbol{A} - \boldsymbol{B}_{\mathrm{N}}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}_{\mathrm{N}}\right]\boldsymbol{x}(t)$$

satisfies the cost constraint. From the second property of the Reliable Control Theorem, this control law is also admissible for any system configuration $I_N \cup I_{\mathrm{f}}$ where $I_{\mathrm{f}} \subseteq I_{\mathrm{F}}$. It is therefore concluded that under this control law, the system is passively fault tolerant with respect to all faults that are "smaller" than I_{F} (i.e. the set of faulty actuators is included in I_{F}). Since the set of those faults corresponds to the set of configurations that are the predecessors of configuration I_{N}, the Reliable Control Theorem can be reformulated as follows.

Theorem 8.2 (Reliable Control reformulated) *Let* I_{N} *be a minimal recoverable configuration. Then, the control law* $\boldsymbol{u}_{\mathrm{N}}(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}_{\mathrm{N}}\boldsymbol{x}(t)$ *where* $\boldsymbol{P}_{\mathrm{N}}$ *is the unique definite positive stabilising solution of Eq. (8.26) is admissible for all its predecessors.*

Practical implementation. This new formulation shows that a mix of passive and active fault tolerance is able to cope with all the recoverable configurations: each minimal recoverable configuration is associated with its own control law (the active part of the strategy), and this control law recovers the set of all its predecessors (the passive part). The result is that instead of containing as many control laws as the number of recoverable configurations, the control bank now contains as many control laws as the number of minimal recoverable configurations, which may be

much smaller. However, there is a non-uniqueness problem to be dealt with, since a recoverable configuration may belong to the predecessors of more than one minimal recoverable configuration, hence several control laws are available for its recovery. Since all of them are admissible the designer is free to select the one that best fits some extra design criterion, for example selecting the one associated with the minimal cost.

Example 8.5 PACT control bank
Consider the linear quadratic problem associated with a system with 6 states and 4 actuators $I = \{1,\ 2,\ 3,\ 4\}$ where the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are as follows

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 & 1 & 2 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 2 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}, \quad \boldsymbol{B}^{\mathrm{T}} = \begin{pmatrix} 1\,0\,1\,0\,1\,1 \\ 1\,1\,1\,0\,0\,0 \\ 1\,0\,0\,0\,1\,0 \\ 0\,0\,1\,1\,1\,0 \end{pmatrix}$$

and the cost is defined by

$$J(\boldsymbol{x}_0,\ \boldsymbol{u}) = \frac{1}{2}\int_0^{\infty} \left[\boldsymbol{x}(t)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(t) + \boldsymbol{u}(t)^{\mathrm{T}}\boldsymbol{R}\boldsymbol{u}(t)\right] \mathrm{d}t.$$

Let $\boldsymbol{Q}$ and $\boldsymbol{R}$ be identity matrices of appropriate dimensions. The optimal control of the nominal system is $\boldsymbol{u}^*(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}(t)$. It results in the minimal cost $J(\gamma) = \frac{1}{2}\gamma^{\mathrm{T}}\boldsymbol{P}\gamma$ where γ is the initial condition and $\boldsymbol{P}$ is the unique symmetric positive definite stabilising solution of the Riccati equation associated with the nominal system. It can be checked that for the nominal configuration, the optimal state-feedback control results in a cost matrix $\boldsymbol{P}$ whose maximal eigenvalue is $\lambda_{\max}(\boldsymbol{P}) = 7.3554$.

Now, controlling a configuration $I_{\mathrm{N}} \subseteq I$ by some control law u results in the cost

$$J_{\mathrm{N}}(\gamma,\ \boldsymbol{u}) = \frac{1}{2}\int_0^{\infty} \left[\boldsymbol{x}(t)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}(t) + \boldsymbol{u}(t)^{\mathrm{T}}\boldsymbol{R}_{\mathrm{N}}\boldsymbol{u}(t)\right] \mathrm{d}t$$

where $\boldsymbol{R}_{\mathrm{N}} = diag\,\{r_i,\ i = 1, 2, 3, 4\}$ and $r_i \in \{0, 1\}$ according to the fact that actuator i is present or not in configuration I_{N} (indeed, switched off actuators do not imply any energy cost, whatever the control signal that is sent to them). The specification is that a control law $\boldsymbol{u}$ is admissible if it satisfies

$$\forall \gamma \in R^n :\ J_{\mathrm{N}}(\gamma,\ \boldsymbol{u}) \leq \frac{1}{2}\rho\gamma^{\mathrm{T}}\boldsymbol{P}\gamma \tag{8.27}$$

where $\rho > 1$ is the admissible performance degradation factor, meaning that performance degradation is accepted as long as the degraded cost does not exceed ρ times the optimal nominal cost. It follows that a recoverable configuration I_{N} is such that there exists a unique symmetric positive definite stabilising solution $\boldsymbol{P}_{\mathrm{N}}$ to the Riccati equation associated with I_{N} which satisfies:

$$\forall \gamma :\ \gamma^{\mathrm{T}}(\boldsymbol{P}_{\mathrm{N}} - \rho\boldsymbol{P})\gamma \leq 0$$

(indeed, the minimal cost achievable by configuration I_{N} is $\frac{1}{2}\gamma^{\mathrm{T}}\boldsymbol{P}_{\mathrm{N}}\gamma$).

Let the specification be defined by $\rho = 15$. Still naming the configurations after the actuators they contain, e.g. the nominal configuration I is 1234, the failure of actuator 2 results in configuration 134, etc., the set of recoverable configurations is

$$\mathcal{R}_{\rho=15} = \{1234,\ 123,\ 124,\ 12,\ 134,\ 234,\ 23,\ 24\}$$

while the minimal recoverable configurations are $\mathcal{M}_{\rho=15} = \{12,\ 134,\ 23,\ 24\}$. Figure 8.5 shows the lattice of configurations where recoverable configurations are white, non-recoverable configurations are grey, and minimal recoverable ones have a bold contour.

Active control bank. Each recoverable configuration being associated with its own control law, the active control bank contains 8 laws, for example the 8 optimal state feedbacks associated with the 8 recoverable configurations:

$$\mathcal{U}_{\text{active}} = \left\{ \boldsymbol{u}_{\text{N}}(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}_{\text{N}}^{\text{T}}\boldsymbol{P}_{\text{N}}\boldsymbol{x}(t),\ I_{\text{N}} \in \mathcal{R}_{\rho=15} \right\}$$

Note that choosing the optimal control law associated with each configuration insures the minimal cost is obtained when this configuration occurs as the result of faults, but there is no need for the control laws to be optimal: they might be chosen arbitrarily provided they satisfy the admissibility constraints.

Passive–active control bank. In this scheme, each minimal recoverable configuration is associated with a control law that is admissible for all its predecessors. The bank now contains only 4 control laws, namely,

$$\mathcal{U}_{\text{PACT}} = \left\{ \boldsymbol{u}_{\text{N}}(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\text{T}}\boldsymbol{P}_{\text{N}}\boldsymbol{x}(t),\ I_{\text{N}} \in \mathcal{M}_{\rho=15} \right\}. \tag{8.28}$$

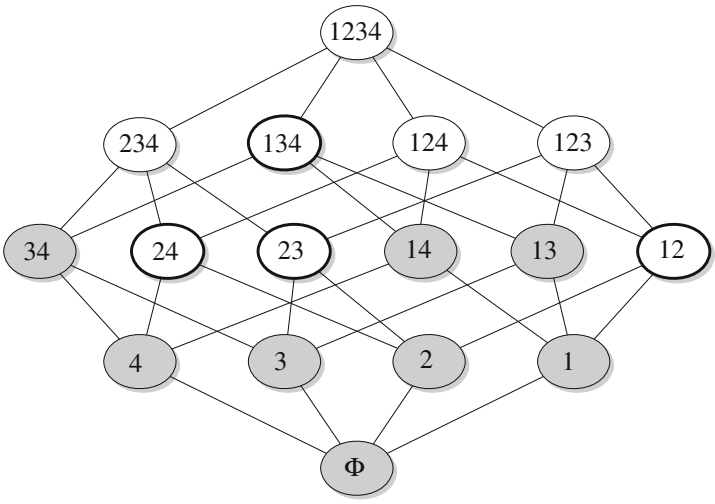


Fig. 8.5 The lattice of actuator configurations

Note that the law $\boldsymbol{u}_{\mathrm{N}}(t)$ is optimal for configuration $I_{\mathrm{N}} \in \mathcal{M}_{\rho=15}$ but is only admissible for the predecessors $\mathcal{P}(I_{\mathrm{N}}) \setminus I_{\mathrm{N}}$. The respective performance indexes are

$$\begin{aligned}\lambda_{\max}(P_{12}) &= 17.4285\\ \lambda_{\max}(P_{134}) &= 32.9450\\ \lambda_{\max}(P_{23}) &= 16.5649\\ \lambda_{\max}(P_{24}) &= 18.6938.\end{aligned}$$

Let $\mathcal{R}(u)$ be the set of configurations recovered by the control law u, one has

$$\begin{aligned}\mathcal{R}(u_{12}) &= \left\{1234,\ 123,\ 124,\ 12,\ \underline{234},\ \underline{23}\right\}\\ \mathcal{R}(u_{134}) &= \left\{1234,\ \underline{123},\ 134,\ \underline{234}\right\}\\ \mathcal{R}(u_{23}) &: \{1234,\ 123,\ 234,\ 23\}\\ \mathcal{R}(u_{24}) &: \{1234,\ 124,\ 234,\ 24\}.\end{aligned}$$

One remarks that $\mathcal{R}\left(u_{I_{\mathrm{N}}}\right)$ indeed not only includes, for each minimal recoverable configuration I_{N}, all its predecessors, but it may also include more (those that are underlined). Another remark is that non-minimal recoverable configurations can be recovered by several control laws, hence the need for a decision procedure.

Table 8.2 gives, for each recoverable configuration, the list of control laws by which it can be recovered. The simplest decision procedure selects the one with the best performance (underlined). □

Reducing the control bank size. Let $\mathcal{U}$ be a passive–active control bank that recovers all the recoverable faults. It may happen that the designer is happy with a bank $\mathcal{U}_{\text{smaller}}$ that contains a smaller number of laws at the price of recovering only a subset of recoverable faults (i.e. at the price of being less fault-tolerant). This trade-off will be considered later since it is connected with the evaluation of fault tolerance, as one can guess from the wording "*being less fault-tolerant*". This section investigates the minimality of the control bank for a given subset of faults to be recovered.

Table 8.2 Control laws for recovery

Configurations	1234	123	124	12	134	234	23	24
Admissible control laws	u_{12} u_{134} $\underline{u_{23}}$ u_{24}	u_{12} u_{134} $\underline{u_{23}}$	$\underline{u_{12}}$ u_{24}	u_{12}	u_{134}	u_{12} u_{134} $\underline{u_{23}}$ u_{24}	u_{12} $\underline{u_{23}}$	u_{24}

Definition 8.7 (*Minimal Control bank*) A control bank $\mathcal{U}$ is minimal with respect to a given set of faults if there is no proper subset of $\mathcal{U}$ that recovers all these faults.

Given a passive–active control bank $\mathcal{U}$ that recovers a set of faults $\mathcal{R}(\mathcal{U})$ the reduction to a minimal control bank problem has to be found in the following problem:

Problem 8.5 (*Reduction problem*)

1. Check whether $\mathcal{U}$ is minimal or not,
2. if not, find a minimal control bank $\mathcal{U}_{\min}$.

Let $\mathcal{R}(\mathcal{U})$ be the set of faults recovered by all the control laws in the bank $\mathcal{U}$ then one has

$$\mathcal{R}(\mathcal{U}) = \cup_{u \in \mathcal{U}} \mathcal{R}(u).$$

Let $U \subset \mathcal{U}$ be a proper subset of control laws. It recovers the set of faults:

$$\mathcal{R}(U) = \cup_{u \in U} \mathcal{R}(u).$$

By comparing $\mathcal{R}(\mathcal{U})$ and $\mathcal{R}(U)$, it is concluded that if $\mathcal{R}(U) = \mathcal{R}(\mathcal{U})$ then $\mathcal{U}$ is not minimal (indeed U is a proper subset that recovers the same faults), and either U or some subsets of U are minimal. On the contrary, if $\mathcal{R}(U) \subset \mathcal{R}(\mathcal{U})$ then neither U nor any of its successors can recover all the faults to be recovered. This remark leads to the following algorithm for the determination of the minimal control banks that recover the same set if faults as a given PACT control bank.

Algorithm 8.1 *Reduction of a PACT bank of control laws*

Given: Bank of control laws $\mathcal{U}$
Recovered faults $\mathcal{R}(\mathcal{U})$.
Initialisation: Put $\mathcal{U}$ into the list "possible", initialize two empty lists "minimal" and "impossible"
While the list "Possible" is not empty
For each member $\mathcal{V}$ of this list
If all its direct successors U are such that $\mathcal{R}(U) \subset \mathcal{R}(\mathcal{U})$
then remove $\mathcal{V}$ from the list "possible" and move it into the list "minimal"
else give the direct successors that satisfy $\mathcal{R}(U) = \mathcal{R}(\mathcal{V})$ the label "possible" and change the label of $\mathcal{V}$ into "impossible"
Result: list of minimal banks that recover all the faults $\mathcal{R}(\mathcal{U})$.

Example 8.6 Minimal control bank
The passive–active control bank of Example 8.5 resulted in a control bank with four control laws:

$$\mathcal{U}_{\text{PACT}} = \{u_{12}, u_{134}, u_{23}, u_{24}\}$$

Table 8.3 Level 1 subsets

Subsets	Recovered faults	Comment
$\{u_{12}, u_{134}, u_{23}\}$	{1234, 123, 124, 12, 234, 23, 134, 24}	Possibly minimal
$\{u_{12}, u_{134}, u_{24}\}$	{1234, 123, 124, 12, 234, 23, 134, 24}	Possibly minimal
$\{u_{12}, u_{23}, u_{24}\}$	{1234, 123, 124, 12, 234, 23, 24}	Impossible
$\{u_{134}, u_{23}, u_{24}\}$	{1234, 123, 134, 234, 23, 124, 24}	Impossible

Table 8.4 Level 2 subsets

Subsets	Recovered faults	Comment
$\{u_{12}, u_{134}\}$	{1234, 123, 124, 12, 234, 23, 134}	Impossible
$\{u_{12}, u_{23}\}$	{1234, 123, 124, 12, 234, 23}	Impossible
$\{u_{134}, u_{23}\}$	{1234, 134, 123, 234, 23}	Impossible

that were able to recover all the recoverable faults { 1234, 123, 124, 12, 134, 234, 23, 24 } according to the following list:

$$\begin{aligned}\mathcal{R}(u_{12}) &= \{1234,\ 123,\ 124,\ 12,\ \underline{234},\ \underline{23}\}\\ \mathcal{R}(u_{134}) &= \{1234,\ \underline{123},\ 134,\ \underline{234}\}\\ \mathcal{R}(u_{23}) &= \{1234,\ 123,\ 234,\ 23\}\\ \mathcal{R}(u_{24}) &= \{1234,\ 124,\ 234,\ 24\}\end{aligned}$$

Exploring the Level 1 subsets of $\mathcal{U}_{\text{PACT}}$ shows that two banks with 3 control laws, namely $\{u_{12}, u_{134}, u_{23}\}$ and $\{u_{12}, u_{134}, u_{24}\}$ are able to recover all the recoverable faults (Table 8.3).

Analysing the subsets of $\{u_{12}, u_{134}, u_{23}\}$ and $\{u_{12}, u_{134}, u_{24}\}$ shows that none of them can recover all the faults (Table 8.4 shows the results for $\{u_{12}, u_{134}, u_{23}\}$).

It is, therefore, concluded that the PACT control bank $\mathcal{U}_{\text{PACT}} = \{u_{12}, u_{134}, u_{23}, u_{24}\}$ can be replaced with no loss of recoverability by a 3-laws control bank: either $\{u_{12}, u_{134}, u_{23}\}$ or $\{u_{12}, u_{134}, u_{24}\}$. □

8.5.3 Reducing the Reliability Over-Cost

Let $\mathcal{M}$ be the set of minimal recoverable configurations of a given system, and let $\mathcal{U}$ be the PACT control bank where each control law $\boldsymbol{u}_{\text{N}}(t) \in \mathcal{U}$ is associated with one minimal recoverable configuration $I_{\text{N}} \in \mathcal{M}$.

Reliability over-cost. As already noted, $\boldsymbol{u}_{\text{N}}(t)$ is optimal for configuration I_{N}, but for any other configuration $I_K \in \mathcal{P}(I_{\text{N}})$ it is only admissible. Indeed, configuration I_K achieves the minimal cost $\frac{1}{2}\boldsymbol{\gamma}^{\text{T}}\boldsymbol{P}_K\boldsymbol{\gamma}$ under the control law $\boldsymbol{u}_K(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\text{T}}\boldsymbol{P}_K\boldsymbol{x}(t)$ where $\boldsymbol{P}_K$ is the unique solution of the Riccati equation

$$\boldsymbol{Q} + \boldsymbol{A}^{\text{T}}\boldsymbol{P}_K + \boldsymbol{P}_K\boldsymbol{A} - \boldsymbol{P}_K\boldsymbol{B}_K\boldsymbol{R}^{-1}\boldsymbol{B}_K^{\text{T}}\boldsymbol{P}_K = \boldsymbol{O}$$

while it is well known that under the (non-optimal) control law

$$\boldsymbol{u}_{\mathrm{N}}(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}_{\mathrm{N}}\boldsymbol{x}(t)$$

the cost is $\frac{1}{2}\gamma^{\mathrm{T}}\boldsymbol{P}_{\mathrm{N,K}}\gamma$ where $P_{\mathrm{N,K}}$ is symmetric positive definite and given by the Lyapunov equation:

$$\boldsymbol{Q} + \boldsymbol{P}_{\mathrm{N}}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}_{\mathrm{N}} + \boldsymbol{P}_{\mathrm{N,K}}\left(\boldsymbol{A} - \boldsymbol{B}_K\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}_{\mathrm{N}}\right) + \left(\boldsymbol{A} - \boldsymbol{B}_K\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}_{\mathrm{N}}\right)^{\mathrm{T}}\boldsymbol{P}_{\mathrm{N,K}} = \boldsymbol{O}$$

It is concluded that for configuration I_K, the reliability over-cost to be paid for using $\boldsymbol{u}_{\mathrm{N}}(t)$ instead of $\boldsymbol{u}_K(t)$ is $\frac{1}{2}\gamma^{\mathrm{T}}\left[\boldsymbol{P}_{\mathrm{N,K}} - \boldsymbol{P}_K\right]\gamma$.

Trade-off control bank. Since the nominal configuration I is expected to occur most of the time, it may be sensible to add the nominal control law to the minimal PACT bank. It follows that no reliability over-cost is paid in the nominal configuration, at the cost of abandoning the minimality of the control bank.

Example 8.7 Trade-off control bank
In Example 8.6, a minimal control bank with three laws, namely $\{u_{12}, u_{134}, u_{23}\}$ or $\{u_{12}, u_{134}, u_{24}\}$ was able to recover all the recoverable configurations. In this case, one would have chosen the first bank, since it contains u_{23} which gives the smallest cost

$$\frac{1}{2}\gamma^{\mathrm{T}}\boldsymbol{P}_{23,1234}\gamma$$

in the nominal situation, as highlighted in Table 8.2. However, implementing the trade-off bank $\{u_{1234}, u_{12}, u_{134}, u_{23}\}$ results in an optimal cost for the nominal system and a minimal number of control laws to obtain an admissible cost for the other (recoverable) configurations. □

More generally, a trade-off control bank can be designed by associating some recoverable configurations (Subset1) with their optimal control law, while the rest (Subset2) is controlled by the PACT bank associated with the minimal recoverable configurations. The system performances are optimal as long as the current configuration belongs to Subset1, at the cost of increasing the number of control laws in the overall control bank. For Subset2, the cost reduction problem can be stated as follows:

Problem 8.6 (*Cost reduction problem*)
Given a minimal recoverable configuration I_{N}, find a control law that minimises the cost achieved by some pre-selected configuration $I_L \in \mathcal{P}(I_{\mathrm{N}})$ under the constraint that it is admissible for all the configurations in $\mathcal{P}(I_{\mathrm{N}})$.

Note that the optimal control of configuration I_L indeed minimises the cost achieved by this configuration, but there is no reason for it to be admissible for all the

configurations in $\mathcal{P}(I_N)$. Conversely, the reliable control $\boldsymbol{u}_N(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}_N\boldsymbol{x}(t)$ is admissible for all the configurations in $\mathcal{P}(I_N)$ but there is no reason for it to yield the minimal cost when applied to I_L.

The cost reduction problem can be addressed by introducing a degree of freedom $\boldsymbol{H}$ in the control law, namely $\boldsymbol{u}(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{H}\boldsymbol{x}(t)$ instead of $\boldsymbol{u}(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}_N\boldsymbol{x}(t)$, where $\boldsymbol{H}$ is symmetric positive definite. Applying this control law to a configuration $I_K \in \mathcal{P}(I_N)$ results in the closed-loop matrix $\boldsymbol{A} - \boldsymbol{B}_K\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{H}$ and (assuming it is stable), in the cost $\frac{1}{2}\gamma^T\boldsymbol{W}_K\gamma$ where $\boldsymbol{W}_K$ is symmetric positive definite and given by the Lyapunov equation:

$$\boldsymbol{Q} + \boldsymbol{H}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{H} + \boldsymbol{W}_K\left(\boldsymbol{A} - \boldsymbol{B}_K\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{H}\right) + \left(\boldsymbol{A} - \boldsymbol{B}_K\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{H}\right)^T\boldsymbol{W}_K = \boldsymbol{O} \tag{8.29}$$

Let $\mathcal{H}$ be the set of symmetric positive definite matrices $\boldsymbol{H}$ that satisfy the conditions that $\boldsymbol{A} - \boldsymbol{B}_K\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{H}$ is stable and $\forall I_K \in \mathcal{P}(I_N)$, $\frac{1}{2}\gamma^T\boldsymbol{W}_K\gamma$ is admissible.

Applying the control law $\boldsymbol{u}(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{H}\boldsymbol{x}(t)$ to configuration I_L results in the cost $\frac{1}{2}\gamma^T\boldsymbol{W}_L\gamma$ and therefore, the cost reduction problem is nothing but the optimisation problem: find $\boldsymbol{H}$ so as to minimise $\lambda_{\max}(\boldsymbol{W}_L)$ under the constraints $\boldsymbol{H} \in \mathcal{H}$.

Unfortunately, this problem appears to be non-convex and difficult to solve. However, it is possible to build a sequence of control laws $\boldsymbol{u}^k(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{H}^k\boldsymbol{x}(t)$, $(k = 1, 2, \ldots)$ that improve the cost of the selected configuration while satisfying the constraints. The following algorithm is based on an adaptation of the Newton-Kleinman procedure. It can be shown that it produces a converging sequence of control laws $\boldsymbol{u}^k(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{H}^k\boldsymbol{x}(t)$, $(k = 1, 2, \ldots)$ that stabilise all the configurations in $\mathcal{P}(I_N)$ and are such that $\boldsymbol{P}_L \leq \cdots \leq \boldsymbol{W}_L^{k+1} \leq \boldsymbol{W}^k \leq \cdots \leq \boldsymbol{P}_N$ where $\boldsymbol{W}_L^{k+1} \leq \boldsymbol{W}^k$ means that for any initial condition γ, the quadratic form $\gamma^T\boldsymbol{W}^k\gamma$ is a decreasing function of k.

Algorithm 8.2 *Cost reduction problem*

Given: A minimal recoverable configuration I_N
a pre-selected configuration $I_L \in \mathcal{P}(I_N)$
an arbitrary small positive number ε, a matrix norm $\|.\|$

Initialisation: $\boldsymbol{H}^0 = \boldsymbol{P}_N$ and $\boldsymbol{W}_L^{-1} = \infty$

While: STOP condition not fulfilled

1. Solve the Lyapunov equation $\boldsymbol{Q} + \boldsymbol{H}^k\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{H}^k + \boldsymbol{W}^k\left(\boldsymbol{A} - \boldsymbol{B}_L\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{H}^k\right) + \left(\boldsymbol{A} - \boldsymbol{B}_L\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{H}^k\right)^T\boldsymbol{W}^k = \boldsymbol{O}$ for $\boldsymbol{W}^k$
2. Update $\boldsymbol{H}^{k+1} = p^k\boldsymbol{H}^k + q^k\boldsymbol{W}^k$, where $q^k = \max\left\{\zeta : \zeta \in [0, 1],\ \boldsymbol{H}^{k+1} \in \mathcal{H}\right\}$ and $p^k = 1 - q^k$

3. Check the STOP condition $\left\| \boldsymbol{W}_L^{k+1} - \boldsymbol{W}^k \right\| \leq \varepsilon$

Result: a convergent sequence of control laws $\boldsymbol{u}^k(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{H}^k\boldsymbol{x}(t)$, $(k = 1, 2, \ldots)$ that satisfy the admissibility constraints for all configurations in $\mathcal{P}(I_{\mathrm{N}})$ and decrease the quadratic cost associated with configuration I_L.

Notice that the pure Newton-Kleinman scheme is obtained if the updating procedure in Step 2 is applied with $p^k = 0$ and $q^k = 1$ for all k. This scheme produces the optimal control matrix associated with configuration I_L under no constraint. The updating procedure in Step 2 is aimed at satisfying the constraints $\boldsymbol{H} \in \mathcal{H}$.

Example 8.8 Cost reduction
In Example 8.7, consider the minimal recoverable configuration 12. The control law u_{12} is admissible for configurations $\{1234,\ 123, 124,\ 12,\ \underline{234},\ \underline{23}\}$ but using it in the nominal configuration 1234 gives the cost matrix $P_{12,1234}$ whose maximal eigenvalue is $\lambda_{\max}\left(P_{12,1234}\right) = 12.267$. However, any control law $\boldsymbol{u}(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x}(t)$, where $\boldsymbol{H} = \boldsymbol{H}^{\mathrm{T}} > 0$ is better than u_{12} and admissible for all configurations in $\mathcal{P}(12)$ if it satisfies the following conditions:

- the predecessors are stable: $\forall I_K \in \{1234,\ 123,\ 124,\ 12\}$, $\boldsymbol{A} - \boldsymbol{B}_K\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{H}$ is Hurwitz,
- the predecessors are admissible: $\forall I_K \in \{1234,\ 123,\ 124,\ 12\}$, $W_K \leq 15P_{1234}$, where W_K is the solution of Eq. (8.29)
- for any initial condition the cost associated with the nominal configuration $\frac{1}{2}\gamma^{\mathrm{T}}\boldsymbol{W}_{1234}\gamma$ is smaller than $\frac{1}{2}\gamma^{\mathrm{T}}\boldsymbol{P}_{12,1234}\gamma$.

It can be checked that, applying the pure Newton-Kleinman algorithm $p^k = 0$, $q^k = 1$ for all k, results in a sequence of cost matrices $\boldsymbol{W}_{1234}^k$ that decrease from the solution $\boldsymbol{W}_{1234}^0 = \boldsymbol{P}_{12,1234}$ of the Lyapunov equation

$$\boldsymbol{Q} + \boldsymbol{P}_{12}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}_{12} + \boldsymbol{W}_{1234}^0\left(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}_{12}\right) + \left(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}_{12}\right)^{\mathrm{T}}\boldsymbol{W}_{1234}^0 = \boldsymbol{O}$$

to the optimal solution $\boldsymbol{W}_{1234}^{\infty} = \boldsymbol{P}_{1234}$ associated with the nominal system. However, as soon as the first iteration, $\boldsymbol{H}^1$ violates the admissibility constraint, so the update law in the algorithm must be used. The result is displayed in Table 8.5. The control $\boldsymbol{u}(t) = -\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{H}^2\boldsymbol{x}(t)$ is

Table 8.5 Results for configuration 12

Iteration	0	1	2
$\lambda_{\max}\left(W_{1234}^k\right)$	12.267	10.159	10.159
$q_{\max}$	0.648	0.000	0.000

Table 8.6 Results for all configurations

	Reliable control	Cost reduction	Decrease
Configuration 12	12.267	10.159	17.18%
Configuration 134	19.340	15.892	17.83%
Configuration 23	12.743	7.873	38.22%
Configuration 24	10.869	9.182	15.53%

admissible for all the predecessors $\mathcal{P}$ (12), and decreases the nominal configuration cost by 17.18 % when compared with the reliable control u_{12}.

Table 8.6 compares the nominal configuration costs achieved by the control law associated with each configuration in $\mathcal{M}_{\rho=15}$ respectively for the reliable control and the cost reduced control. □

8.6 Fault-Tolerance Evaluation

The system is tolerant to actuator faults, when the reconfiguration strategy is used, as long as the current configuration $I_N(t)$ belongs to the set of recoverable configurations $\mathcal{R}$. Introducing some measure $\mu(\mathcal{R})$ of this set should therefore give an idea about the overall system fault tolerance. On another hand, let the system configuration be $I_N(t)$ at time t and assume there is no repair during its operation, then actuator failures can only move the discrete state to configurations within the set of successors $\mathcal{S}(I_N(t))$, among which only those in the intersection $\mathcal{R} \cap \mathcal{S}(I_N(t))$ are recoverable. The "remaining" fault tolerance at time t can therefore be evaluated by the measure $\mu(\mathcal{R} \cap \mathcal{S}(I_N(t)))$. Note that $\mathcal{R} = \mathcal{R} \cap \mathcal{S}(I)$ because $\mathcal{S}(I) = 2^I$, and therefore the measure $\mu(\mathcal{R})$ is the "remaining" fault tolerance at the initial time, assuming that the system is then in its nominal configuration. Two kinds of measures, namely deterministic or probabilistic measures can be used.

8.6.1 Deterministic Measures

Deterministic measures do not use any model of the transitions from one configuration to another. The most important ones are the redundancy degrees which are based on the number of levels, in the lattice of system configurations, between a recoverable configuration $I_N(t)$ and the set of non-recoverable ones.

Definition 8.8 (*Strong redundancy degree*) The strong redundancy degree is the measure $\mu(\mathcal{R} \cap \mathcal{S}(I_N(t)))$ defined by:

$$k_{\text{strong}}[I_N(t)] = \min\left\{|\Sigma| : \Sigma \subseteq I_N(t) \wedge I_N(t) \setminus \Sigma \in \overline{\mathcal{R}}\right\}, \tag{8.30}$$

where $|\Sigma|$ is the cardinal number of the set Σ.

$k_{\text{strong}}[I_{\text{N}}(t)]$ is the length of the shortest path, in the lattice of system configurations, between a recoverable configuration $I_{\text{N}}(t)$ and the set of non-recoverable configurations. In other words, no matter which actuators are lost, as long as their number does not exceed $k_{\text{strong}}[I_{\text{N}}(t)] - 1$, the fault is recoverable.

Definition 8.9 (*Weak redundancy degree*) The weak redundancy degree is the measure $\mu(\mathcal{R} \cap \mathcal{S}(I_{\text{N}}(t)))$ defined by:

$$k_{\text{weak}}[I_{\text{N}}(t)] = \max\{|\Sigma| : \Sigma \subseteq I_{\text{N}}(t) \wedge I_{\text{N}}(t) \setminus \Sigma \in \mathcal{R}\} \tag{8.31}$$

It is the length of the longest path, in the lattice of system configurations, between a recoverable configuration $I_{\text{N}}(t)$ and the set of non-recoverable ones. In other words, the largest set of actuators whose loss can be tolerated from the current configuration $I_{\text{N}}(t)$ is of size $k_{\text{weak}}[I_{\text{N}}(t)]$.

The redundancy degrees enjoy nice practical interpretations. It follows from their definition that

$$\forall I_{\text{n}}(t) \in \mathcal{R},\ k_{\text{strong}}[I_{\text{N}}(t)] \leq k_{\text{weak}}[I_{\text{N}}(t)].$$

The *coverage* is another deterministic measure that is sometimes used in addition to the redundancy degrees.

Definition 8.10 (*Coverage*) The coverage is the measure $\mu(\mathcal{R} \cap \mathcal{S}(I_{\text{N}}(t)))$ defined by the ratio between the number of recoverable configurations and the total number of possible configurations.

Its interpretation is not so straightforward as that of the redundancy degrees, but it is easy to compute, and it may provide some useful insight with respect to the usefulness of the individual system components. For example, it allows a quick evaluation of the individual components usefulness, as discussed in Sect. 8.6.3.

8.6.2 Probabilistic Measures

Probabilistic measures assume that a model that governs the transitions from one configuration to another one is available. Then, the set $\mathcal{R} \cap \mathcal{S}(I_{\text{N}}(t))$ can be measured using reliability concepts. Indeed, for any pair of time instants t_1, t_2 such that $t_2 > t_1$ let $\pi_\sigma(t_1, t_2)$ be the probability for actuator σ to be healthy at time t_2 subject to the condition that it was healthy at time t_1. Assume this function is known for all actuators, that actuators faults are independent, and that the nominal configuration I is the current one at the initial time. Then, the probability for the system discrete state to be I_{N} at time t is given by

$$\Pr[I_{\text{N}},\ 0,\ t] = \prod_{\sigma \in I_{\text{N}}} \pi_\sigma(t,\ 0) \prod_{\sigma \notin I_{\text{N}}} [1 - \pi_\sigma(t,\ 0)]. \tag{8.32}$$

Let the time window $[0,\ T]$ define the duration of the system mission, then fault tolerance is guaranteed provided no configuration in $\overline{\mathcal{R}}$ becomes active on $[0,\ T]$ (indeed, the specification is satisfied as long as the current configuration belongs to $\mathcal{R}$). It follows that the success probability on $[0,\ T]$ is given by

$$\Pr[I,\ 0,\ T] = \sum_{I_{\mathrm{N}} \in \mathcal{R}} \Pr[I_{\mathrm{N}},\ 0,\ T]. \tag{8.33}$$

Starting with the nominal configuration I at the initial time, the time during which the system will operate successfully is the time before it enters a configuration in $\overline{\mathcal{R}}$. This is a random variable, whose probability distribution is given by Eq. (8.33). A possible alternative measure of the fault-tolerance capability is the mean-time to failure:

$$MTTF(I,\ 0) = \int_0^\infty \Pr[I,\ 0,\ T]\,\mathrm{d}T. \tag{8.34}$$

8.6.3 Sensitivity

The size of $\mathcal{R}$ (and consequently the size of $\mathcal{R} \cap \mathcal{S}(I_{\mathrm{N}}(t))$ depends on the difficulty for the specification to be satisfied and on the size of I. It follows that two kinds of sensitivities can be considered.

Sensitivity with respect to the specifications. Consider the triple $(I,\ \mathrm{Spec1},\ \mathrm{Spec2})$, where Spec1 and Spec2 are two specifications, then one has

$$(\mathrm{Spec1} \Rightarrow \mathrm{Spec2}) \Rightarrow \mathcal{R}_{\mathrm{Spec1}} \subseteq \mathcal{R}_{\mathrm{Spec2}}, \tag{8.35}$$

where Spec1 $\Rightarrow$ Spec2 means that Specification 2 is *weaker* than—or is a *degraded* specification with respect to—Specification 1, and $\mathcal{R}_{\mathrm{Spec1}}$ (resp. $\mathcal{R}_{\mathrm{Spec2}}$) is the set of configurations that are recoverable when the nominal set of actuators is I and the specification is Spec1 (resp. Spec2). Indeed, any configuration that satisfies Spec1 also satisfies Spec2.

The sensitivity with respect to the specifications is easily evaluated from the difference of the above deterministic or probabilistic measures associated with each set of recoverable configurations. For example, the strong redundancy degree of a given configuration $I_{\mathrm{N}}(t)$ is

$$k_{\mathrm{strong}}[I_{\mathrm{N}}(t)] = \min\left\{|\Sigma| : \Sigma \subseteq I_{\mathrm{N}}(t) \wedge I_{\mathrm{N}}(t) \setminus \Sigma \in \overline{\mathcal{R}}_{\mathrm{Spec1}}\right\}$$

or

$$k_{\mathrm{strong}}[I_{\mathrm{N}}(t)] = \min\left\{|\Sigma| : \Sigma \subseteq I_{\mathrm{N}}(t) \wedge I_{\mathrm{N}}(t) \setminus \Sigma \in \overline{\mathcal{R}}_{\mathrm{Spec2}}\right\}$$

according to the selected specifications.

An interesting development is associated with specifications Spec (θ) that are monotonous with respect to some parameter θ in the following sense:

$$\theta_1 \leq \theta_2 \Rightarrow \mathcal{R}_{\text{Spec}(\theta_1)} \subseteq \mathcal{R}_{\text{Spec}(\theta_2)}$$

θ may for example be interpreted as a cost: the more one is ready to spend, the larger the set of recoverable configurations. With this interpretation in mind, let $\mathcal{R}_{\text{wish}}$ be a set of configurations that are wished to be recoverable. The value

$$\theta_{\text{optimal}} = \min \left\{ \theta : \mathcal{R}_{\text{Spec}(\theta)} \supseteq \mathcal{R}_{\text{wish}} \right\}$$

is the minimal cost at which the wished fault-tolerance specifications are achieved.

Note that the particular value θ_{critical} associated with $\mathcal{R}_{\text{wish}} = \{ I \}$ appears as the minimal cost to be paid for the existence of at least one solution (the nominal one) to the specification satisfaction problem.

Sensitivity with respect to the components. Similarly, for a given specification Spec let I_{Comp1} and I_{Comp2} be two sets of actuators (more generally two sets of components), then:

$$\left(I_{\text{Comp1}} \subseteq I_{\text{Comp2}} \right) \Rightarrow \mathcal{R}_{\text{Comp1}} \subseteq \mathcal{R}_{\text{Comp2}}, \tag{8.36}$$

where $\mathcal{R}_{\text{Comp1}}$ (resp. $\mathcal{R}_{\text{Comp2}}$) is the set of configurations that are recoverable when the nominal set of actuators is I_{Comp1} (resp. I_{Comp2}). The sensitivity with respect to the components is easily evaluated from the difference of the above deterministic or probabilistic measures associated with the sets I_{Comp1} and I_{Comp2}.

Two consequences of Eq. (8.36) may be of interest:

- Assume that two sets of components are such that $I_{\text{Comp1}} \subset I_{\text{Comp2}}$ and $\mathcal{R}_{\text{Comp1}} = \mathcal{R}_{\text{Comp2}}$. Then the components in $I_{\text{Comp2}} \setminus I_{\text{Comp1}}$ are useless for achieving the system objectives. It is concluded that the difference of the measures associated with $\mathcal{R}_{\text{Comp1}}$ and $\mathcal{R}_{\text{Comp2}}$ gives an idea of the usefulness of the subset of components $I_{\text{Comp2}} \setminus I_{\text{Comp1}}$.
- Assume that two sets of components are such that $I_{\text{Comp1}} \subset I_{\text{Comp2}}$ and that $\mathcal{R}_{\text{Comp1}} = \emptyset$ while $\mathcal{R}_{\text{Comp2}} \neq \emptyset$, then the subset $I_{\text{Comp2}} \setminus I_{\text{Comp1}}$ is (or contains) a critical component subset. Therefore, its removal from I_{Comp2} implies the impossibility that it will be impossible to satisfy the system specifications.

Example 8.9 Fault-tolerance evaluation

Assume the system in the previous example is expected to operate on the time interval $[0, T]$ with $T = 10^5$ h. Actuators 1 and 2 reliability data are $r_1(t, 0) = r_2(t, 0) = \exp\left(-4 \times 10^{-6} t\right)$, while actuators 3 and 4 are less prone to failures, namely $r_3(t, 0) = r_4(t, 0) = \exp\left(-4 \times 10^{-7} t\right)$.

Starting with the nominal configuration 1234 at the initial time, the PACT control bank $\mathcal{U}_{\text{PACT}} = \{u_{12}, u_{134}, u_{23}, u_{24}\}$ allows to recover all recoverable configurations. This results

in the redundancy degrees $k_{\text{strong}}\,[1234] = 1$, and $k_{\text{weak}}\,[1234] = 2$, meaning that the system operation can go on in the single failure case, whatever the failed actuator (the system is said to be fail-operational with respect to the first fault), and still works when some double faults occur. Note that using the smaller control bank $\mathcal{U} = \{u_{12},\ u_{134},\ u_{23}\}$ does not change the redundancy degrees, since the set of recoverable configurations remains unchanged (however, the performance of some configurations will be lower, although still admissible).

Using $\mathcal{U}_{\text{PACT}} = \{u_{12},\ u_{134},\ u_{23},\ u_{24}\}$, and assuming actuator failures are independent, the success probability computed from Eq. (8.33) is $\Pr\left[1234,\ 0,\ 10^5\right] = 0.8740$. Decreasing $\mathcal{U}_{\text{PACT}}$ to $\mathcal{U} = \{u_{12},\ u_{134},\ u_{23}\}$ does not change this figure.

Assuming the minimality of the control bank is an important point, note that if the bank $\{u_{12},\ u_{134},\ u_{23}\}$ is further decreased to $\{u_{12},\ u_{134}\}$ the recoverable configuration 24 cannot be recovered anymore. However, the probability for this configuration to occur within the mission time is so small (0.0083) that one could decide to implement the bank $\{u_{12}, u_{134}\}$ at the cost of not recovering fault 24 should it occur. Note that in this case, the redundancy degrees are still $k_{\text{strong}}\,[1234] = 1$ and $k_{\text{weak}}\,[1234] = 2$, but the success probability decreases from 0.8740 to 0.8657.

The admissible cost specification was defined by $\rho = 15$: a configuration is recoverable if there exists a control law such that the quadratic cost does not exceed 15 times the optimal cost of the nominal configuration. Table 8.7 shows the results obtained for different values of the cost parameter $\rho \in \{1, 2, \ldots, 7\}$. Only the values at which changes occur are displayed, and the last column recalls the results for $\rho = 15$.

To evaluate the sensitivity with respect to components, the effect of removing actuator subsets from I is computed. Table 8.8 shows the results for $\rho = 15$. Subsets whose removal results in $\mathcal{R} = \emptyset$ are not shown.

It is clearly seen that there is no useless component and that the critical component subsets are {12, 23, 24}: the failure of any of these subsets results in a non-recoverable configuration. This analysis is very useful for the architecture design problem, which consists in selecting the appropriate actuators to control the system in a fault tolerant way. For example, implementing only actuators123 would give $\mathcal{R} = \{123,\ 12,\ 23\}$, $k_{\max}\,[123] = 0$ and $k_{\min}\,[123] = 1$, meaning that the single fault fail operational property is lost. The success probability is drastically decreased to 0.0259. □

Table 8.7 Sensitivity to cost specification

ρ	1	2	4	5	15
Minimal recoverable configurations	1234	234	123 124 234	124 23	12 134 23 24
Strong redundancy degree	0	0	0	0	1
Weak redundancy degree	0	1	1	2	2
Success probability	0.4148	0.6188	0.6526	0.6609	0.8740

Table 8.8 Sensitivity to components

Removed subsets	1	2	3	4	13	14	34
Recoverable configurations	234 24 23	134	124 24 12	123 12 23	24	23	12
Strong redundancy degree	0	0	0	0	0	0	0
Weak redundancy degree	1	0	1	1	0	0	0
Success probability ($\times 10^2$)	21.06	20.40	2.59	2.59	0.83	0.83	0.07

8.7 Exercises

Exercise 8.1 Lattice-based analysis
Consider an over-actuated system with three actuators and two sensors:

$$\begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \end{pmatrix}$$

$$\begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}$$

In order to understand the generality of the lattice-based analysis, this exercise considers, instead of the quadratic control problem, a simple specification that allows hand calculations. The specification is as follows: the two closed-loop eigenvalues are wished to be real and equal to -2 when output feedback is used, namely for $i = 1, 2, 3$ one has $u_i(t) = k_{i1}y_1(t) + k_{i2}y_2(t)$ where k_{i1}, k_{i2} are the control gains to be designed.

1. Characterise the set of admissible nominal control laws.
2. Assuming the two sensors are not faulty, analyse the effect of actuator faults under the reconfiguration strategy.
3. Is it possible to analyse the effect of sensor faults under the reconfiguration strategy in the same way? □

Exercise 8.2 Reliable control
Let $abcd$ be the four actuators of a linear time-invariant system:

$$\boldsymbol{A} = \begin{pmatrix} 0 & 0.17 & 0.17 & 0.33 \\ -0.17 & -0.17 & 0.17 & 0 \\ 0.33 & 0.33 & 0 & 0.17 \\ 0 & 0.17 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{B}_0 = \begin{pmatrix} 0.50 & 0 & 0 & 0 \\ 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0.25 & 0 \\ 0 & 0 & 0 & 0.25 \end{pmatrix},$$

where matrix $\boldsymbol{A}$ is unstable, having the following set of eigenvalues:

$$\Lambda(\boldsymbol{A}) = \{-0.39; -0.031 \pm 0.141j; 0.28\}.$$

We are interested in the optimal quadratic control using the following weighting matrices:

$$\boldsymbol{Q} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad \boldsymbol{R} = I_4.$$

Faulty actuators are recovered, if possible, using the reconfiguration strategy. Under the recoverability specification that the optimal cost of the reconfigured system should not exceed four times the optimal cost of the healthy system, all configurations are recoverable except $\{ac, ad, bc, a, b, c, d\}$ as shown in Fig. 8.6, where the white nodes are recoverable while the grey nodes are not.

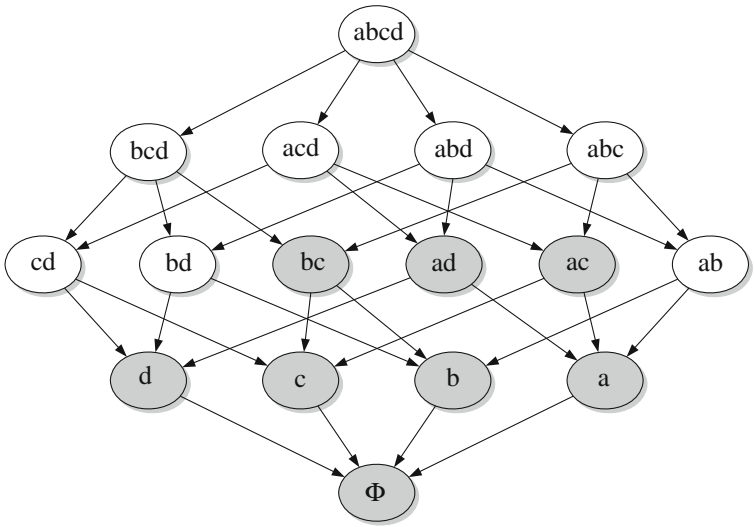


Fig. 8.6 Recoverable configurations

1. From Fig. 8.6 identify the minimal recoverable configurations.

2. Compute the coverage and the redundancy degrees. Is the system fail-operational with respect to the first fault? Configurations ab, bd, cd are respectively recovered by the optimal state feedbacks $\boldsymbol{u}_{ab} = \boldsymbol{K}_{ab}\boldsymbol{x}$, $\boldsymbol{u}_{bd} = \boldsymbol{K}_{bd}\boldsymbol{x}$ and $\boldsymbol{u}_{cd} = \boldsymbol{K}_{cd}\boldsymbol{x}$ where the feedback gains are given below and result in the cost matrices $\boldsymbol{W}^*_{ab}$, $\boldsymbol{W}^*_{bd}$, $\boldsymbol{W}^*_{cd}$ whose maximal eigenvalues are 18.53, 23.76 and 21.60:

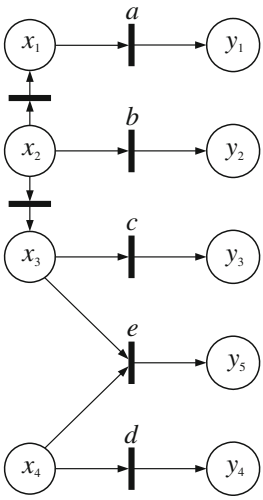


Fig. 8.7 Structural graph of the measurement system

$$\boldsymbol{K}_{ab} = \begin{pmatrix} -1.27 & -0.95 & -1.10 & -0.88 \\ -0.47 & -1.85 & -1.64 & -1.36 \\ -0.55 & -1.64 & -1.91 & -1.09 \\ -0.44 & -1.36 & -1.09 & -1.19 \end{pmatrix}$$

$$\boldsymbol{K}_{bd} = \begin{pmatrix} -3.18 & -2.18 & -2.32 & -2.41 \\ -1.09 & -1.88 & -1.82 & -1.49 \\ -1.16 & -1.82 & -2.23 & -1.28 \\ -1.20 & -1.49 & -1.28 & -1.63 \end{pmatrix}$$

$$\boldsymbol{K}_{cd} = \begin{pmatrix} -3.15 & -1.72 & -1.73 & -2.37 \\ -0.86 & -1.87 & -1.51 & -1.58 \\ -0.86 & -1.51 & -1.67 & -1.17 \\ -1.18 & -1.58 & -1.17 & -1.82 \end{pmatrix}$$

3. Let $\mathcal{U}$ be the reliable control bank that recovers all the recoverable configurations. List the control laws in $\mathcal{U}$. For each recoverable configuration list the control laws by which it is recovered. If several control laws allow to recover a given configuration, which one is to be selected?
4. Assume the control bank can implement only two control laws. What is the control law to be discarded? What is the influence on the coverage and the redundancy degrees? Is the system still fail-operational with respect to the first fault? □

Exercise 8.3 Sensor network design

Consider a measurement system with four unknown variables x_1, x_2, x_3, x_4 and five sensors a, b, c, d, e that provide five measurement signals y_1, y_2, y_3, y_4, y_5. Its structure graph is given by Fig. 8.7.

We are interested in the output-connection property (denoted $\mathcal{P}$), which is a very important structural property of sensor networks. A system is output-connected if there is a path in the

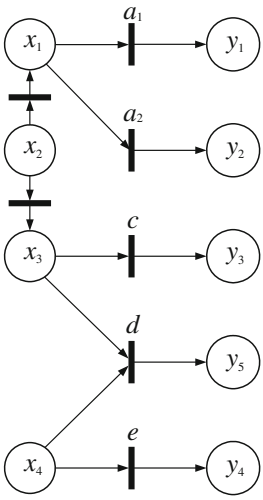


Fig. 8.8 The new system with b removed and a duplicated

structural graph from any unknown variable to a sensor (this is a necessary condition for the structural observability of the unknown variables). From Fig. 8.7, the system is clearly output-connected when the five sensors are used.

1. The lattice of system configurations allows to analyse the situations in which sensors are lost or removed from the sensor network. Determine whether property $\mathcal{P}$ holds or not for all the 4 sensor configurations (the configurations where one sensor is lost from the nominal configuration).
2. We now wish to determine whether the property holds or not for the sensor configurations where two sensors are lost. Do we need to analyse the subsets of $bcde$?
3. What is the output-connection span, what are its minimal configurations.
4. Compute the coverage, and the weak and strong redundancy degrees of the nominal configuration $abcde$. Is property $\mathcal{P}$ fail operational with respect to the first fault?
5. What are the critical sensor subsets.
6. What can be said about sensor b.
7. Note that the critical subset a is a singleton; therefore the probability to loose property $\mathcal{P}$ because of the loss of a is one order of magnitude larger than the probability to loose property $\mathcal{P}$ because of the loss of ce or de (assuming their failures are independent). Since b is useless, it might be interesting to remove sensor b from the sensor network and to duplicate sensor a. The new system a_1a_2cde is shown on Fig. 8.8.

   Go through questions 1–6 with the new system, and make comparisons. □

8.8 Bibliographical Notes

The fault-tolerant control problem. Defining the fault-tolerant control problem and understanding the differences with the classical control problem has motivated many early works [30, 32]. A formalisation of the problem can be found in [122, 329].

Recoverability is concerned with the possibility either to accommodate the faults or to reconfigure the system when faults occur. Early works on the recoverability problem are [113, 121, 170, 392] for a class of switched systems.

Recoverable faults can be handled by fault accommodation or system reconfiguration. A survey on fault accommodation is given in [264, 285], and interesting results can be found in [172, 327]. Many approaches have been developed to provide the model of the faulty system that is required by fault accommodation, most of them based on the development of adaptive or learning observers [171, 335, 336]. A control mixer approach to deal with actuator faults was pursued by [400, 401]. A wider area of reconfiguration was studied in [393, 399]. The general model of reconfiguration based fault tolerance was introduced in [330] and the use of generic models for reconfiguration analysis was considered in [331].

When faults are not recoverable, human intervention is most commonly needed to find another achievable system objective, using decision support from the diagnosis and overall goals for the plant [199]. Appropriate switching of the system operating mode is the goal of the supervisory system [169]. In fact, due to the discrete nature of fault occurrence and reconfiguration, fault-tolerant control systems are hybrid in nature according to [112, 113].

The properties of combined fault diagnosis and control were treated in [264].

Fault-tolerant linear quadratic design. The fault-tolerant linear quadratic design problem was introduced in [321] for actuator faults. Sensor faults and sensor network design were addressed in [149].

The model-predictive control technique allows to take into account inequality constraints, that are rather difficult to consider in linear quadratic control, at the price of an increased on-line computing power. This technique was used in [223] for fault accommodation and reconfiguration. The model-predictive controller uses all available input signals u_i and measurable output signals y_i which comprise the vectors $\boldsymbol{u}$ and $\boldsymbol{y}$ as before rather than only those input and output signals are used in the nominal feedback loop. If on the supervision level a fault f is detected, the inequality constraints included in the optimisation problem can be changed so that the model-predictive controller adapts to the faulty system. This can be done in a very easy way for actuator faults. If the diagnostic algorithm shows that the jth actuator is faulty, the equality constraint $u_j = 0$ is included in the optimisation problem in order to ensure that the controller does not really use the jth input. Then the model-predictive controller moves its control activity towards the available actuators, which can be interpreted as an on-line reconfiguration of the control loop.

As model-predictive control necessitates a rather large on-line computing capacity and as its reconfigurability property is, more or less, restricted to actuator faults this

method should be used for ensuring fault tolerance only if the advantages of model-predictive control have to be exploited for the faultless plant as well. For applications, where a fixed (linear) controller is sufficient for satisfying the control requirements for the faultless plant, the reconfiguration should be carried out by methods described in the earlier sections, which eventually result in a new fixed control law.

Implementation issues. The general theory of reconfiguration-based fault tolerance, including the passive–active design, was developed in [333]. The optimisation of the reliable control specifications was analysed in [332] while the reduction of the reliability over-cost was first presented in [16].

Fault-tolerance evaluation. Fault-tolerance evaluation is in some sense a measure of how many faults are or are not recoverable. It has been considered from the point of view of the system structural properties, e.g. observability or controllability, extending the evaluation of these properties to the faulty system. For example, the smallest second-order mode, first introduced in [231], has been proposed as a reconfigurability measure in [392]. A general approach to fault-tolerance evaluation under the reconfiguration strategy was presented in [68] with application to the measure of the system components' usefulness, and specification to the structural analysis approach.

Chapter 9
Fault Accommodation and Reconfiguration Methods

Abstract This chapter gives an overview of methods for re-adjusting the controller to faulty plants. Small faults can be tackled by fault accommodation, where the controller parameters are adapted to the parameters of the faulty plant. When accommodation cannot be used like in the case of an actuator or sensor breakdown, the control loop has to be reconfigured and a new control law designed.

9.1 Fault-Tolerant Model-Matching Design

9.1.1 Reconfiguration Problem

The basic scheme of fault-tolerant control is depicted in Fig. 1.1 on p. 2. At the execution level, a feedback controller

$$\boldsymbol{u}(t) = \boldsymbol{k}(\boldsymbol{y}(t),\ \boldsymbol{y}_{\mathrm{ref}}(t))$$

is used to attenuate the disturbance $\boldsymbol{d}$ and to ensure command tracking with respect to the command input $\boldsymbol{y}_{\mathrm{ref}}$. The control law $\boldsymbol{k}$ is designed so that the closed-loop system satisfies the given requirements for the faultless plant. Before a fault f occurs the supervision level shown in the figure only checks that the plant has its nominal behaviour.

If the diagnostic unit detects and identifies a fault, the adaptation of the controller to the faulty system is accomplished at the supervision level. This process results in new controller parameters and possibly in a new control configuration. If the sensors and actuators work differently as before but the faulty plant is still observable and controllable, the control configuration can remain as before but the controller parameters have to be adapted to the faulty system. This process is called fault accommodation.

However, if the sensor or actuator faults break the control loop, new sensors or actuators, respectively, have to be used. Then, the control loop has to be "reconfigured" in the sense that the whole process of selecting a suitable structure and

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8_9

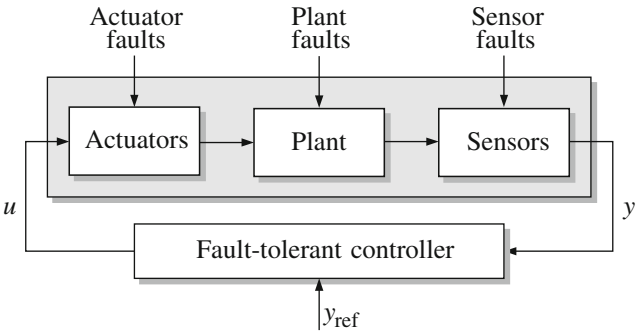


Fig. 9.1 Fault-tolerant controller

appropriate controller parameters has to be repeated after the fault is present. The control problem has to be considered "from the scratch" by appropriately choosing

- the signal vector $\boldsymbol{y}$ to be controlled and the input vector $\boldsymbol{u}$ to be used,
- the control law $\boldsymbol{k}$ including the controller parameters,
- the set-point $\boldsymbol{y}_{\text{ref}}$ of underlying control loops.

Control reconfiguration can be thought of as an "analytical repair" of the closed-loop system, where instead of repairing the plant the control algorithm and, hence, the controller software is changed while exploiting the redundant measurement or control signals for satisfying the control specifications in spite of the fault (Fig. 9.1).

To solve the fault-tolerant control problem, it is assumed that a state-space model

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{g}(\boldsymbol{x}(t),\ \boldsymbol{u}(t),\ f), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{9.1}$$

$$\boldsymbol{y}(t) = \boldsymbol{h}(\boldsymbol{x}(t),\ \boldsymbol{u}(t),\ f) \tag{9.2}$$

with state $\boldsymbol{x} \in |\mathcal{R}^n$, input $\boldsymbol{u} \in |\mathcal{R}^m$ and output $\boldsymbol{y} \in |\mathcal{R}^r$ is available, which also describes the dependence of the plant dynamics upon the faults $f \in \mathcal{F}$. Furthermore, it is assumed that a diagnostic algorithm has identified the current fault f.

According to these assumptions, the fault-tolerant control problem can be summarised as follows:

Problem 9.1 (*Fault-tolerant control problem*)

Given: Model (9.1), (9.2) of the plant
Nominal controller $\boldsymbol{k}$
Control specifications
Fault f

Find: Control configuration and new control law $\boldsymbol{k}_f$.

Note that in contrast to the usual controller design problem, also a nominal controller $\boldsymbol{k}$ is given. One of the important aspects of fault-tolerant control is to take advantage of the knowledge of the nominal controller $\boldsymbol{k}$ when solving the control problem stated above.

9.1.2 Pseudo-Inverse Method

One of the earliest methods for the controller redesign is based on model-matching. As the nominal closed-loop system is known, the model of this system can be used as a description of the dynamical properties that the new controller should produce in connection with the faulty plant. That is, the closed-loop system should match the model of the nominal loop.

The idea of model-matching is depicted in Fig. 9.2. The nominal closed-loop system is composed of the linear nominal plant

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \tag{9.3}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) \tag{9.4}$$

and a nominal controller, which is assumed to be a state feedback controller $\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{x}(t)$. Both components yield the model of the closed-loop system

$$\dot{\boldsymbol{x}}(t) = (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K})\,\boldsymbol{x}(t)$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t).$$

If the controller does not use all the inputs u_i of the input vector $\boldsymbol{u}$, the matrix $\boldsymbol{K}$ has zero rows, which is typical for plants with redundant actuators. When the fault f occurs, the faulty plant is given by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_f\boldsymbol{x}(t) + \boldsymbol{B}_f\boldsymbol{u}(t) \tag{9.5}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}_f\boldsymbol{x}(t), \tag{9.6}$$

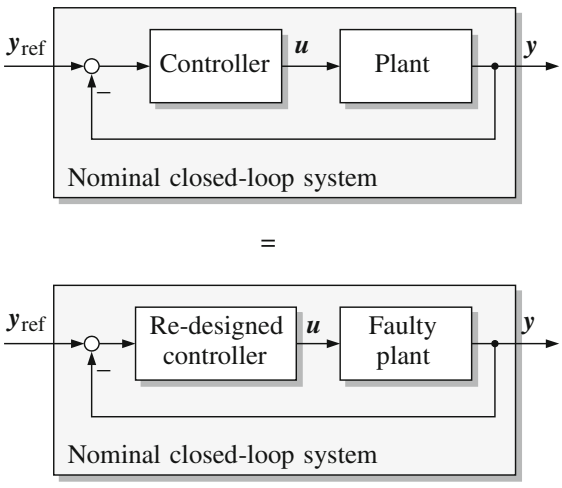


Fig. 9.2 Idea of the model-matching approach to control reconfiguration

where the fault f has changed the system properties, which are now described by the matrices $\boldsymbol{A}_f$, $\boldsymbol{B}_f$ and $\boldsymbol{C}_f$. If the sets of available input or output signals have changed, the matrices $\boldsymbol{B}_f$ and $\boldsymbol{C}_f$ have vanishing columns or rows, respectively. A new state feedback controller

$$\boldsymbol{u}(t) = -\boldsymbol{K}_f \boldsymbol{x}(t)$$

should be found such that the closed-loop system

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= (\boldsymbol{A}_f - \boldsymbol{B}_f \boldsymbol{K}_f)\, \boldsymbol{x}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}_f \boldsymbol{x}(t) \end{aligned}$$

behaves like the nominal loop. For the models used here, model-matching means to satisfy the relation

$$\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K} = \boldsymbol{A}_f - \boldsymbol{B}_f \boldsymbol{K}_f, \tag{9.7}$$

which means that both closed-loop systems have similar dynamics.

Equation (9.7) cannot be satisfied unless $\boldsymbol{B}$ and $\boldsymbol{B}_f$ have the same image (like in the case of a redundant actuator). Therefore, the new controller $\boldsymbol{K}_f$ is chosen so as to minimise the difference

$$\|(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}) - (\boldsymbol{A}_f - \boldsymbol{B}_f \boldsymbol{K}_f)\|. \tag{9.8}$$

The solution to this problem is given by

$$\boldsymbol{K}_f = \boldsymbol{B}_f^{+} \left(\boldsymbol{A}_f - \boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}\right) = \left(\boldsymbol{B}_f^{\mathrm{T}} \boldsymbol{B}_f\right)^{-1} \boldsymbol{B}_f^{\mathrm{T}} \left(\boldsymbol{A}_f - \boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}\right), \tag{9.9}$$

where $\boldsymbol{B}_f^{+}$ denotes the pseudoinverse of $\boldsymbol{B}_f$ given on the right-hand side of (9.9). Its use provides the reason for the name *pseudo-inverse method* of this approach.

The new controller (9.9) is adapted to the faulty system and minimises the difference (9.8) between the dynamical properties of the nominal loop and the closed-loop system with the faulty plant. Although the controller $\boldsymbol{K}_f$ is the best possible solution to the controller redesign problem, it does not ensure that the closed-loop system behaves satisfactorily. In particular, it does not ensure the stability of the closed-loop system. Therefore, the stability of $\boldsymbol{A}_f - \boldsymbol{B}_f \boldsymbol{K}_f$ and the performance of the control loop have to be evaluated separately. Extensions of this method ensure the stability without a separate test.

Fault accommodation and control reconfiguration. The method described so far is rather general. It includes both fault accommodation and control reconfiguration. Depending on the sensors and the actuators used, the controller is simply adapted to the new plant dynamics or it uses sensors or actuators that have not been used in the nominal case. In the latter case, vanishing rows in the nominal controller $\boldsymbol{K}$ are

replaced by non-zero elements, which means that new actuators are used and, hence, a new control configuration results.

9.1.3 Model-Matching Control for Sensor Failures

This section considers the case of complete sensor failures. If the ith sensor fails, the output y_i is set to zero. In the plant model the matrix $\boldsymbol{C}$ changes to $\boldsymbol{C}_f$, whose ith row is zero, but the other matrices remain the same as in the nominal case. The corresponding reconfiguration problem will be investigated here for output feedback

$$\boldsymbol{u}(t) = -\boldsymbol{K}\,\boldsymbol{y}(t),$$

for which the nominal closed-loop system is described by

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}\boldsymbol{C})\,\boldsymbol{x}(t)\\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t).\end{aligned}$$

For the faulty plant, the new controller

$$\boldsymbol{u}(t) = -\boldsymbol{K}_f\,\boldsymbol{y}_f(t)$$

should be found such that the closed loop

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}_f\,\boldsymbol{C}_f)\,\boldsymbol{x}(t)\\ \boldsymbol{y}_f(t) &= \boldsymbol{C}_f\,\boldsymbol{x}(t)\end{aligned}$$

has the same dynamics as the nominal loop.

The controller has to satisfy the simplified version of Eq. (9.7)

$$\boldsymbol{K}_f\,\boldsymbol{C}_f = \boldsymbol{K}\boldsymbol{C}. \tag{9.10}$$

To find an appropriate matrix $\boldsymbol{K}_f$ is possible only if the condition

$$\text{Kern}(\boldsymbol{C}_f) \subseteq \text{Kern}(\boldsymbol{C}) \tag{9.11}$$

is satisfied, where Kern denotes the kernel[1] of a matrix. The condition means that the measurement information obtained by the full output vector $\boldsymbol{y}$ is the same as the information obtained by the remaining sensors through $\boldsymbol{y}_f$. The condition (9.11) can be written in an equivalent form as

[1] The kernel of $\boldsymbol{C}$ is the set of vectors $\boldsymbol{x}$ for which $\boldsymbol{C}\boldsymbol{x} = \boldsymbol{0}$ holds.

$$\operatorname{rank} \boldsymbol{C}_f = \operatorname{rank} \begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{C}_f \end{pmatrix}.$$

Lemma 9.1 *In case of sensor failures, exact model-matching can be reached if the relation (9.11) holds. Then, the controller*

$$\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{P}\boldsymbol{y}(t) \tag{9.12}$$

solves the reconfiguration problem where

$$\boldsymbol{P} = \boldsymbol{C}\boldsymbol{C}_f^{+} = \boldsymbol{C}\boldsymbol{C}_f^{\mathrm{T}} \left(\boldsymbol{C}_f \boldsymbol{C}_f^{\mathrm{T}}\right)^{-1} \tag{9.13}$$

satisfies the relation

$$\boldsymbol{C} = \boldsymbol{P}\boldsymbol{C}_f. \tag{9.14}$$

The reconfigured controller $\boldsymbol{K}_f = \boldsymbol{K}\boldsymbol{P}$ produces a closed-loop system that has exactly the same properties as the faultless closed-loop system.

Situations where the requirement (9.11) is satisfied include the following:

- The fault has changed the sensitivity of the sensor, but the signal is not completely lost. Hence, $\boldsymbol{y}_f = a\,\boldsymbol{y}$ holds for some scalar a.
- A sensors is at fault which has at least one parallel redundant sensor. The matrix $\boldsymbol{P}$ switches the output to the redundant sensor.
- An analytic relation between the faulty output and several other output values exists, which can be reformulated by using the matrix $\boldsymbol{P}$.

The later two cases are only possible if $\boldsymbol{C}$ does not have full rank, which is likely in special applications only.

9.1.4 Model-Matching Control for Actuator Failures

In case of an actuator failure, the matrix $\boldsymbol{B}$ is replaced by the matrix $\boldsymbol{B}_f$ with zero column for the failing actuator. The output feedback

$$\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{y}(t)$$

which leads to the closed-loop system

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \left(\boldsymbol{A} - \boldsymbol{B}_f\,\boldsymbol{K}\boldsymbol{C}\right)\,\boldsymbol{x}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t). \end{aligned}$$

should be replaced by a new controller

$$\boldsymbol{u}_f(t) = -\boldsymbol{K}_f\, \boldsymbol{y}(t)$$

such that the closed loop

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= (\boldsymbol{A} - \boldsymbol{B}_f\, \boldsymbol{K}_f\, \boldsymbol{C})\, \boldsymbol{x}(t)\\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t)\end{aligned}$$

has the same dynamics as the nominal loop.

The controller has to satisfy the simplified version of Eq. (9.7)

$$\boldsymbol{B}_f\, \boldsymbol{K}_f = \boldsymbol{B}\boldsymbol{K}. \tag{9.15}$$

A solution $\boldsymbol{K}_f$ to this equation exists only if the condition

$$\text{Im}\,(\boldsymbol{B}_f) \supseteq \text{Im}\,(\boldsymbol{B}) \tag{9.16}$$

holds, where Im denotes the image[2] of a matrix. An equivalent formulation of the condition (9.16) is given by

$$\text{rank}\ \boldsymbol{B}_f = \text{rank}\ \begin{pmatrix}\boldsymbol{B} & \boldsymbol{B}_f\end{pmatrix}.$$

Lemma 9.2 *In case of actuator failures, exact model-matching is possible if Eq. (9.16) holds. Then, the reconfigured controller is given by*

$$\boldsymbol{u}(t) = -\boldsymbol{N}\boldsymbol{K}\, \boldsymbol{y}(t), \tag{9.17}$$

where

$$\boldsymbol{N} = \boldsymbol{B}_f^{+}\boldsymbol{B} = \left(\boldsymbol{B}_f^{\mathrm{T}}\boldsymbol{B}_f\right)^{-1}\boldsymbol{B}_f^{\mathrm{T}}\boldsymbol{B} \tag{9.18}$$

is a matrix satisfying the relation

$$\boldsymbol{B}_f\boldsymbol{N} = \boldsymbol{B}. \tag{9.19}$$

The new controller $\boldsymbol{K}_f = \boldsymbol{N}\boldsymbol{K}$ yields a closed-loop system with exactly the same properties as the nominal loop.

Example 9.1 Model-matching for actuator failures
This example demonstrates the model-matching approach for actuator failures and shows the main idea and a situation in which this approach fails.

[2]The image of $\boldsymbol{C}$ is the set of vectors $\boldsymbol{y}$, for which a vector $\boldsymbol{x}$ exists such that $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$ holds.

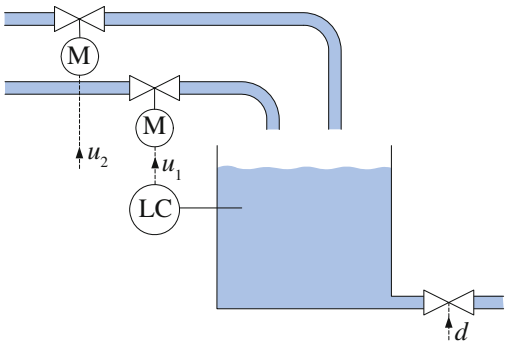


Fig. 9.3 Example demonstrating the model-matching reconfiguration strategy

Consider the tank system shown in Fig. 9.3 which has two input pipes. Obviously, for level control, only one pipe is necessary as control input and the redundant input can be used in case of an actuator failure.

Assume first, that the valve positions are used directly as the control inputs. Then the system can be described by a state-space model (9.3), (9.4) where the matrix

$$\boldsymbol{B} = (\boldsymbol{b} \;\; k\boldsymbol{b})$$

has two linearly depending columns because the two inputs influence the system in the same way and the effects of the two actuators distinguish only with respect to some constant factor k. In the nominal system, the first control input is used:

$$u_1(t) = u_C(t) = -\boldsymbol{K}\,\boldsymbol{y}(t)$$

for some controller $\boldsymbol{K}$ and some output $\boldsymbol{y}$ of the tank system.

If the corresponding actuator fails, the controller should be switched to the second input, where

$$\boldsymbol{B}_f = (\boldsymbol{0} \;\; k\boldsymbol{b})$$

holds. The model-matching solutions yields the (2, 1)-element of the matrix $\boldsymbol{N}$

$$\boldsymbol{N}_{21} = (k^2\boldsymbol{b}^{\mathrm{T}}\boldsymbol{b})^{-1}k\boldsymbol{b}^{\mathrm{T}}\boldsymbol{b} = \frac{1}{k},$$

which means that the output $u_C(t)$ of the nominal controller is transformed into the input

$$u_2(t) = \frac{1}{k}u_C(t)$$

to the second actuator. This is an obvious solution: As the gain of the new actuator is k-times the gain of the old one, the old input u_C is multiplied by $\frac{1}{k}$. A perfect reconfiguration results.

Now change the situation by including the motors for the valves as shown in Fig. 9.3. As these motors have integral dynamics, two additional states have to be added to the state

$$\tilde{\boldsymbol{x}} = \begin{pmatrix} x_{a1} \\ x_{a2} \\ \boldsymbol{x} \end{pmatrix}$$

such that the model now reads as

$$\begin{aligned} \dot{\tilde{\boldsymbol{x}}}(t) &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \boldsymbol{b} & k\boldsymbol{b} & \boldsymbol{A} \end{pmatrix} \tilde{\boldsymbol{x}}(t) + \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix} \\ \boldsymbol{y}(t) &= (\boldsymbol{O} \quad \boldsymbol{O} \quad \boldsymbol{C})\, \tilde{\boldsymbol{x}}(t) \end{aligned}$$

In principle, the same solution as before is possible. However, the model-matching approach yields for

$$\boldsymbol{B} = \begin{pmatrix} 1 \\ 0 \\ \boldsymbol{0} \end{pmatrix} \quad \text{and} \quad \boldsymbol{B}_f = \begin{pmatrix} 0 \\ 1 \\ \boldsymbol{0} \end{pmatrix}$$

the solution

$$\boldsymbol{N}_{21} = \left(\begin{pmatrix} 0 \\ 1 \\ \boldsymbol{0} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} 0 \\ 1 \\ \boldsymbol{0} \end{pmatrix} \right)^{-1} \begin{pmatrix} 0 \\ 1 \\ \boldsymbol{0} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} 1 \\ 0 \\ \boldsymbol{0} \end{pmatrix} = \boldsymbol{O},$$

where the pseudo-inverse matrix has been built after the zero columns for the no longer available inputs have been deleted. Hence, there is no control input at all. The model-matching approach fails.

The reason for this result lies in the fact that the model-matching idea tries to reproduce the effect $\boldsymbol{B}u$ of the nominal controller by the reconfigured controller $\boldsymbol{B}_f\boldsymbol{N}u$. This is impossible in this example, because the nominal controller has a direct effect only on the state variable x_{a1} an no effect at all on the state variable x_{a2} whereas the redundant input leads to the reverse situation. Hence, no choice of $\boldsymbol{N}$ can reproduce any of the effects of the nominal input. The failure of the model-matching approach lies in this idea and can be circumvented by extending the model-matching aim to the whole plant as described below. □

9.1.5 Markov Parameter Approach to Control Reconfiguration for Actuator Failures

The model-matching approach using the pseudo-inverse of the input matrix fails because if concentrates on the forcing action at point Ⓟ in Fig. 9.4. In the approach shown in this section, the goal refers to the I/O-behaviour of the plant. By this formulation, analytical redundancies become amenable which are based on internal couplings via the system matrix on the one hand and the selection of relevant states via the output matrix on the other hand, see point Ⓠ in the figure. Such redundancies are hidden from a forcing action perspective.

The *Markov parameters*

$$\boldsymbol{G}_i = \boldsymbol{C}\boldsymbol{A}^{i-1}\boldsymbol{B}, \quad i = 1, \ldots, n \tag{9.20}$$

completely describe the I/O-behaviour of a linear system (9.3), (9.4) in terms of its transfer function

$$\boldsymbol{P}(s) = \sum_{i=0}^{\infty} \boldsymbol{G}_i s^{-i}. \tag{9.21}$$

The Markov parameter-based approach to control reconfiguration tries to recover the nominal plant Markov parameters after an actuator failure by using the static reconfiguration block

$$\boldsymbol{u}_\mathrm{c}(t) = \boldsymbol{N}\boldsymbol{u}_f(t). \tag{9.22}$$

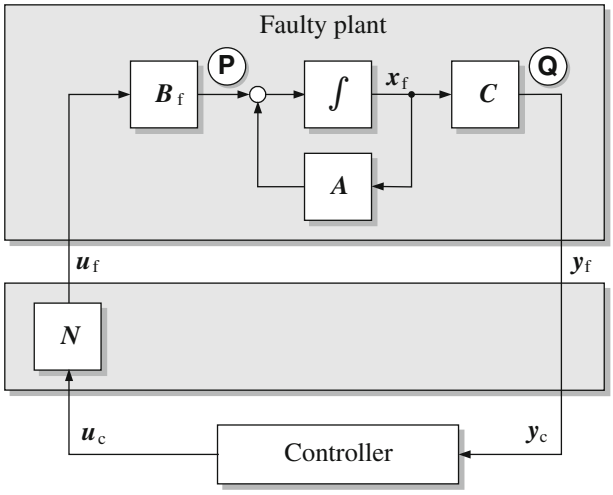


Fig. 9.4 Input/output-based reconfiguration after actuator failures

If the Markov parameters of a reconfigured plant match those of the nominal plant (9.3), (9.4) exactly, the dynamical I/O-behaviour is recovered exactly, which is both necessary and sufficient for successful static I/O-reconfiguration.

With the observability matrix

$$\boldsymbol{S}_O = \begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{C}\boldsymbol{A} \\ \vdots \\ \boldsymbol{C}\boldsymbol{A}^{n-1} \end{pmatrix} \in |\mathcal{R}^{n\cdot m\times n} \tag{9.23}$$

the design problem to Markov parameter recovery can be posed as

$$\boldsymbol{N} = \arg\min_{\boldsymbol{N}} \|\boldsymbol{S}_O \boldsymbol{B}_f \boldsymbol{N} - \boldsymbol{S}_O \boldsymbol{B}\| \tag{9.24}$$

with the solution

$$\boldsymbol{N} = (\boldsymbol{S}_O \boldsymbol{B}_f)^+ \boldsymbol{S}_O \boldsymbol{B}. \tag{9.25}$$

If the condition

$$\text{Im}\,(\boldsymbol{S}_O \boldsymbol{B}_f) \supseteq \text{Im}\,(\boldsymbol{S}_O \boldsymbol{B}) \tag{9.26}$$

holds, then perfect I/O-reconfiguration results in the sense that all Markov parameters are exactly recovered. This condition is equivalent to

$$\text{rank}\,(\boldsymbol{S}_O \boldsymbol{B}_f) = \text{rank}\,(\boldsymbol{S}_O \boldsymbol{B}_f \quad \boldsymbol{S}_O \boldsymbol{B}). \tag{9.27}$$

If this condition is violated, an approximate solution is obtained in this way which matches the original Markov parameters as closely as possible.

Lemma 9.3 *In case of actuator failures, exact model-matching with respect to the I/O-behaviour can be reached if the condition (9.26) holds. Then the reconfigured controller is given by*

$$\boldsymbol{u}(t) = -\boldsymbol{N}\boldsymbol{K}\boldsymbol{y}(t), \tag{9.28}$$

where

$$\boldsymbol{N} = (\boldsymbol{S}_O \boldsymbol{B}_f)^+ \boldsymbol{S}_O \boldsymbol{B} \tag{9.29}$$

is a matrix satisfying the relation

$$\boldsymbol{C}\boldsymbol{A}^{(i-1)} \boldsymbol{B}_f \boldsymbol{N} = \boldsymbol{C}\boldsymbol{A}^{(i-1)} \boldsymbol{B}, \quad i = 1, \ldots, n. \tag{9.30}$$

The new controller yields a closed-loop system with exactly the same I/O-behaviour as the nominal loop.

Remark 9.1 (*Generality of the method*) The approach is valid in connection with any controller, since the plant I/O-response is recovered and the fault is hidden from the controller. If the nominal loop was internally stable, this property is preserved under reconfiguration if condition (9.26) holds, as an analysis using the Kalman decomposition reveals. □

Example 9.1 (cont.) Model-matching for actuator failures: Markov approach
The example is now solved using the Markov parameter approach. It is shown that the problems of the pseudo-inverse method are overcome.

The construction of the observability matrix (9.23) yields

$$\boldsymbol{S}_O \boldsymbol{B} = (\gamma \;\; k\gamma)\boldsymbol{b} \;\text{ with }\; \gamma = (\boldsymbol{C} \;\; \boldsymbol{C}\boldsymbol{A}\ldots)^{\mathrm{T}}, \tag{9.31}$$

whereas after the fault the relation

$$\boldsymbol{S}_O \boldsymbol{B}_f = (\boldsymbol{0} \;\; k\gamma)\boldsymbol{b} \tag{9.32}$$

holds. Condition (9.26) is met and the admissible solution to the problem

$$\boldsymbol{S}_O \boldsymbol{B}_f \boldsymbol{N} = \boldsymbol{S}_O \boldsymbol{B} \tag{9.33}$$

is found using Eq. (9.25) as

$$\boldsymbol{N} = \begin{pmatrix} 0 & 0 \\ \frac{1}{k} & 1 \end{pmatrix}. \tag{9.34}$$

As expected, the control input meant for the first valve is redirected to the second valve with the correct gain adjustment. □

Example 9.2 Markov parameter approach applied to the two-tank example
The plant consists of the two tanks T_1 and T_2 interconnected by valves u_L, u_H, where T_1 is filled via pump u_P as shown in Fig. 9.5. Valves are electromechanically driven with the motor states v_L, v_H. The controlled quantities are the levels h_1 and h_2. With the state $\boldsymbol{x} = (v_L, \; v_H, \; h_1, \; h_2)^{\mathrm{T}}$, the tank system is described by the linear model (9.3), (9.4) with

$$\boldsymbol{A} = 10^3 \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -3.2 & -3.4 & -7.1 & 3.6 \\ 3.2 & 3.4 & 7.1 & -18 \end{pmatrix}$$

$$\boldsymbol{B} = 10^3 \begin{pmatrix} 0 & 10^{-3} & 0 \\ 0 & 0 & 10^{-3} \\ 8.1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \qquad \boldsymbol{B}_f = 10^3 \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 10^{-3} \\ 8.1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{C} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and controlled by two decentralised proportional controllers *LC*.

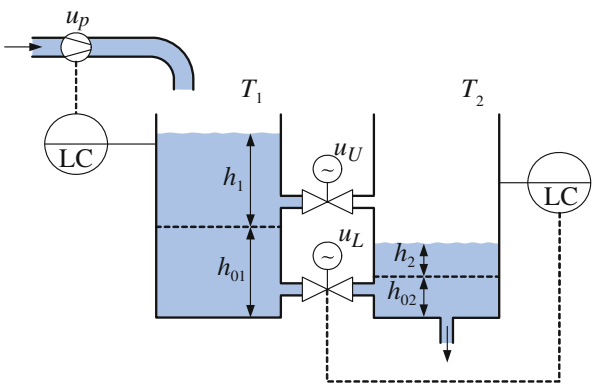


Fig. 9.5 Reconfiguration of a two-tank system

After a blocking of the lower valve at fault time t_f, which yields $u_L(t) = 0$ for $t \geq t_f$, the plant is statically I/O-reconfigurable according to the condition (9.27). The reconfiguration (9.29) yields

$$\boldsymbol{N} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0.9167 & 1 \end{pmatrix}.$$

The behaviour of the successfully reconfigured plant with fault f occurring at $t_f = 250\,\mathrm{s}$ is shown in Fig. 9.6. After the fault appearing at $t_f = 250\,\mathrm{s}$ and the reconfiguration accomplished at $t = 260\,\mathrm{s}$, the control action is redirected from the lower to the upper valve. This action appears logical, but it cannot be found by the pseudo-inverse method. □

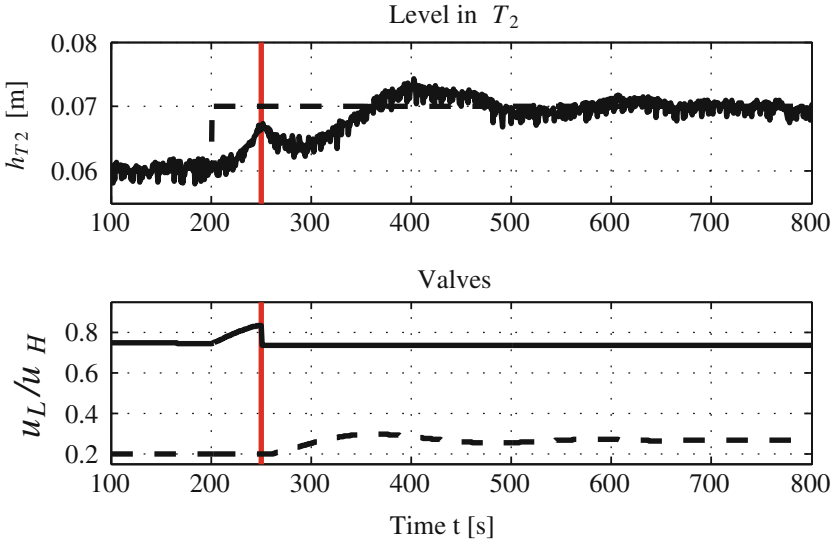


Fig. 9.6 Experimental results with the reconfigured tank system: After the failure of the lower valve (u_L, *solid line*) the controller acts at the upper valve (u_H, *dashed line*)

9.2 Control Reconfiguration for Actuator or Sensor Failures

9.2.1 The Idea of Virtual Sensors and Virtual Actuators

Severe faults such as the complete failure of actuators or sensors open the control loop with the nominal controller. In order to hold the system in operation, it is necessary to use a different set of input or output signals to accomplish the control task. Once the new control configuration is selected, new controller parameters have to be found. The goal of the reconfiguration is to stabilise the faulty process and to keep it operational with sufficient performance.

Figure 9.7 shows the main idea of the methods explained in this section. Instead of adapting the controller to the faulty plant, a reconfiguration block is used to adapt the faulty plant to the nominal controller. The faulty plant together with the reconfiguration block should produce, for a given input $\boldsymbol{u}_\mathrm{c}$, the same (or approximately the same) output $\boldsymbol{y}_\mathrm{c}$ as the nominal plant. Hence, the controller "sees" the same plant as before and reacts in the same way as before.

This solution of the reconfiguration problem tries to apply a minimal change to the control loop. In particular, the nominal controller remains an unchanged block of the control loop. The rationale for keeping the controller as before is given by the fact that the existing control law includes valuable implicit knowledge about the process and the possible performance of the closed-loop system. This knowledge was acquired during the design cycle and is not represented in the process model. For example, during the design it became obvious, which control objectives (like overshoot, bandwidth, settling time) can be met with reasonable control effort and which not. The trade-off between the different control objectives is represented by the nominal controller.

In case of a sensor breakdown, the reconfiguration block results from the application of a Luenberger observer to reconstruct the immeasurable output. It is called

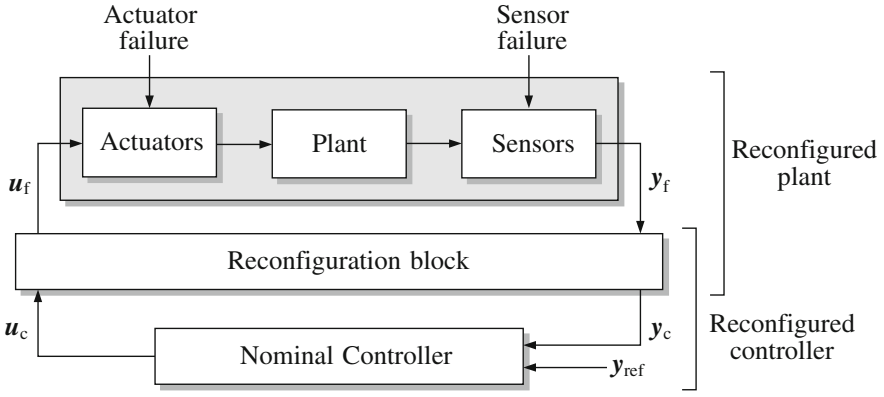


Fig. 9.7 Principle of control reconfiguration for actuator or sensor failures

a "virtual sensor", because it reconstructs that element y_i of the output vector $\boldsymbol{y}_c$ from the other measured output signals that the faulty sensor does no longer measure. If an actuator becomes faulty, the reconfiguration block is obtained in a dual way. The reconfiguration block is called a "virtual actuator", because it acts like the faulty actuator but replaces the effect of this actuator by using the control input of the other actuators appropriately. The reconfigured controller, which is to be applied to the faulty plant, consists of the nominal controller and the reconfiguration block (Fig. 9.7).

The way to find appropriate reconfiguration blocks, which will be described in this section, uses an alternative interpretation of the reconfigured control loop: The faulty process and the reconfiguration block together are called the reconfigured plant, which is connected to the nominal controller. If the reconfigured plant behaves like the nominal plant, the loop consisting of the reconfigured plant and the controller behaves like the nominal closed-loop system. This is true for an arbitrary nominal controller.

Example 9.3 Two-tank reconfiguration problem

The reconfiguration problem and a way of its solution are illustrated by the two coupled tanks depicted in Fig. 9.8.

The main mission of the system is to store water at a certain level in the right tank for some consumer. During the nominal operation there exists two level controllers, with the set-points $y_{1\mathrm{ref}}$ and $y_{2\mathrm{ref}}$. The right controller uses the upper valve, whose position is given by the input u_2. A redundant control input is provided by the lower valve with input signal u_3. In the nominal case, the valve V_{12} is closed. The right controller has to attenuate the disturbance d and to hold the tank level at a given value $y_{2\mathrm{ref}}$. The control specifications include the stability, the set-point following requirement and the specification that the command step response should not have a large overshoot.

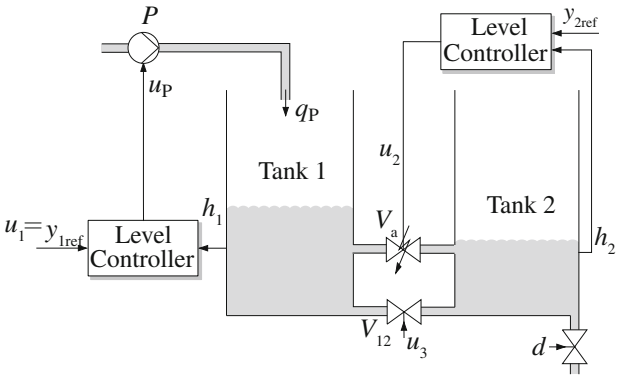


Fig. 9.8 Reconfiguration problem for the tank example

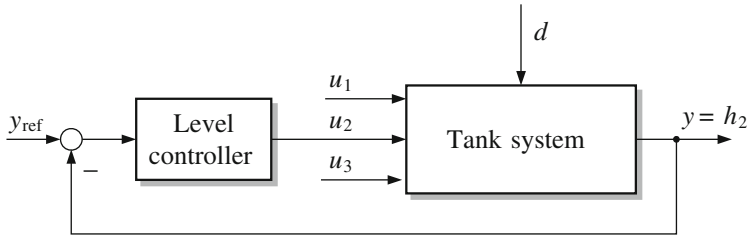


Fig. 9.9 Block diagram of the reconfiguration problem

For the reconfiguration problem, three actuator faults are considered:

- Valve V_a is closed and blocked.
- Valve V_a is open and blocked.
- A level sensor is faulty.

In these cases, one of the two control loops does no longer work. The reconfiguration task consists in finding a new control structure by selecting appropriate actuators, new control laws and new set-points for the control loops such that the control aims described above are obtained (Fig. 9.9).

Obviously, the reconfiguration task cannot be solved by simply changing the parameters of the given controllers, but a structural change of the control configuration is necessary:

- If Valve V_a is closed and blocked, the level controller of the right tank has to use the lower valve V_{12} as control input. In this case, the controller of the left tank can remain unchanged.
- If Valve V_a is open and blocked, in addition to the change of the level controller of the right tank as before, the set-point of the level controller of the left tank has to be set to a value which is lower then the position of Valve V_a. Another possibility is to use the set-point of the level controller of the left tank as control input of the level controller of the right tank.
- In case of the sensor fault, the missing sensor reading has to be reconstructed by means of the remaining output measurements.

All these solutions, which for this simple example seem to be obvious, have to be found automatically by a fault-tolerant control algorithm. □

9.2.2 Reconfiguration Problem

Before explaining the reconfiguration method, the problem to be solved is formally stated. The model of the nominal process is given in state-space form:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{E}\boldsymbol{d}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{9.35}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t). \tag{9.36}$$

The standard model is extended by the disturbance $\boldsymbol{d} \in |\mathcal{R}^p$.

It is important that the process model includes all available input and output signals including those that are not used by the nominal controller. Unlike in the traditional design problem, $\boldsymbol{B}$ and $\boldsymbol{C}$ may not have full rank.

The nominal process is stabilised by a nominal controller with output $\boldsymbol{u}(t)$ and inputs $\boldsymbol{y}(t)$ and $\boldsymbol{y}_{\mathrm{ref}}(t)$. The reconfiguration method explained here can be applied without further assumptions on the controller, which may have arbitrary dynamics and even be nonlinear. However, to demonstrate the properties of the resulting control loop a linear feedback controller

$$\boldsymbol{u}_{\mathrm{c}}(t) = -\boldsymbol{K}\,\boldsymbol{y}_{\mathrm{c}}(t) + \boldsymbol{V}\,\boldsymbol{y}_{\mathrm{ref}}(t) \tag{9.37}$$

is used.

Process and controller form the nominal control loop for $\boldsymbol{u}(t) = \boldsymbol{u}_{\mathrm{c}}(t)$ and $\boldsymbol{y}(t) = \boldsymbol{y}_{\mathrm{c}}(t)$:

$$\dot{\boldsymbol{x}}(t) = (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}\boldsymbol{C})\,\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{V}\,\boldsymbol{y}_{\mathrm{ref}}(t) + \boldsymbol{E}\boldsymbol{d}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{9.38}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t). \tag{9.39}$$

This control loop is assumed to be stable and to satisfy the performance requirements concerning set-point tracking and disturbance rejection.

Fault cases. In the case that the fault f indicates a loss of sensor i, the ith row of the matrix $\boldsymbol{C}$ is changed into zeros and the new matrix is denoted by $\boldsymbol{C}_f$. If the jth actuator fails, the jth column of the matrix $\boldsymbol{B}$ is set to zero and the resulting matrix denoted by $\boldsymbol{B}_f$. In this way, the number of input signals, output signals and state variables is not changed in the model, though some of them may have lost their function. It is assumed that the faulty process is still controllable and observable. This assumption implies that a stabilising controller exists. The input and the output of the faulty plant are denoted by $\boldsymbol{u}_f$ or $\boldsymbol{y}_f$, respectively.

Reconfiguration task. The aim is to find a reconfigured controller that makes the closed-loop system satisfy the following conditions, which, depending on the control task, refer to the autonomous behaviour, reference tracking and disturbance rejection:

- **Strong reconfiguration goal**:
  The controller should make the reconfigured control loop behave in exactly the same way as the nominal control loop, i.e. the relation

$$\boldsymbol{y}_f(t) = \boldsymbol{y}(t)$$

  should hold for any $\boldsymbol{d}(t)$, $\boldsymbol{y}_{\mathrm{ref}}(t)$ and $\boldsymbol{x}_0$.

It will be demonstrated that this strong goal is only feasible in very special cases. Therefore, a weaker goal is defined in terms of the dynamical and the static behaviour of the reconfigured loop.

- **Weak reconfiguration goal**:
  The weak goal consists of a static and a dynamical part. Considering the static behaviour, the output $\boldsymbol{y}_f$ of the reconfigured loop should have the same value as for the nominal system. This means that for constant values of $\boldsymbol{y}_{\text{ref}}$ and $\boldsymbol{d}$, the relation

$$\boldsymbol{y}_f(t) \to \boldsymbol{y}(t) \text{ for } t \to \infty$$

  should hold. The transient behaviour is determined by the poles and zeros of the system which should not differ significantly in the nominal and the reconfigured control loop. This requirement applies for the autonomous, the disturbance, and the command following behaviour of the reconfigured loop. Additional poles (and zeros) are allowed only if they are fast enough not to dominate the system behaviour.

9.2.3 Virtual Sensor

This section describes a reconfiguration block that reconstructs a measurement y_i from the remaining sensor signals after the ith sensor is no longer available. The main idea is to use an observer for the faulty system, which represents the main part of the reconfiguration block to be built. This block is called *virtual sensor* due to its function of replacing a broken sensor.

The plant with faulty sensor is described by the state-space model

$$\dot{\boldsymbol{x}}_f(t) = \boldsymbol{A}\boldsymbol{x}_f(t) + \boldsymbol{B}\boldsymbol{u}_f(t) + \boldsymbol{E}\boldsymbol{d}(t), \quad \boldsymbol{x}_f(0) = \boldsymbol{x}_{\text{f0}} \tag{9.40}$$

$$\boldsymbol{y}_f(t) = \boldsymbol{C}_f\boldsymbol{x}_f(t), \tag{9.41}$$

where the sensor failure is reflected by the matrix $\boldsymbol{C}_f$. If the condition (9.11) is satisfied, the complete output vector $\boldsymbol{y}$ can be reconstructed from $\boldsymbol{y}_f$ and the reconfigured controller (9.12) can be used. This new control structure can be interpreted as consisting of a reconfiguration block

$$\begin{aligned} \boldsymbol{y}_{\text{c}}(t) &= \boldsymbol{P}\boldsymbol{y}_f(t) + \boldsymbol{y}_\Delta \\ \boldsymbol{u}_f(t) &= \boldsymbol{u}_{\text{c}}(t) \end{aligned}$$

and the nominal controller. That is, under the condition (9.11) the virtual sensor is a static reconfiguration block.

In the following, the general case is considered, where the condition (9.11) is violated. Then, the reconfiguration block includes a state observer and a direct feedthrough:

Definition 9.1 (*Virtual sensor*) Consider the plant (9.40), (9.41) with faulty sensor. The virtual sensor is defined as the system

$$\dot{\boldsymbol{x}}_{\mathrm{V}}(t) = \boldsymbol{A}_{\mathrm{V}}\boldsymbol{x}_{\mathrm{V}}(t) + \boldsymbol{B}_{\mathrm{V}}\boldsymbol{u}_{\mathrm{c}}(t) + \boldsymbol{L}\boldsymbol{y}_f(t), \quad \boldsymbol{x}_{\mathrm{V}}(0) = \boldsymbol{x}_{\mathrm{V0}} \tag{9.42}$$

$$\boldsymbol{u}_f(t) = \boldsymbol{u}_{\mathrm{c}}(t) \tag{9.43}$$

$$\boldsymbol{y}_{\mathrm{c}}(t) = \boldsymbol{C}_{\Delta}\boldsymbol{x}_{\mathrm{V}}(t) + \boldsymbol{P}\boldsymbol{y}_f(t) \tag{9.44}$$

with the state $\boldsymbol{x}_{\mathrm{V}} \in |\mathcal{R}^n$ and with matrices

$$\boldsymbol{A}_{\mathrm{V}} = \boldsymbol{A} - \boldsymbol{L}\boldsymbol{C}_f \tag{9.45}$$

$$\boldsymbol{B}_{\mathrm{V}} = \boldsymbol{B} \tag{9.46}$$

$$\boldsymbol{C}_{\mathrm{V}} = \boldsymbol{C} - \boldsymbol{P}\boldsymbol{C}_f. \tag{9.47}$$

$\boldsymbol{P}$ and $\boldsymbol{L}$ denote matrices that can be freely chosen.

The main part of the virtual sensor is the state observer with the state vector $\boldsymbol{x}_{\mathrm{V}}(t)$. The complete output $\boldsymbol{y}_{\mathrm{c}}(t)$ of the plant can be approximately determined: $\boldsymbol{y}_{\mathrm{c}}(t) \approx \boldsymbol{C}\boldsymbol{x}_{\mathrm{V}}(t)$. This observation result is improved by using the available sensor values and by observing only the difference between the nominal and the faulty output. In a generalised form, this approach is represented by Eq. (9.44), where the matrix $\boldsymbol{P}$ is a design parameter. For $\boldsymbol{P} = \boldsymbol{O}$ only observed values are used.

Model of the reconfigured plant. The plant together with the virtual sensor is described by Eqs. (9.40)–(9.47):

$$\begin{pmatrix} \dot{\boldsymbol{x}}_f(t) \\ \dot{\boldsymbol{x}}_{\mathrm{V}}(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{O} \\ \boldsymbol{L}\boldsymbol{C}_f & \boldsymbol{A} - \boldsymbol{L}\boldsymbol{C}_f \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_{\mathrm{V}}(t) \end{pmatrix} + \begin{pmatrix} \boldsymbol{B} \\ \boldsymbol{B} \end{pmatrix} \boldsymbol{u}_{\mathrm{c}}(t) + \begin{pmatrix} \boldsymbol{E} \\ \boldsymbol{O} \end{pmatrix} \boldsymbol{d}(t) \tag{9.48}$$

$$\boldsymbol{y}_{\mathrm{c}}(t) = \begin{pmatrix} \boldsymbol{P}\boldsymbol{C}_f & \boldsymbol{C}_{\mathrm{V}} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_{\mathrm{V}}(t) \end{pmatrix}. \tag{9.49}$$

A state transformation is performed in order to introduce the observation error $\boldsymbol{x}_{\Delta}(t) = \boldsymbol{x}_{\mathrm{V}}(t) - \boldsymbol{x}_f(t)$: Eqs. (9.48), (9.49) are equivalent to

$$\begin{pmatrix} \dot{\boldsymbol{x}}_f(t) \\ \dot{\boldsymbol{x}}_{\Delta}(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{A} - \boldsymbol{L}\boldsymbol{C}_f \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_{\Delta}(t) \end{pmatrix} + \begin{pmatrix} \boldsymbol{B} \\ \boldsymbol{O} \end{pmatrix} \boldsymbol{u}_{\mathrm{c}}(t) + \begin{pmatrix} \boldsymbol{E} \\ -\boldsymbol{E} \end{pmatrix} \boldsymbol{d}(t) \tag{9.50}$$

$$\boldsymbol{y}_{\mathrm{c}}(t) = \begin{pmatrix} \boldsymbol{C} & \boldsymbol{C}_{\mathrm{V}} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_{\Delta}(t) \end{pmatrix} \tag{9.51}$$

$$\begin{pmatrix} \boldsymbol{x}_f(0) \\ \boldsymbol{x}_{\Delta}(0) \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_{\mathrm{f0}} \\ \boldsymbol{x}_{\mathrm{V0}} - \boldsymbol{x}_{\mathrm{f0}} \end{pmatrix}.$$

Model of the reconfigured loop. For the analysis of the closed-loop behaviour the model of the reconfigured plant is combined with the linear feedback controller (9.37):

$$\begin{pmatrix} \dot{\boldsymbol{x}}_f(t) \\ \dot{\boldsymbol{x}}_\Delta(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}-\boldsymbol{BKC} & -\boldsymbol{BKC}_\mathrm{V} \\ \boldsymbol{O} & \boldsymbol{A}-\boldsymbol{LC}_f \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_\Delta(t) \end{pmatrix} + \\ + \begin{pmatrix} \boldsymbol{E} \\ -\boldsymbol{E} \end{pmatrix} \boldsymbol{d}(t) + \begin{pmatrix} \boldsymbol{BV} \\ \boldsymbol{O} \end{pmatrix} \boldsymbol{y}_\mathrm{ref}(t) \tag{9.52}$$

$$\boldsymbol{y}_f(t) = \begin{pmatrix} \boldsymbol{C}_f & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_\Delta(t) \end{pmatrix}. \tag{9.53}$$

The trajectory of this system depends on the initial state, the reference input $\boldsymbol{y}_\mathrm{ref}$ and the disturbance $\boldsymbol{d}$ (Fig. 9.10). As the system is linear, the behaviour can be analysed separately for these three excitations.

Autonomous behaviour. For $\boldsymbol{y}_\mathrm{ref}(t) = \boldsymbol{0}$ and $\boldsymbol{d}(t) = \boldsymbol{0}$ the system (9.52), (9.53) simplifies to

$$\begin{pmatrix} \dot{\boldsymbol{x}}_f(t) \\ \dot{\boldsymbol{x}}_\Delta(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}-\boldsymbol{BKC} & -\boldsymbol{BKC}_\mathrm{V} \\ \boldsymbol{O} & \boldsymbol{A}-\boldsymbol{LC}_f \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_\Delta(t) \end{pmatrix} \tag{9.54}$$

$$\boldsymbol{y}_f(t) = \begin{pmatrix} \boldsymbol{C}_f & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_\Delta(t) \end{pmatrix} \tag{9.55}$$

$$\begin{pmatrix} \boldsymbol{x}_f(0) \\ \boldsymbol{x}_\Delta(0) \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_\mathrm{f0} \\ \boldsymbol{x}_\mathrm{V0} - \boldsymbol{x}_\mathrm{f0} \end{pmatrix}.$$

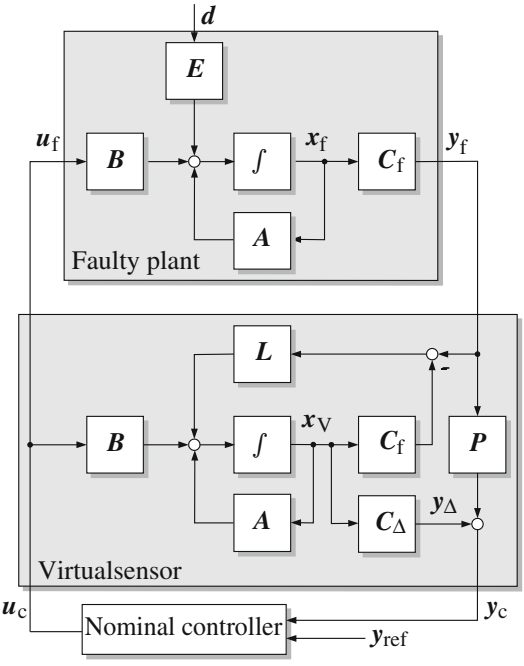


Fig. 9.10 Reconfiguration by using a virtual sensor

The separation principle of state observers applies: The matrix $\boldsymbol{K}$ influences the behaviour of the process state $\boldsymbol{x}_f(t)$ through the submatrix $\boldsymbol{A} - \boldsymbol{BKC}$ (controller design), while $\boldsymbol{L}$ affects the behaviour of the observation error $\boldsymbol{x}_\Delta$ through the submatrix $\boldsymbol{A} - \boldsymbol{LC}_f$ (observer design). There are cross-couplings in one direction only from $\boldsymbol{x}_\Delta(t)$ to $\boldsymbol{x}_f(t)$. The strength of the couplings and the influence of $\boldsymbol{x}_\Delta(t)$ on the output can be reduced by a suitable choice of the matrix $\boldsymbol{P}$.

Theorem 9.1 (Separation principle for the virtual sensor) *The set σ of eigenvalues of the reconfigured closed-loop system (9.54), (9.55) consists of the set of eigenvalues of the nominal closed-loop system (9.38), (9.39) and the set of eigenvalues of the virtual sensor (9.42):*

$$\sigma = \sigma\{\boldsymbol{A} - \boldsymbol{BKC}\} \cup \sigma\{\boldsymbol{A} - \boldsymbol{LC}_f\}.$$

The stability of the closed-loop is guaranteed if the nominal control loop is stable (depending on $\boldsymbol{K}$) and if the observer is stable (depending on $\boldsymbol{L}$). The second condition can be satisfied by an appropriate choice of $\boldsymbol{L}$ because the pair $(\boldsymbol{A}, \boldsymbol{C}_f)$ is assumed to be observable. The equilibrium state is zero for both the faulty and the nominal system.

Tracking behaviour. For $\boldsymbol{x}_{\mathrm{f0}} = \boldsymbol{x}_{\mathrm{V0}} = \boldsymbol{0}$ and $\boldsymbol{d} = \boldsymbol{0}$, the system (9.52), (9.53) simplifies to

$$\begin{aligned}\dot{\boldsymbol{x}}_f(t) &= (\boldsymbol{A} - \boldsymbol{BKC})\,\boldsymbol{x}_f(t) + \boldsymbol{BV}\,\boldsymbol{y}_{\mathrm{ref}}(t), \quad \boldsymbol{x}_f(0) = \boldsymbol{O}\\ \boldsymbol{y}_f(t) &= \boldsymbol{C}_f\boldsymbol{x}_f(t),\end{aligned}$$

which is identical to the behaviour of the nominal closed-loop system (9.38), (9.39). Hence, the reference tracking behaviour of the reconfigured control loop is identical to that of the nominal control loop.

Disturbance behaviour. For the disturbance behaviour it is assumed that the initial state and the reference input are zero. This leads to the following closed-loop system:

$$\begin{aligned}\begin{pmatrix}\dot{\boldsymbol{x}}_f(t)\\ \dot{\boldsymbol{x}}_\Delta(t)\end{pmatrix} &= \begin{pmatrix}\boldsymbol{A} - \boldsymbol{BKC} & -\boldsymbol{BKC}_{\mathrm{V}}\\ \boldsymbol{O} & \boldsymbol{A} - \boldsymbol{LC}_f\end{pmatrix}\begin{pmatrix}\boldsymbol{x}_f(t)\\ \boldsymbol{x}_\Delta(t)\end{pmatrix} + \begin{pmatrix}\boldsymbol{E}\\ -\boldsymbol{E}\end{pmatrix}\boldsymbol{d}(t)\\ \boldsymbol{y}_f(t) &= \begin{pmatrix}\boldsymbol{C}_f & \boldsymbol{O}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}_f(t)\\ \boldsymbol{x}_\Delta(t)\end{pmatrix}\\ \begin{pmatrix}\boldsymbol{x}_f(0)\\ \boldsymbol{x}_\Delta(0)\end{pmatrix} &= \begin{pmatrix}\boldsymbol{O}\\ \boldsymbol{O}\end{pmatrix}.\end{aligned}$$

It is obvious that the output $\boldsymbol{y}_f$ is different from the output $\boldsymbol{y}$ of the nominal control loop. The dynamical disturbance behaviour is much more complex because the number of states of the reconfigured process is $2n$ instead of n for the nominal process. The poles of the disturbance rejection behaviour depend on $\boldsymbol{K}$ and $\boldsymbol{L}$, while the zeros are affected by $\boldsymbol{P}$.

These results are summarised in the following theorem.

Theorem 9.2 (Virtual sensor) *For sensor faults, the virtual sensor (9.42)–(9.44) solves the reconfiguration problem such that the weak reconfiguration goal is reached provided that the faulty process is observable. The strong goal is reached for the reference tracking behaviour.*

The analysis has shown how the virtual sensor works. The direct feedthrough $\boldsymbol{P}$ reconstructs or at least approximates the output $\boldsymbol{y}_{\mathrm{c}}$ of the faultless plant from the remaining output $\boldsymbol{y}_f$. If the condition (9.11) is satisfied and $\boldsymbol{P}$ is chosen according to Eq. (9.13), the virtual sensor shrinks to a static reconfiguration block

$$\begin{aligned} \boldsymbol{y}_{\mathrm{c}}(t) &= \boldsymbol{C}\boldsymbol{C}_f^{\mathrm{T}} \left(\boldsymbol{C}_f^{\mathrm{T}} \boldsymbol{C}_f \right)^{-1} \boldsymbol{y}_f(t) \\ \boldsymbol{u}_f(t) &= \boldsymbol{u}_{\mathrm{c}}(t), \end{aligned}$$

because $\boldsymbol{C}_{\mathrm{V}} = \boldsymbol{O}$ results. This solution to the reconfiguration problem coincides with the solution obtained by the model-matching approach. The strong reconfiguration goal is satisfied.

If the condition (9.11) is not satisfied, the virtual sensor reconstructs the missing sensor information. Its state $\boldsymbol{x}_{\mathrm{V}}(t)$ approximates the plant state $\boldsymbol{x}_f(t)$. The strong reconfiguration goal is satisfied only for the reference tracking behaviour. The disturbance behaviour of the reconfigured closed-loop system is typically slower compared with the nominal behaviour. The smaller the state $\boldsymbol{x}_{\Delta}(t)$ in the model of the disturbance behaviour is, the better approximates the reconfigured loop the nominal behaviour.

9.2.4 Virtual Actuator

This section develops a solution to the reconfiguration problem for actuator failures. The notion of a virtual actuator is introduced as the dual system to the virtual sensor.

The system under consideration is described by

$$\dot{\boldsymbol{x}}_f(t) = \boldsymbol{A}\boldsymbol{x}_f(t) + \boldsymbol{B}_f \boldsymbol{u}_f(t) + \boldsymbol{E}\boldsymbol{d}(t), \quad \boldsymbol{x}_f(0) = \boldsymbol{x}_{\mathrm{f0}} \tag{9.56}$$

$$\boldsymbol{y}_f(t) = \boldsymbol{C}\boldsymbol{x}_f(t), \tag{9.57}$$

where zero columns in the matrix $\boldsymbol{B}_f$ reflect the failing actuators. If the condition (9.16) is satisfied, the static reconfiguration block

$$\begin{aligned} \boldsymbol{u}_f(t) &= \boldsymbol{N}\boldsymbol{u}_{\mathrm{c}}(t) \\ \boldsymbol{y}_{\mathrm{c}}(t) &= \boldsymbol{y}_f(t) \end{aligned}$$

can be used. In the following, the more general case is investigated, where this condition is not satisfied.

To explain the structure of the virtual actuator, the dual system of the reconfigured control loop for sensor faults shown in Fig. 9.11 is constructed. The result is shown in Fig. 9.12.

Definition 9.2 (*Virtual actuator*) Consider the plant (9.56), (9.57) with faulty actuator. The virtual actuator is defined as the system

$$\dot{\boldsymbol{x}}_{\Delta}(t) = \boldsymbol{A}_{\Delta}\boldsymbol{x}_{\Delta}(t) + \boldsymbol{B}_{\Delta}\boldsymbol{u}_{\mathrm{c}}(t), \quad \boldsymbol{x}_{\Delta}(0) = \boldsymbol{x}_{\Delta 0} \tag{9.58}$$

$$\boldsymbol{u}_{f}(t) = \boldsymbol{C}_{\Delta}\boldsymbol{x}_{\Delta}(t) + \boldsymbol{D}_{\Delta}\boldsymbol{u}_{\mathrm{c}}(t) \tag{9.59}$$

$$\boldsymbol{y}_{\mathrm{c}}(t) = \boldsymbol{C}\boldsymbol{x}_{\Delta}(t) + \boldsymbol{y}_{f}(t) \tag{9.60}$$

with the state $\boldsymbol{x}_{\Delta} \in |\mathcal{R}^{n}$ and the matrices

$$\boldsymbol{A}_{\Delta} = \boldsymbol{A} - \boldsymbol{B}_{f}\boldsymbol{M} \tag{9.61}$$

$$\boldsymbol{B}_{\Delta} = \boldsymbol{B} - \boldsymbol{B}_{f}\boldsymbol{N} \tag{9.62}$$

$$\boldsymbol{C}_{\Delta} = \boldsymbol{M} \tag{9.63}$$

$$\boldsymbol{D}_{\Delta} = \boldsymbol{N}. \tag{9.64}$$

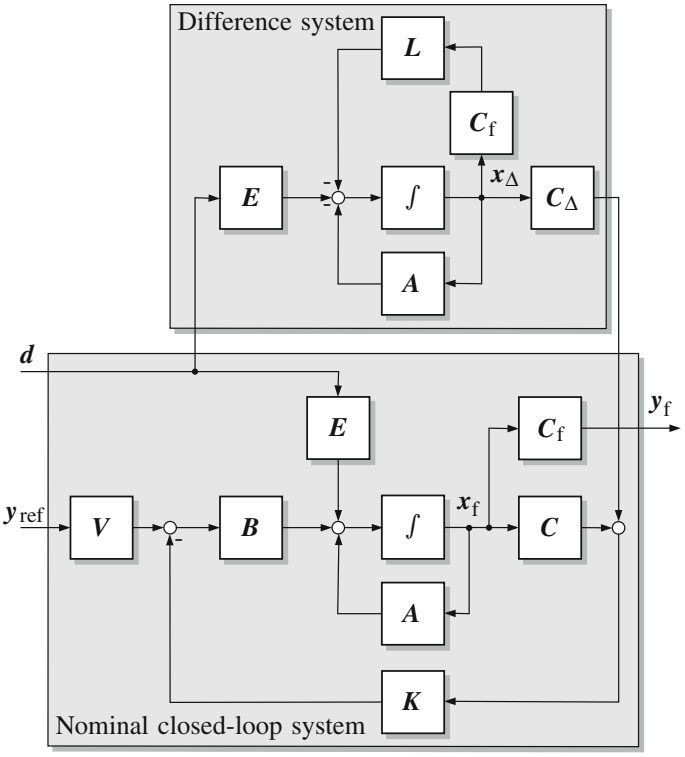


Fig. 9.11 Analysis of the closed-loop system with virtual sensor

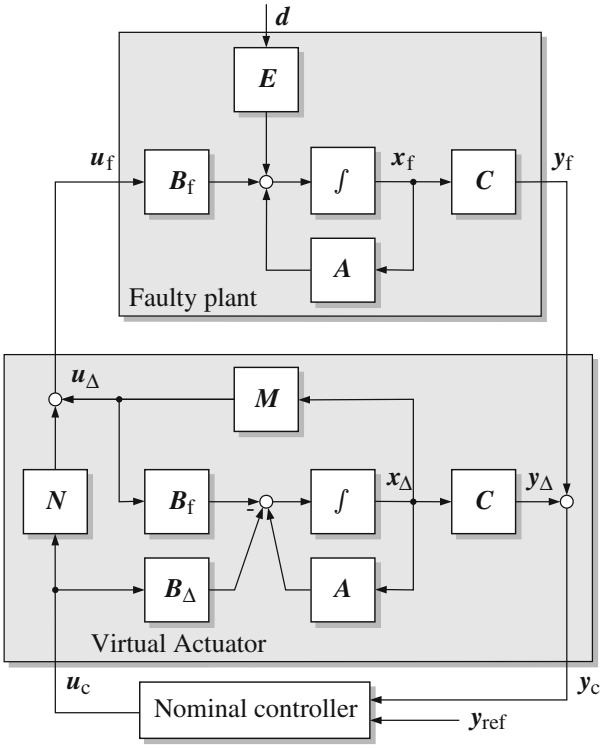


Fig. 9.12 Reconfiguration by means of a virtual actuator

$\boldsymbol{M}$ and $\boldsymbol{N}$ denote matrices that can be freely chosen.

Analysis of the reconfigured plant. The plant together with the virtual actuator leads to the following model of the reconfigured plant:

$$\begin{pmatrix} \dot{\boldsymbol{x}}_f(t) \\ \dot{\boldsymbol{x}}_\Delta(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B}_f \boldsymbol{M} \\ \boldsymbol{O} & \boldsymbol{A} - \boldsymbol{B}_f \boldsymbol{M} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_\Delta(t) \end{pmatrix} + \begin{pmatrix} \boldsymbol{B}_f \boldsymbol{N} \\ \boldsymbol{B} - \boldsymbol{B}_f \boldsymbol{N} \end{pmatrix} \boldsymbol{u}_\mathrm{c}(t) + \begin{pmatrix} \boldsymbol{E} \\ \boldsymbol{O} \end{pmatrix} \boldsymbol{d}(t) \tag{9.65}$$

$$\boldsymbol{y}_\mathrm{c}(t) = \begin{pmatrix} \boldsymbol{C} & \boldsymbol{C} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_\Delta(t) \end{pmatrix}. \tag{9.66}$$

The introduction of the new state $\hat{\boldsymbol{x}}(t) = \boldsymbol{x}_f(t) + \boldsymbol{x}_\Delta(t)$ leads to the following equivalent model:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\hat{\boldsymbol{x}}(t)\\ \boldsymbol{x}_\Delta(t)\end{pmatrix} &= \begin{pmatrix}\boldsymbol{A} & \boldsymbol{O}\\ \boldsymbol{O} & \boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{M}\end{pmatrix}\begin{pmatrix}\hat{\boldsymbol{x}}(t)\\ \boldsymbol{x}_\Delta(t)\end{pmatrix}\\ &\quad + \begin{pmatrix}\boldsymbol{B}\\ \boldsymbol{B}-\boldsymbol{B}_f\boldsymbol{N}\end{pmatrix}\boldsymbol{u}_\mathrm{c}(t) + \begin{pmatrix}\boldsymbol{E}\\ \boldsymbol{O}\end{pmatrix}\boldsymbol{d}(t)\\ \boldsymbol{y}_\mathrm{c}(t) &= \begin{pmatrix}\boldsymbol{C} & \boldsymbol{O}\end{pmatrix}\begin{pmatrix}\hat{\boldsymbol{x}}(t)\\ \boldsymbol{x}_\Delta(t)\end{pmatrix}\\ \begin{pmatrix}\hat{\boldsymbol{x}}(0)\\ \boldsymbol{x}_\Delta(0)\end{pmatrix} &= \begin{pmatrix}\boldsymbol{x}_0+\boldsymbol{x}_{\Delta 0}\\ \boldsymbol{x}_{\Delta 0}\end{pmatrix}.\end{aligned}$$

Note that the state $\boldsymbol{x}_\Delta$ of the second subsystem is not observable by $\boldsymbol{y}_\mathrm{c}$. Hence, this state does not influence the I/O-behaviour of the reconfigured plant, whose model can be reduced to

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}_\mathrm{c}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 + \boldsymbol{x}_{\Delta 0}\\ \boldsymbol{y}_\mathrm{c}(t) &= \boldsymbol{C}\boldsymbol{x}(t).\end{aligned}$$

This model is identical to the nominal plant provided that $\boldsymbol{x}_{\Delta 0} = \boldsymbol{0}$ holds.

Theorem 9.3 *The reconfigured plant (9.56)–(9.64) has the same I/O-behaviour as the nominal plant (9.38), (9.39) for arbitrary parameter matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ of the virtual actuator.*

Hence, the virtual actuator yields a reconfigured plant that satisfies the fault-hiding goal for arbitrary matrices $\boldsymbol{M}$ and $\boldsymbol{N}$.

Separation principle for the virtual actuator. The reconfigured closed-loop system consists of the reconfigured plant and the controller (9.37), both of which are considered for vanishing disturbance $\boldsymbol{d}$ and command input $\boldsymbol{y}_\mathrm{ref}$. If the transformed model is used, the reconfigured closed-loop system is described by

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\hat{\boldsymbol{x}}(t)\\ \boldsymbol{x}_\Delta(t)\end{pmatrix} &= \begin{pmatrix}\boldsymbol{A}-\boldsymbol{B}\boldsymbol{K}\boldsymbol{C} & \boldsymbol{O}\\ -\boldsymbol{B}_\Delta\boldsymbol{K}\boldsymbol{C} & \boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{M}\end{pmatrix}\begin{pmatrix}\hat{\boldsymbol{x}}(t)\\ \boldsymbol{x}_\Delta(t)\end{pmatrix}\\ \begin{pmatrix}\hat{\boldsymbol{x}}(0)\\ \boldsymbol{x}_\Delta(0)\end{pmatrix} &= \begin{pmatrix}\boldsymbol{x}_0+\boldsymbol{x}_{\Delta 0}\\ \boldsymbol{x}_{\Delta 0}\end{pmatrix}.\end{aligned}$$

As the system matrix is a block triangular matrix, the following result is obtained:

Theorem 9.4 (Separation principle for the virtual actuator) *The set σ of eigenvalues of the reconfigured closed-loop system (9.37), (9.56)–(9.64) consists of the set of eigenvalues of the nominal closed-loop system (9.37)–(9.39) and the set of eigenvalues of the virtual actuator (9.58):*

$$\sigma = \sigma\{\boldsymbol{A}-\boldsymbol{B}\boldsymbol{K}\boldsymbol{C}\} \cup \sigma\{\boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{M}\}.$$

This theorem holds true for arbitrary matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ of the virtual actuator. Clearly, a corollary of this theorem is that the matrix $\boldsymbol{M}$ has to be chosen so that the matrix $\boldsymbol{A} - \boldsymbol{B}_f \boldsymbol{M}$ has eigenvalues with negative real parts in order to ensure the stability of the reconfigured closed-loop system.

Corollary 9.1 *The stability of the reconfigured closed-loop system can be ensured by appropriately choosing the matrix $\boldsymbol{M}$ of the virtual actuator if and only if the pair $(\boldsymbol{A}, \boldsymbol{B}_f)$ is stabilisable.*

This corollary shows that the stabilisation goal can be satisfied by using the generalised virtual actuator as long as the faulty plant is stabilisable.

I/O-behaviour of the reconfigured closed-loop system. The following investigates the I/O-behaviour of the reconfigured closed-loop system and derives guidelines for choosing the parameter matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ of the virtual actuator. If the models of the faulty plant (9.56), (9.57) is combined with the virtual actuator (9.58)–(9.60) and the controller (9.37), the following model is obtained after the state $\hat{\boldsymbol{x}}$ has been introduced as before:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix}\hat{\boldsymbol{x}}(t)\\ \boldsymbol{x}_\Delta(t)\end{pmatrix} &= \begin{pmatrix}\boldsymbol{A}-\boldsymbol{B}\boldsymbol{K}\boldsymbol{C} & \boldsymbol{O}\\ -\boldsymbol{B}_\Delta\boldsymbol{K}\boldsymbol{C} & \boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{M}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}(t)\\ \boldsymbol{x}_\Delta(t)\end{pmatrix} \\ &\quad + \begin{pmatrix}\boldsymbol{B}\boldsymbol{V}\\ \boldsymbol{B}_\Delta\boldsymbol{V}\end{pmatrix}\boldsymbol{y}_{\mathrm{ref}}(t) + \begin{pmatrix}\boldsymbol{E}\\ \boldsymbol{O}\end{pmatrix}\boldsymbol{d}(t) \qquad (9.67)\\ \begin{pmatrix}\hat{\boldsymbol{x}}(0)\\ \boldsymbol{x}_\Delta(0)\end{pmatrix} &= \begin{pmatrix}\boldsymbol{x}_0+\boldsymbol{x}_{\Delta 0}\\ \boldsymbol{x}_{\Delta 0}\end{pmatrix}\\ \boldsymbol{y}_{\mathrm{c}}(t) &= (\boldsymbol{C} \quad \boldsymbol{O})\begin{pmatrix}\hat{\boldsymbol{x}}(t)\\ \boldsymbol{x}_\Delta(t)\end{pmatrix} \qquad (9.68)\\ \boldsymbol{y}_f(t) &= (\boldsymbol{C} \quad -\boldsymbol{C})\begin{pmatrix}\hat{\boldsymbol{x}}(t)\\ \boldsymbol{x}_\Delta(t)\end{pmatrix}. \qquad (9.69)\end{aligned}$$

The block diagram that illustrates this model is shown in Fig. 9.13. The lower block represents the nominal closed-loop system. The control error $\boldsymbol{e} = \boldsymbol{V}\boldsymbol{y}_{\mathrm{ref}} - \boldsymbol{y}_{\mathrm{c}}$ is fed into the "difference system"

$$\dot{\boldsymbol{x}}_\Delta(t) = (\boldsymbol{A} - \boldsymbol{B}_f\boldsymbol{M})\boldsymbol{x}_\Delta(t) + \boldsymbol{B}_\Delta\boldsymbol{e}(t), \quad \boldsymbol{x}_\Delta(0) = \boldsymbol{x}_{\Delta 0} \qquad (9.70)$$

$$\boldsymbol{y}_\Delta(t) = \boldsymbol{C}\boldsymbol{x}_\Delta(t), \qquad (9.71)$$

whose name results from its output $\boldsymbol{y}_\Delta$, which is the difference between the output $\boldsymbol{y}_{\mathrm{c}}$ of the nominal closed-loop system and the output $\boldsymbol{y}_f$ of the reconfigured closed-loop system. Hence, $\boldsymbol{y}_\Delta$ shows how the reconfigured closed-loop system differs from the nominal loop.

This model yields two corollaries:

- The I/O-behaviour with respect to the disturbance input $\boldsymbol{d}$ or the command input $\boldsymbol{y}_{\mathrm{ref}}$, respectively, and to the output $\boldsymbol{y}_{\mathrm{c}}$ is identical to the corresponding I/O-behaviour of the nominal closed-loop system.

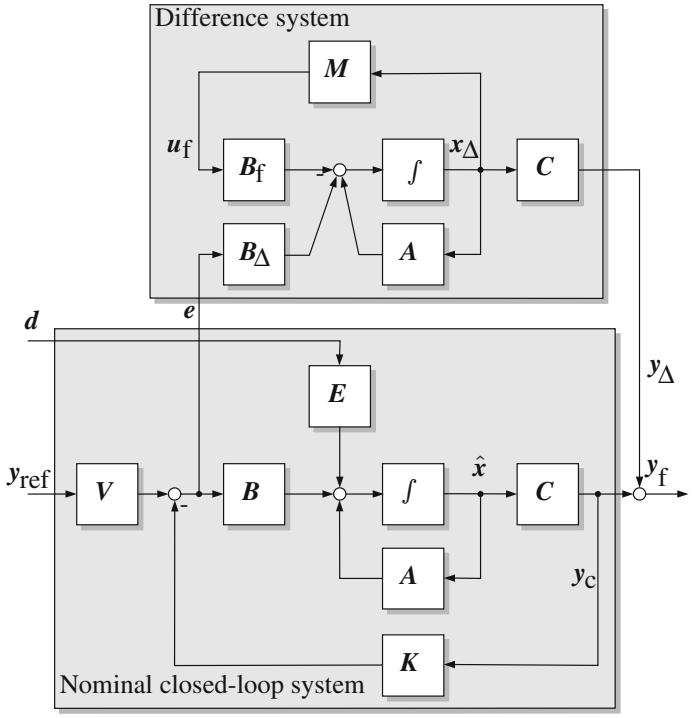


Fig. 9.13 Transformed closed-loop system showing the separation principle

- The I/O-behaviour with respect to the disturbance input $\boldsymbol{d}$ or the command input $\boldsymbol{y}_{\text{ref}}$, respectively, and to the output $\boldsymbol{y}_f$ differs from that of the nominal closed-loop system due to the influence of the difference system (9.70), (9.71).

To summarise these results, the virtual actuator presents a successful reconfiguration block in case of actuator failures. It creates a stable control loop with n placeable additional poles. However, it does not restore the original equilibrium unless the equilibrium is zero.

Theorem 9.5 (Virtual actuator) *For actuator failures, the virtual actuator (9.58)–(9.64) is a solution to the reconfiguration problem such that the weak reconfiguration goal is reached provided that the faulty process is controllable.*

The following part of this section concerns the question how to choose the matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ of the virtual actuator in order to get a small difference $\boldsymbol{y}_\Delta$ between the behaviour of the nominal and the reconfigured closed-loop system.

Complete reconfiguration. As Fig. 9.13 and Eqs. (9.70), (9.71) show, a complete reconfiguration is possible if the matrix $\boldsymbol{N}$ can be chosen such that the matrix $\boldsymbol{B}_\Delta$ vanishes.

Corollary 9.2 *If the matrix* $\boldsymbol{N}$ *can be chosen such that*

$$\boldsymbol{B}_{\Delta} = \boldsymbol{B} - \boldsymbol{B}_f \boldsymbol{N} = \boldsymbol{O} \tag{9.72}$$

holds, the I/O-behaviour of the reconfigured closed-loop system is identical to that of the nominal control loop for both the disturbance input $\boldsymbol{d}$ *and the command input* $\boldsymbol{y}_{\mathrm{ref}}$. *Furthermore, if*

$$\boldsymbol{x}_{\Delta}(0) = \boldsymbol{0} \tag{9.73}$$

holds, the reconfigured loop has the same free motion as the nominal loop.

The condition (9.72) can be satisfied for an arbitrary controller (9.37) if and only if the relation (9.16) holds. Then the virtual actuator (9.58), (9.60) reduces to the static reconfiguration block

$$\boldsymbol{u}_f(t) = (\boldsymbol{B}_f^{\mathrm{T}} \boldsymbol{B}_f)^{-1} \boldsymbol{B}_f^{\mathrm{T}} \boldsymbol{B} \boldsymbol{u}_{\mathrm{c}}(t) \tag{9.74}$$

$$\boldsymbol{y}_{\mathrm{c}}(t) = \boldsymbol{y}_f(t), \tag{9.75}$$

which is identical to the reconfiguration solution described in Sect. 9.1.4.

If the condition (9.16) is violated, this static reconfiguration block does not solve the reconfiguration problem, the inequality $\boldsymbol{B}_{\Delta} \neq \boldsymbol{O}$ holds and the dynamical part of the virtual actuator becomes active.

Design of the virtual actuator by disturbance decoupling methods. If the transfer function matrix of the difference system (9.70), (9.71) vanishes

$$\boldsymbol{G}(s) = \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A} + \boldsymbol{B}_f \boldsymbol{M})^{-1}(\boldsymbol{B} - \boldsymbol{B}_f \boldsymbol{N}) = \boldsymbol{O}, \tag{9.76}$$

the reconfiguration is complete as well. Then the difference model (9.70), (9.71), which can be equivalently written as

$$\dot{\boldsymbol{x}}_{\Delta}(t) = \boldsymbol{A}\boldsymbol{x}_{\Delta}(t) + \boldsymbol{B}\boldsymbol{u}_{\mathrm{c}}(t) + \boldsymbol{B}_f \boldsymbol{u}_f(t), \quad \boldsymbol{x}_{\Delta}(0) = \boldsymbol{x}_{\Delta 0} \tag{9.77}$$

$$\boldsymbol{u}_{\Delta}(t) = \boldsymbol{M}\boldsymbol{x}_{\Delta}(t) + \boldsymbol{N}\boldsymbol{u}_{\mathrm{c}}(t) + \boldsymbol{Q}\tilde{\boldsymbol{u}}(t) \tag{9.78}$$

has a vanishing output. To select the matrices $\boldsymbol{N}$ and $\boldsymbol{M}$ such that the condition (9.76) holds is a disturbance decoupling problem for known disturbance $\boldsymbol{u}_{\mathrm{c}}$. It has been shown in [340] that the solution to this problem yields a complete reconfiguration. This solution exist, however, only under restrictive conditions.

Restoration of the static behaviour. The static behaviour is completely reconstructed if the gain of the difference system vanishes:

$$\boldsymbol{G}(0) = -\boldsymbol{C}(\boldsymbol{A} - \boldsymbol{B}_f \boldsymbol{M})^{-1}(\boldsymbol{B} - \boldsymbol{B}_f \boldsymbol{N}) = \boldsymbol{O}. \tag{9.79}$$

Approximate solution. The generalised virtual actuator has the property that the effect of the virtual actuator "disappears" if the matrix $\boldsymbol{B}_\Delta$ can be made very small by choosing the matrix $\boldsymbol{N}$ appropriately.

Corollary 9.3 *For $\|\boldsymbol{B}_\Delta\| \to 0$, the behaviour of the reconfigured closed-loop system approaches that of the nominal loop:*

$$\|\boldsymbol{y}_c(t) - \boldsymbol{y}_f(t)\| \to 0.$$

Hence, if $\|\boldsymbol{B}_\Delta\|$ is sufficiently small it is reasonable to use the static reconfiguration block only.

Example 9.4 Reconfiguration of the two-tank system
To illustrate the reconfiguration by means of the virtual actuator, the problem posed in Example 9.3 is considered. The tank system is described by the nonlinear state-space model

$$\begin{aligned}
\dot{h}_1(t) &= \frac{Q_{1\max}}{A_1}(-k_I x_r(t) - k_P(h_1(t) - u_1(t))) \\
&\quad - \frac{Q_{1\max}}{S}\sqrt{2g(h_1(t) - h_v)}u_2(t) - \frac{Q_{1\max}}{S}\sqrt{2gh_1(t)}u_3(t) \\
\dot{x}_r(t) &= h_1(t) - u_1(t) \\
\dot{h}_2(t) &= \frac{1}{A_2}\left(S\sqrt{2g(h_1(t) - h_v)}u_2(t) + S\sqrt{2gh_1(t)}u_3(t) - S\sqrt{2gh_2(t)}d(t)\right) \\
y_c(t) &= h_2(t)
\end{aligned}$$

that includes the controller of the left tank, which is a PI controller

$$\begin{aligned}
\dot{x}_r(t) &= h_1(t) - u_1(t) \\
\tilde{u}_1(t) &= -k_I x_r(t) - k_P(h_1(t) - u_1(t)).
\end{aligned}$$

This model uses the following parameters:

Symbol	Physical meaning
A_1, A_2	Cross section areas of the two tanks
$Q_{1\max}$	Maximum flow through the pump
h_v	Height of the upper pipe above the tank bottom
S	Constant of the valves
g	gravity constant
k_I, k_P	Controller parameters

After the linearisation of the model around the operation point described by $\bar{h}_1$, $\bar{h}_2$, $\bar{u}_1$, $\bar{u}_2$, $\bar{u}_3$, the linear model (9.35), (9.36) with

$$\boldsymbol{A} = \begin{pmatrix} -0.0478 & -0.0004 & 0 \\ 1.0000 & 0 & 0 \\ 0.0058 & 0 & -0.0058 \end{pmatrix}$$
$$\boldsymbol{B} = \begin{pmatrix} 0.0406 & -0.0058 & -0.0092 \\ -1.0000 & 0 & 0 \\ 0 & 0.0046 & 0.0073 \end{pmatrix}$$
$$\boldsymbol{C} = (0 \quad 0 \quad 1)$$
$$\boldsymbol{E} = \begin{pmatrix} 0 \\ 0 \\ -0.0454 \end{pmatrix}$$

is obtained. Is is assumed that the upper valve fails and is, therefore, completely closed and no longer used as actuator of the right level controller. Then, the second column in the matrix $\boldsymbol{B}$ has to be set to zero to obtain the matrix $\boldsymbol{B}_f$:

$$\boldsymbol{B} = \begin{pmatrix} 0.0406 & 0 & -0.0092 \\ -1.0000 & 0 & 0 \\ 0 & 0 & 0.0073 \end{pmatrix}.$$

Static reconfiguration. A complete reconfiguration of the controller is possible, because the condition (9.16) is satisfied due to the lower valve, which represents a redundant control input with similar effects on the tank system as the upper valve. In fact, the last column of $\boldsymbol{B}$ is linearly dependent upon the second column:

$$0.6325 \begin{pmatrix} -0.0092 \\ 0 \\ 0.0073 \end{pmatrix} = \begin{pmatrix} -0.0058 \\ 0 \\ 0.0046 \end{pmatrix}.$$

Hence, the reconfiguration is possible with a static reconfiguration block (9.74), for which the following parameters are obtained (Fig. 9.14):

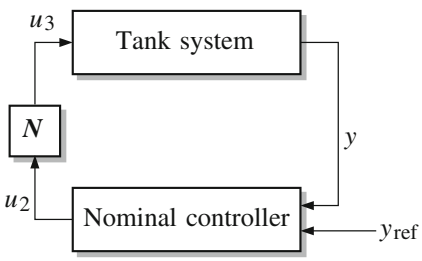


Fig. 9.14 Static reconfiguration of the tank system

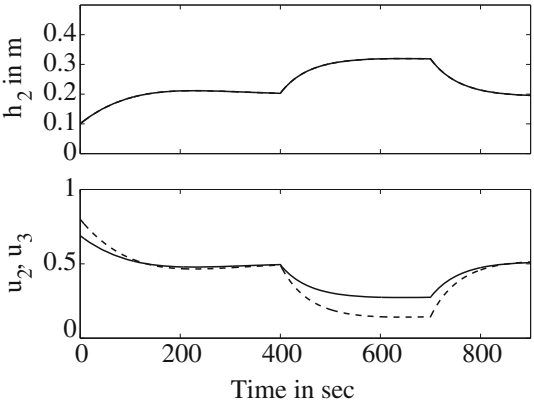


Fig. 9.15 Behaviour of the reconfigured closed-loop system where the reconfigured controller uses the input u_3

$$u_f(t) = \begin{pmatrix} 0 & 0 & -2.7039 \\ 0 & 0 & 0.6325 \\ 0 & 0 & 1 \end{pmatrix} u_c(t).$$

Figure 9.15 shows the reference tracking behaviour of the right tank for changing level set-point. The right tank has the same behaviour with the reconfigured controller as in the nominal case. In the lower subplot the control input u_3 used by the reconfigured controller is compared to the input u_2 of the nominal controller, which is shown by the dashed lines. Clearly, the new input has to be smaller than the nominal one, because the lower valve between the tanks has a higher effectiveness than the upper one, which can be seen by comparing the corresponding columns in the matrix $\boldsymbol{B}$.

Reconfiguration by means of the virtual actuator. If the lower valve is not available for the reconfiguration, the right controller has only the input u_1, which is the command signal of the left controller, as its disposal. With the third columns deleted, the matrices $\boldsymbol{B}$ and $\boldsymbol{B}_f$ do no longer satisfy the condition (9.16). Hence, a dynamical reconfiguration block has to be used. The matrix $\boldsymbol{N}$ of the virtual actuator is chosen according to Eq. (9.18):

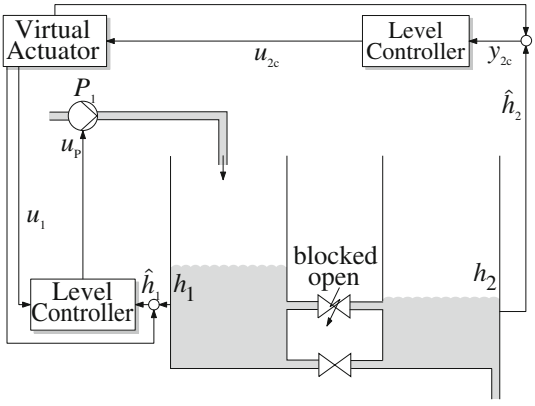


Fig. 9.16 Reconfigured system with virtual actuator

$$\boldsymbol{N} = \begin{pmatrix} 1 & -0.0002 & -0.0004 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

The nominal closed-loop system has the eigenvalues -0.0427, $-0.0124 \pm 0.0058i$. Therefore, the matrix $\boldsymbol{M}$ of the virtual actuator is chosen so as to place the eigenvalues of the matrix $\boldsymbol{A} - \boldsymbol{B}_f \boldsymbol{M}$ to the left of these eigenvalues, namely at -0.05, -0.06 and -0.07. As $\boldsymbol{M}$ should use only the first input, its non-zero elements are restricted to the first row:

$$\boldsymbol{M} = \begin{pmatrix} -0.9968 & -0.0048 & -0.0002 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

In summary, the virtual actuator (9.58), (9.59) results

$$\begin{aligned} \dot{\boldsymbol{x}}_\Delta(t) = & \begin{pmatrix} -0.0074 & -0.0002 & 0 \\ 0.0032 & -0.0048 & -0.0002 \\ 0.0058 & 0 & -0.0058 \end{pmatrix} \boldsymbol{x}_\Delta(t) \; + \\ & + \begin{pmatrix} 0 & -0.0058 & -0.0091 \\ 0 & -0.0002 & -0.0004 \\ 0 & 0.0046 & 0.0073 \end{pmatrix} u_{2c}(t) \\ u_1(t) = & (-0.9968 \quad -0.0048 \quad -0.0002)\, \boldsymbol{x}_\Delta(t) \; + \\ & + (1 \quad -0.0002 \quad 0.0004)\, \boldsymbol{y}_c(t), \end{aligned}$$

where $u_{2c}(t)$ is the control input generated by the nominal level controller of the right tank. This signal is used now as an input of the virtual actuator (Fig. 9.16).

Figure 9.17 shows the disturbance behaviour of the tank system after the controller has been extended by a virtual actuator shown above. The response is slower than the nominal

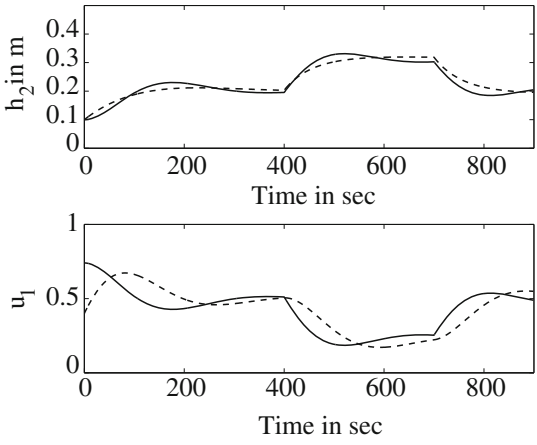


Fig. 9.17 Behaviour of the reconfigured closed-loop system where the reconfigured controller uses the input u_1

response, which is drawn by dashed lines to make a comparison possible. The slower response results from the fact that the controller of the right tank uses now the command input of the controller of the left tank as control input. □

9.2.5 Duality Between Virtual Sensors and Virtual Actuators

The comparison of the reconfiguration blocks developed in the preceding sections clearly shows the duality of the approaches for sensor and actuator failures. The variables correspond to each other in the following way:

Note that the duality involves more than just swapping input and output and transposing the matrices. It requires that the directions of the signals be reversed. Summation points become signal knots and vice versa. The diagrams also require mirroring to preserve the clockwise signal direction of the control loop.

A short mathematical demonstration of the duality is given here. If the system (9.48) is transposed, the input and output matrices are exchanged, and all system matrices are transposed, the following model results:

$$\begin{aligned}\begin{pmatrix}\dot{\boldsymbol{x}}_f(t)\\ \dot{\hat{\boldsymbol{x}}}(t)\end{pmatrix} &= \begin{pmatrix}\boldsymbol{A}^{\mathrm{T}} & \boldsymbol{C}_f^{\mathrm{T}}\boldsymbol{L}^{\mathrm{T}}\\ \boldsymbol{O} & \boldsymbol{A}^{\mathrm{T}}-\boldsymbol{C}_f^{\mathrm{T}}\boldsymbol{L}^{\mathrm{T}}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}_f(t)\\ \hat{\boldsymbol{x}}(t)\end{pmatrix}\\ &\quad+\begin{pmatrix}\boldsymbol{C}_f^{\mathrm{T}}\boldsymbol{P}^{\mathrm{T}}\\ \boldsymbol{C}^{\mathrm{T}}-\boldsymbol{C}_f^{\mathrm{T}}\boldsymbol{P}^{\mathrm{T}}\end{pmatrix}\boldsymbol{u}_f(t)\\ \boldsymbol{y}_{\mathrm{c}}(t) &= \begin{pmatrix}\boldsymbol{B} & \boldsymbol{B}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}_f(t)\\ \hat{\boldsymbol{x}}(t)\end{pmatrix}.\end{aligned}$$

Apart from the different variable names according to Table 9.1, the result is identical to (9.65)–(9.66). The duality holds for most properties, but not for the reference tracking. The reason is that $\boldsymbol{y}_{\mathrm{ref}}$ and $\boldsymbol{y}$ do not have symmetric positions in the system.

9.2.6 Experimental Evaluation: Level and Temperature Control

Reconfiguration of a level and temperature control loop. For a demonstration of the control reconfiguration in case of an actuator failure the part of the chemical process shown in Fig. 9.18 is considered. The control objectives are to maintain a constant liquid level and a constant temperature in the reactor tank B_1 and, thus, producing a constant product outflow. To achieve this, hot and cold liquid can be

Table 9.1 Duality of the system variables

Virtual sensor	$\boldsymbol{A}$	$\boldsymbol{B}$	$\boldsymbol{C}$	$\boldsymbol{K}$	$\boldsymbol{L}$	$\boldsymbol{P}$	$\hat{x}$	u	y
Virtual actuator	$\boldsymbol{A}^{\mathrm{T}}$	$\boldsymbol{C}^{\mathrm{T}}$	$\boldsymbol{B}^{\mathrm{T}}$	$\boldsymbol{K}^{\mathrm{T}}$	$\boldsymbol{M}^{\mathrm{T}}$	$\boldsymbol{N}^{\mathrm{T}}$	$\tilde{x}$	y	u

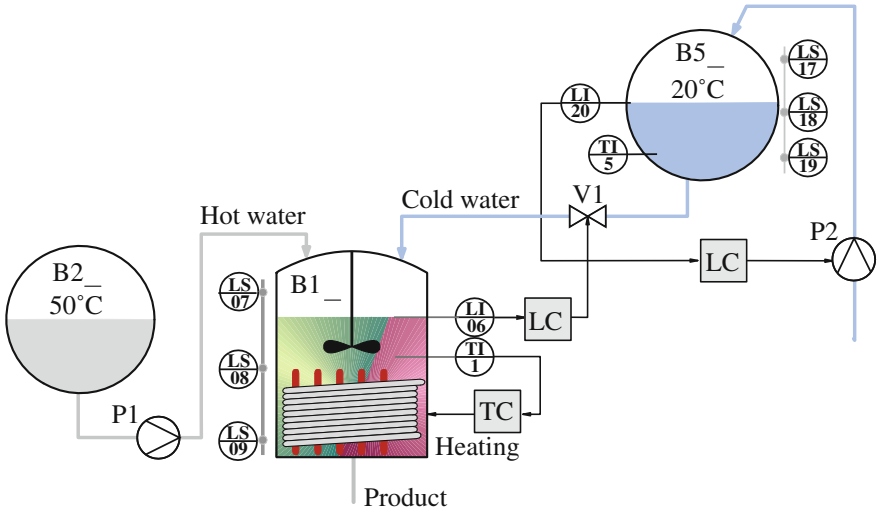


Fig. 9.18 Plant used for control reconfiguration (LC - level control, TC - temperature control)

brought into the reactor from Tanks B_2 and B_5. The main reactor B_1 can be heated and cooled.

In the nominal case, the liquid level is controlled by adjusting the cold liquid inflow from Tank B_5 and the temperature by means of the heating.

Plant model. The plant model contains three states: the reactor content V_{B1}, the reactor temperature ϑ_{B1} and the content of the cold liquid tank V_{B5}. From a mass balance, the following equations are obtained

$$\begin{aligned}
\dot{V}_{\mathrm{B5}}(t) &= k_{\mathrm{P2}} u_{\mathrm{P2}}(t) - q_{51}(t) \\
\dot{V}_{\mathrm{B1}}(t) &= q_{21}(t) + q_{51}(t) - q_{1\mathrm{out}}(t) \\
\dot{\vartheta}_{\mathrm{B1}}(t) &= (\vartheta_{B2}(t) - \vartheta_{\mathrm{B1}}(t)) \frac{q_{21}(t)}{V_{\mathrm{B1}}(t)} + (\vartheta_{\mathrm{B5}}(t) - \vartheta_{\mathrm{B1}}(t)) \frac{q_{51}(t)}{V_{\mathrm{B1}}(t)} \\
&\quad + \frac{u_{\mathrm{heat}}(t) k_{\mathrm{heat}}}{V_{\mathrm{B1}}(t)},
\end{aligned}$$

where for the liquid flows the relations

$$\begin{aligned}
q_{21}(t) &= k_{\mathrm{P1}} u_{\mathrm{P1}}(t) \\
q_{51}(t) &= k_{\mathrm{V1}}\, 124.5^{u_{\mathrm{V1}}(t)} \sqrt{h_{\mathrm{B5}}(t) + 1.07} \\
q_{1\mathrm{out}}(t) &= k_{\mathrm{V2}} \sqrt{\frac{V_{\mathrm{B1}}(t)}{A_{\mathrm{B1}}} + 1.4}
\end{aligned}$$

hold. $h_{B5}(t)$ is the liquid level in the spherical tank B_5, $u_{heat}(t)$ the heating power, k_{heat} a heating coefficient, $u_{P1}(t)$, $u_{P2}(t)$ and $u_{V1}(t)$ the control input to the two pumps and to the Valve V_1 and A_{B1} the cross-section area of the Tank B_1. After a linearisation of this nonlinear model around the operating point of $\vartheta_{B1} = 40\,^{\circ}\mathrm{C}$, the following linear model is obtained:

$$\begin{pmatrix} \dot{V}_{B5}(t) \\ \dot{V}_{B1}(t) \\ \dot{\vartheta}_{B1}(t) \end{pmatrix} = 10^{-3} \begin{pmatrix} -0.46 & 0 & 0 \\ +0.46 & -0.33 & 0 \\ -0.48 & 0.008 & -1.1 \end{pmatrix} \begin{pmatrix} V_{B5}(t) \\ V_{B1}(t) \\ \vartheta_{B1}(t) \end{pmatrix} + \begin{pmatrix} 0.09 & -0.023 & 0 & 0 \\ 0 & +0.023 & +0.05 & 0 \\ 0 & -0.024 & +0.02 & 0.223 \end{pmatrix} \begin{pmatrix} u_{P2}(t) \\ u_{V1}(t) \\ u_{P1}(t) \\ u_{heat}(t) \end{pmatrix}$$

$$\boldsymbol{y} = \begin{pmatrix} h_{B5}(t) \\ h_{B1}(t) \\ \vartheta_{B1}(t) \end{pmatrix}.$$

The nominal proportional controllers are defined by:

$$\begin{aligned} u_{V1}(t) &= -0.5\, V_{B1}(t) \\ u_{heat}(t) &= -0.5\, \vartheta_{B1}(t) \\ u_{P2}(t) &= -1\, V_{B5}(t). \end{aligned}$$

They can be represented as

$$\boldsymbol{u}(t) = -\boldsymbol{K}\,\boldsymbol{y}(t) \quad \text{with} \quad \boldsymbol{K} = \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.5 \end{pmatrix}.$$

Note that these controllers do not use the control input u_{P1}, because the matrix $\boldsymbol{K}$ has a vanishing third row.

Faults. Several severe faults can occur that open the control loops. For example, due to a heating failure, the reactor can no longer be heated, or clogging or blockage of Valve V_1 can bring the level controller out of operation. In the following, the heating failure and a blockage of Valve V_1 in its nominal position will be considered.

Controller reconfiguration after a heating failure. After a heating failure has occurred, the temperature controller

$$u_{heat}(t) = -0.5\, \vartheta_{B1}(t)$$

has no influence on the process. The system in the nominal and the faulty case has the matrices

$$\boldsymbol{B} = \begin{pmatrix} 0.09 & -0.023 & 0 & 0 \\ 0 & +0.023 & +0.05 & 0 \\ 0 & -0.024 & +0.02 & 0.223 \end{pmatrix}$$
$$\boldsymbol{B}_f = \begin{pmatrix} 0.09 & -0.023 & 0 & 0 \\ 0 & +0.023 & +0.05 & 0 \\ 0 & -0.024 & +0.02 & 0 \end{pmatrix},$$

which distinguish in the last column. Both matrices have the same rank and can be related to one another by the matrix

$$\boldsymbol{N} = \begin{pmatrix} 1 & 0 & 0 & -1.72 \\ 0 & 1 & 0 & -6.72 \\ 0 & 0 & 1 & 3.09 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

such that the equation

$$\boldsymbol{B}_f \boldsymbol{N} = \boldsymbol{B}$$

holds. Hence, a complete reconfiguration is possible by using the third control input, which is not used in the nominal case. The reconfigured controller

$$\boldsymbol{u}(t) = -\boldsymbol{N}\boldsymbol{K}\,\boldsymbol{y}(t)$$

has the controller matrix

$$\boldsymbol{N}\boldsymbol{K} = \begin{pmatrix} 0.5 & 0 & -0.86 \\ 0 & 1 & -3.36 \\ 0 & 0 & 1.55 \\ 0 & 0 & 0 \end{pmatrix}.$$

Obviously, the fourth actuator is no longer used. The effect of this actuator is distributed among the three remaining actuators, which can be seen in the last column of the new controller matrix. With the reconfigured controller, the behaviour of the nominal system is completely reproduced.

Controller reconfiguration by means of a virtual actuator. The loss of the actuator V_1 does not affect the operation point, but it breaks the level control loop for the reactor B_1. The use of a reduced virtual actuator allows to keep the nominal controller while changing the control structure as little as possible.

In the terminology of Sect. 9.2.3, the directly influenceable part $\boldsymbol{x}_{F1}$ of the plant state is defined by V_{B5} and ϑ_{B1}, while $\boldsymbol{x}_{F2}$ is the single state variable V_{B1}:

$$\boldsymbol{x}_{\mathrm{f1}}(t) = \begin{pmatrix} B_{\mathrm{B5}}(t) \\ \vartheta_{\mathrm{B5}}(t) \end{pmatrix}, \qquad \boldsymbol{x}_{\mathrm{f2}}(t) = V_{\mathrm{B1}}(t).$$

The $(1, 2)$-parameter matrix $\boldsymbol{M}$ is determined by pole placement. The element of $\boldsymbol{M}$ that is acting on ϑ_{B1} has no influence on the actuator pole and is, therefore, set to 0. The other value is chosen so that the actuator pole lies at -0.004 in order to make the influence of the virtual actuator on the closed-loop dynamics as small as possible. The application of the method explained in Sect. 9.2.4 to this example leads to

$$\begin{aligned}
\dot{\hat{x}}_2(t) &= -0.004\,\hat{x}_2(t) + 0.0229\,u_{\mathrm{V2,R}}(t) \\
\hat{\boldsymbol{u}}(t) &= \begin{pmatrix} 0.015 \\ -0.318 \\ 0 \end{pmatrix} \hat{x}_2(t) + \begin{pmatrix} -0.107 \\ 1.78 \\ 0 \end{pmatrix} u_{\mathrm{V2,R}}(t) \\
\hat{\boldsymbol{y}}(t) &= \begin{pmatrix} -8 \\ 0 \\ 1 \end{pmatrix} \hat{x}_2(t).
\end{aligned}$$

The function of the reduced virtual actuator can be described as follows (Fig. 9.19). The input $u_{\mathrm{V1}}(t)$ is not available to control the inflow into the main reactor, but this inflow also depends on the level in Tank B_5 and, hence, on V_{B5}. In order to reach the same effect as the broken actuator, $V_{\mathrm{B5}}(t)$ is increased or decreased by influencing the Pump P_2 via the input $u_{\mathrm{P2}}(t)$. As $V_{\mathrm{B5}}(t)$ cannot be changed instantaneously, this "replacement action" is slower than the direct action of the nominal control loop on the valve V_1 and leads to a slower reaction of the system under the influence of the reconfigured controller.

In mathematical terms, the virtual actuator brings about an additional pole which yields the slower dynamics. The difference between the nominal and the new behaviour is determined by the virtual actuator and deducted from the measurements of

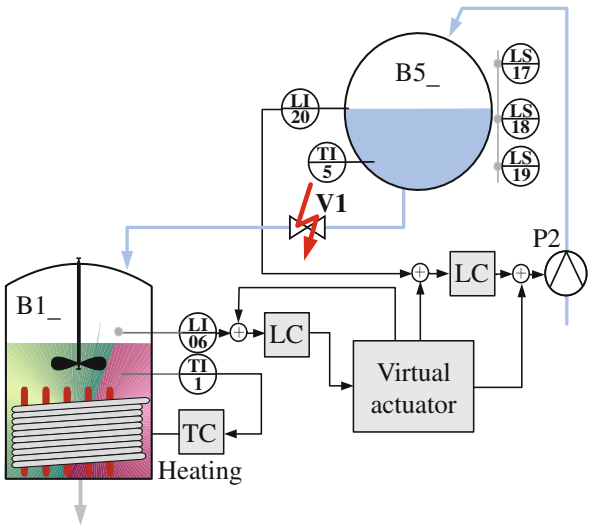


Fig. 9.19 Reconfigured controller including a virtual actuator

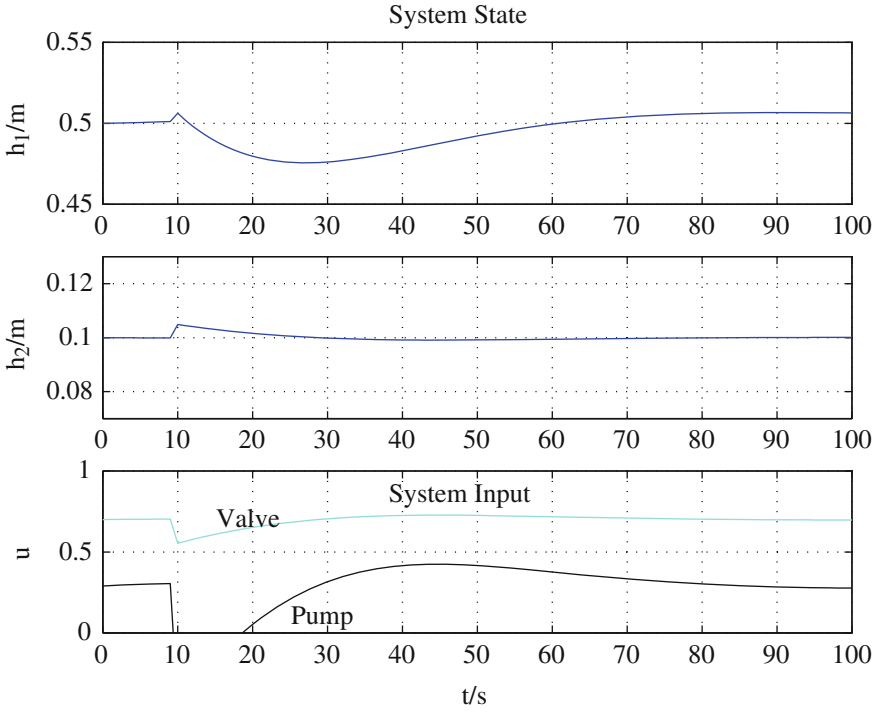


Fig. 9.20 Results of the reconfiguration experiment (Reactor temperature $\vartheta_{B1}(t)$ (*top*), reactor content $V_{B1}(t)$ (*middle*) and reactor content $V_{B5}(t)$

$V_{B1}(t)$ and $V_{B5}(t)$. In this way, the additional pole remains hidden from the level controller and this controller acts like in the nominal case.

The experimental results are shown in Fig. 9.20. The state $V_{B1}(t)$ is disturbed by withdrawing a considerable amount of liquid until time $t = 10\,\text{s}$. The virtual actuator increases the level $V_{B5}(t)$ in Tank B_5 by increasing the pump input $u_{P1}(t)$. The effect of this manipulation and of the fault is "simulated" by the virtual actuator, subtracted from the sensor's data and, therefore, hidden from the nominal controller. After 180 s the tank level $V_{B5}(t)$ reaches its maximum and after another 800 s the state deviation has been reasonably compensated. A static deviation remains because of some modelling inaccuracies.

The dashed lines show the behaviour of the faultless closed-loop system. The slower reaction of the level controller results in the slower disturbance attenuation shown in the middle part of the figure, where the nominal system reaches the set-point of $19\,\text{dm}^3$ quicker than the reconfigured system. Hence, the operation of the main reactor can be restored with a minor performance degradation.

In the lower part of the figure the different behaviour of Tank B_5 can be seen. The difference is due to the different functions that this tank has in both situations. In the faultless case the level controller of this tank adjusts the liquid content to the set-point, whereas under faulty conditions this variable is used as a means to control the inflow into Tank B_1 and, thus, to control the contents of B_1.

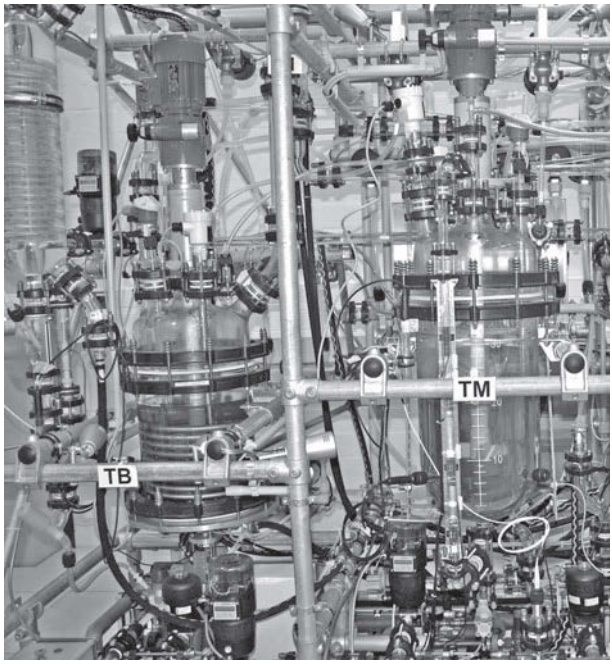


Fig. 9.21 Part of the chemical plant VERA used in the experiment

9.2.7 Experimental Evaluation: Conductivity Control Loop

The second application of the reconfiguration method that uses the virtual actuator is the fault-tolerant control of the conductivity of a liquid. Figure 9.21 shows the experimental set-up and Fig. 9.22 the schematic diagram of the three reactors involved in the control loop considered. The sequence of the two Reactors *TM* and *TB* with the Reactor *TS* is used to produce a liquid with prescribed temperature and conductivity. Several control loops have to be used, which are shown in the schematic diagram with the abbreviations *LC* for level controller, *TC* for temperature controller and *CC* for concentration controller. If actuator failures occur, these loops are brought out of operation. Typical failures concern the valve V_{CW} and the heating P_{el}.

The nominal controller uses the inputs u_{PS}, u_{TS} and u_{TB}, the three variables to be controlled are the temperature ϑ_{TB}, the liquid level l_{TS} in the Reactor *TS* and the conductivity λ_{TS} of the liquid in the Reactor *TS* (Fig. 9.23). The block diagram also shows the redundant inputs u_{CW} and u_{TM}, which will be used for the reconfiguration.

Nonlinear model. The following nonlinear model is obtained from balance equations that concern the different components of the plant. To shorten the notation of the equations, the dependency of the signals from the time t is omitted:

• Change of the liquid temperature of Reactor *TS*:

$$\dot{\vartheta}_{\mathrm{TS}} = \frac{1}{A_{\mathrm{TS}}\rho l_{\mathrm{TS}}}\left\{\frac{P_{\mathrm{el,TS}} - \dot{Q}_{PL,TS}}{c_p} + \dot{m}_{\mathrm{TB}}(\vartheta_{\mathrm{TB}} - \vartheta_{\mathrm{TS}}) + \right.$$
$$\left. \dot{m}_{\mathrm{TM}}(\vartheta_{\mathrm{TM}} - \vartheta_{\mathrm{TS}}) + \dot{m}_{\mathrm{CW}}(\vartheta_{\mathrm{CW}} - \vartheta_{\mathrm{TS}})\right\}$$

• Change of the liquid volume in Reactor *TS*:

$$\dot{l}_{\mathrm{TS}}(t) = \frac{\dot{m}_{\mathrm{TB}}(t) + \dot{m}_{\mathrm{TM}}(t) + \dot{m}_{\mathrm{CW}}(t) - \dot{m}_{\mathrm{TW}}(t)}{A_{\mathrm{TS}}\rho}$$

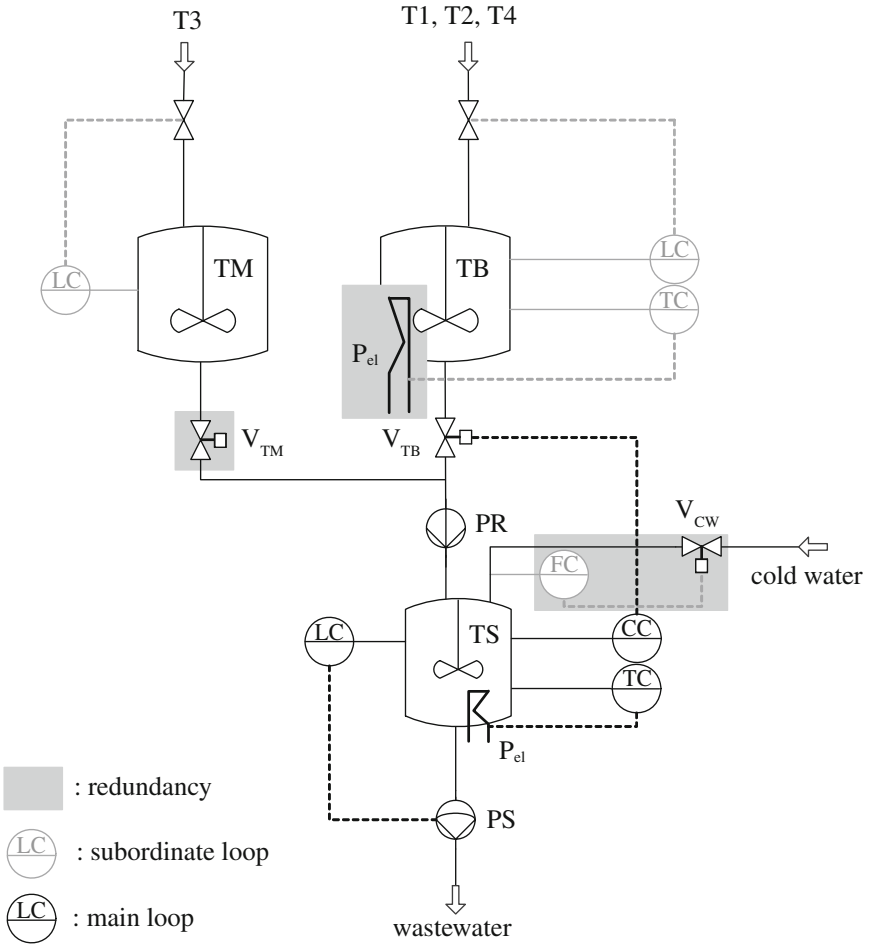


Fig. 9.22 Schematic diagram of the process

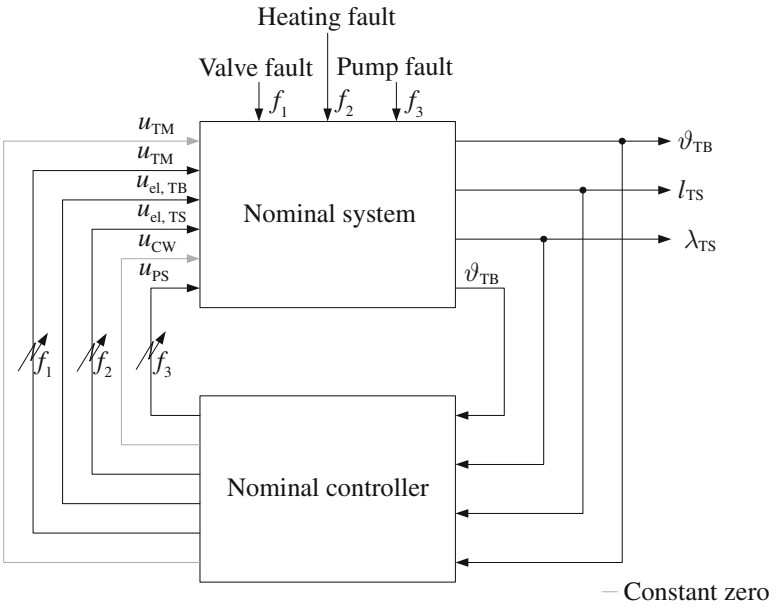


Fig. 9.23 Schematic diagram of the process

- Change of the concentration in Reactor *TS*:

$$\dot{c}_{\mathrm{TS}}(t) = \frac{\dot{m}_{\mathrm{TB}}(t)(c_{\mathrm{TB}} - c_{\mathrm{TS}}(t)) + \dot{m}_{\mathrm{TM}}(t)(c_{\mathrm{TM}} - c_{\mathrm{TS}}(t)) - \dot{m}_{\mathrm{CW}}(t)c_{\mathrm{TS}}(t)}{A_{\mathrm{TS}}\rho l_{\mathrm{TS}}(t)}$$

- Change of the liquid temperature in Reactor *TB*:

$$\dot{\vartheta}_{\mathrm{TB}}(t) = \frac{1}{A_{\mathrm{TB}}\rho l_{\mathrm{TB}}}\left\{\frac{P_{\mathrm{el,TB}}(t) - \dot{Q}_{PL,TB}(t)}{c_p} + \dot{m}_{T124}(t)(\vartheta_{T124} - \vartheta_{\mathrm{TB}}(t))\right\}$$

- Behaviour of the cold water Valve V_{CW}:

$$\dot{x}_{\mathrm{CW}}(t) = = -\frac{1}{T_{\mathrm{CW}}}x_{\mathrm{CW}}(t) + \frac{1}{T_{\mathrm{CW}}}u_{\mathrm{CW}}(t)$$
$$\dot{m}_{\mathrm{CW}}(t) = x_{\mathrm{CW}}(t) \quad \text{with} \quad T_{\mathrm{CW}} = 3,7\,\mathrm{s}$$

- Actuator dynamics of the heating of the Reactor *TB*:

$$\begin{aligned}\dot{x}_{\mathrm{TB}}(t) = &= -\frac{1}{T_{\mathrm{el,TB}}}x_{\mathrm{TB}}(t) + \frac{1}{T_{\mathrm{el,TB}}}u_{\mathrm{TB}}(t)\\ P_{\mathrm{el,TB}}(t) &= k_{\mathrm{TB}}x_{\mathrm{TB}}(t),\\ &\text{with } T_{\mathrm{el,TB}} = 27\,\mathrm{s}, \quad k_{\mathrm{TB}} = 18\,\mathrm{kW}\end{aligned}$$

• Actuator dynamics of the heating of the Reactor *TS*:

$$\begin{aligned}\dot{x}_{\mathrm{TS}}(t) = &= -\frac{1}{T_{\mathrm{el,TS}}}x_{\mathrm{TS}}(t) + \frac{1}{T_{\mathrm{el,TS}}}u_{\mathrm{TS}}(t)\\ P_{\mathrm{el,TS}}(t) &= k_{\mathrm{TS}}x_{\mathrm{TS}}(t),\\ &\text{with } T_{\mathrm{el,TS}} = 65\,\mathrm{s}, \quad k_{\mathrm{TS}} = 4\,\mathrm{kW}\end{aligned}$$

Besides the state variables ϑ_{TB} and l_{TS}, the conductivity is the third variable to be controlled. This signal is obtained by the following relation:

$$\lambda_{\mathrm{TS}}(t) = 0{,}4469\frac{\mathrm{mS}}{\mathrm{cm}} + 2047{,}7\frac{\mathrm{mS}}{\mathrm{cm}}c_{\mathrm{TS}}(t).$$

All these equations use the following mass and heat flows:

• Mass flow from Rector *TB* towards Reactor *TS*:

$$\dot{m}_{\mathrm{TB}}(t) = \begin{cases} \left(0{,}019\frac{\mathrm{kg}}{\mathrm{s}\sqrt{\mathrm{m}}} + 0{,}727\frac{\mathrm{kg}}{\mathrm{s}\sqrt{\mathrm{m}}}(u_{\mathrm{TB}}(t) - 0{,}13)\right)\sqrt{l_{\mathrm{TB}} + 0{,}3\ \mathrm{m}}, & \text{if } u_{\mathrm{TB}} \geq 0{,}13\\ 0\frac{\mathrm{kg}}{\mathrm{s}}, & \text{else}\end{cases}$$

• Mass flow from Rector *TM* towards Reactor *TS*:

$$\dot{m}_{\mathrm{TM}}(t) = \begin{cases} \left(0{,}047\ \frac{\mathrm{kg}}{\mathrm{s}\sqrt{\mathrm{m}}} + 0.605\ \frac{\mathrm{kg}}{\mathrm{s}\sqrt{\mathrm{m}}}(u_{\mathrm{TM}}(t) - 0{,}04)\right)\sqrt{l_{\mathrm{TM}} + 0.3\ \mathrm{m}} & \text{if } u_{\mathrm{TM}} \geq 0.04\\ 0\frac{\mathrm{kg}}{\mathrm{s}} & \text{else.}\end{cases}$$

• Mass flow out of the Reactor *TS*:

$$\dot{m}_{\mathrm{PS}}(t) = \dot{m}_{\mathrm{TW}}(t) = 0{,}1679\frac{\mathrm{kg}}{\mathrm{s}\sqrt{\mathrm{m}}}u_{\mathrm{PS}}(t)\sqrt{l_{\mathrm{TS}}(t) + 0{,}36\ \mathrm{m}}$$

• Heat balance of the Reactor *TS*:

$$\dot{Q}_{\mathrm{PL,TS}}(\vartheta_{\mathrm{TS}}(t)) = \begin{cases} \dot{Q}_{\mathrm{PL,TS,on}}(\vartheta_{\mathrm{TS}}(t)), & \text{if heating is on}\\ \dot{Q}_{\mathrm{PL,TS,off}}(\vartheta_{\mathrm{TS}}(t)), & \text{if heating is off}\end{cases}$$

with

$$\dot{Q}_{\mathrm{PL,TS,on}}(\vartheta_{\mathrm{TS}}(t)) = \begin{cases} 46{,}9403\,\frac{\mathrm{W}}{^\circ\mathrm{C}}(\vartheta_{\mathrm{TS}}(t) - 22{,}5\,^\circ\mathrm{C}), & \text{if } \vartheta_{\mathrm{TS}} \geq 22{,}5\,^\circ\mathrm{C} \\ 0\,\mathrm{W}, & \text{if } \vartheta_{\mathrm{TS}} < 22{,}5\,^\circ\mathrm{C} \end{cases}$$

$$\dot{Q}_{\mathrm{PL,TS,off}}(\vartheta_{\mathrm{TS}}(t)) = \begin{cases} 4{,}8968\,\frac{\mathrm{W}}{^\circ\mathrm{C}}(\vartheta_{\mathrm{TS}}(t) - 22{,}5\,^\circ\mathrm{C}), & \text{if } \vartheta_{\mathrm{TS}} \geq 22{,}5\,^\circ\mathrm{C} \\ 0\,\mathrm{W}, & \text{if } \vartheta_{\mathrm{TS}} < 22{,}5\,^\circ\mathrm{C} \end{cases}$$

• Heat balance of the Reactor *TB*:

$$\dot{Q}_{PL,TB}(\vartheta_{\mathrm{TB}}(t)) = \begin{cases} \dot{Q}_{PL,TB,on}(\vartheta_{\mathrm{TB}}(t)), & \text{if heating is on} \\ \dot{Q}_{PL,TB,off}(\vartheta_{\mathrm{TB}}(t)), & \text{if heating is off} \end{cases}$$

$$\dot{Q}_{PL,TB,on}(\vartheta_{\mathrm{TB}}(t)) = \begin{cases} 135{,}468\,\frac{\mathrm{W}}{^\circ\mathrm{C}}(\vartheta_{\mathrm{TB}}(t) - 22{,}5\,^\circ\mathrm{C}), & \text{if } \vartheta_{\mathrm{TB}} \geq 22{,}5\,^\circ\mathrm{C} \\ 0\,\mathrm{W}, & \text{if } \vartheta_{\mathrm{TB}} < 22{,}5\,^\circ\mathrm{C} \end{cases}$$

$$\dot{Q}_{PL,TB,off}(\vartheta_{\mathrm{TB}}(t)) = \begin{cases} 4{,}8968\,\frac{\mathrm{W}}{^\circ\mathrm{C}}(\vartheta_{\mathrm{TB}}(t) - 22{,}5\,^\circ\mathrm{C}), & \text{if } \vartheta_{\mathrm{TB}} \geq 22{,}5\,^\circ\mathrm{C} \\ 0\,\mathrm{W}, & \text{if } \vartheta_{\mathrm{TB}} < 22{,}5\,^\circ\mathrm{C} \end{cases}$$

The given equations can be lumped together to get a nonlinear state-space model

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{g}(\boldsymbol{x}(k), \boldsymbol{u}(k)), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ \boldsymbol{y}(k) &= \boldsymbol{h}(\boldsymbol{x}(k), \boldsymbol{u}(k)) \end{aligned}$$

with the state, input and output vectors

$$\boldsymbol{x}(t) = \begin{pmatrix} \vartheta_{\mathrm{TS}}(t) \\ l_{\mathrm{TS}}(t) \\ c_{\mathrm{TS}}(t) \\ \vartheta_{\mathrm{TB}}(t) \\ x_{\mathrm{CW}}(t) \\ x_{\mathrm{TB}}(t) \\ x_{\mathrm{TS}}(t) \end{pmatrix}, \quad \boldsymbol{u}(t) = \begin{pmatrix} u_{\mathrm{TM}}(t) \\ u_{\mathrm{TB}}(t) \\ u_{\mathrm{TB}}(t) \\ u_{\mathrm{TS}}(t) \\ u_{\mathrm{CW}}(t) \\ u_{\mathrm{PS}}(t) \end{pmatrix}, \quad \boldsymbol{y}(t) = \begin{pmatrix} \vartheta_{\mathrm{TS}}(t) \\ l_{\mathrm{TS}}(t) \\ \lambda_{\mathrm{TS}}(t) \\ \vartheta_{\mathrm{TB}}(t) \end{pmatrix}.$$

Linearised model. A linearised state-space model

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) \end{aligned}$$

is obtained from the nonlinear model with the following matrices:

$$\boldsymbol{A} = 10^{-3} \cdot \begin{pmatrix} -3,46 & 0 & 0 & 1,46 & -59,12 & 0 & 39,36 \\ 0 & -0,76 & 0 & 0 & 1,41 & 0 & 0 \\ 0 & 0 & -3,15 & 0 & -0,0034 & 0 & 0 \\ 0 & 0 & 0 & -1,34 & 0 & 157,46 & 0 \\ 0 & 0 & 0 & 0 & -270,27 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -37,03 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -15,38 \end{pmatrix}$$

$$\boldsymbol{B} = 10^{-3} \cdot \begin{pmatrix} -10,62 & 0 & 0 & 0 & 0 & 0 \\ 7,11 & 8,49 & 0 & 0 & 0 & -1,98 \\ 0,0249 & 0,0235 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 270,27 & 0 \\ 0 & 0 & 37,03 & 0 & 0 & 0 \\ 0 & 0 & 0 & 15,38 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{C} = \begin{pmatrix} 1\,0 & 0 & 0\,0\,0\,0 \\ 0\,1 & 0 & 0\,0\,0\,0 \\ 0\,0 & 2047,7 & 0\,0\,0\,0 \\ 0\,0 & 0 & 1\,0\,0\,0 \end{pmatrix}$$

$$\boldsymbol{D} = \boldsymbol{O}.$$

The set of eigenvalues of the matrix $\boldsymbol{A}$

$$\sigma = \{-0,2703;\ -0,0370;\ -0,0154;\ -0,0035;\ -0,0032;\ -0,0013;\ -0,0008\}$$

gives an impression of the dynamical properties of the plant.

Models of the faulty system. The three actuator failures cause a change of the matrix $\boldsymbol{B}$ of the linearised state-space model:

• Failure f_1 of the Valve V_{TB}, which gets the input signal u_{TB}:

$$\boldsymbol{B}_{\mathrm{f_1}} = 10^{-3} \cdot \begin{pmatrix} -10,62 & 0 & 0 & 0 & 0 & 0 \\ 7,11 & 0 & 0 & 0 & 0 & -1,98 \\ 0,0249 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 270,27 & 0 \\ 0 & 0 & 37,03 & 0 & 0 & 0 \\ 0 & 0 & 0 & 15,38 & 0 & 0 \end{pmatrix}$$

• Failure f_2 of the heating of the Reactor *TS*, which acts according to the control input u_{TS}:

$$\boldsymbol{B}_{\mathrm{f}_2} = 10^{-3} \cdot \begin{pmatrix} -10,62 & 0 & 0 & 0 & 0 & 0 \\ 7,11 & 8,49 & 0 & 0 & 0 & -1,98 \\ 0,0249 & 0,0235 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 270,27 & 0 \\ 0 & 0 & 37,03 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

• Failure f_3 of the Pump PS, which runs according to the control input u_{PS}:

$$\boldsymbol{B}_{\mathrm{f}_3} = 10^{-3} \cdot \begin{pmatrix} -10,62 & 0 & 0 & 0 & 0 & 0 \\ 7,11 & 8,49 & 0 & 0 & 0 & 0 \\ 0,0249 & 0,0235 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 270,27 & 0 \\ 0 & 0 & 37,03 & 0 & 0 & 0 \\ 0 & 0 & 0 & 15,38 & 0 & 0 \end{pmatrix}.$$

These matrices differ from the matrix $\boldsymbol{B}$ for the nominal model with respect to one column each, which is set to zero for the failed actuator.

Control reconfiguration by a virtual actuator. For all three fault cases, the virtual actuator described in Definition 9.2 is used for the control reconfiguration (Fig. 9.24). The scheme is the same in all cases, only the matrix $\boldsymbol{B}_f$, which is a parameter of the virtual actuator, differs. This shows that the control reconfiguration is completely automatic in the sense that a general reconfiguration algorithm can be applied, which adapts the effect of the nominal controller to the failure that has occurred.

The first experiment concerns the reconfiguration with the goal to retain the stability of the closed-loop system. For this task, a virtual actuator with parameter matrix $\boldsymbol{N} = \boldsymbol{O}$ is used.

In case of the failure of the Valve V_{TB}, the virtual actuator has been designed to have the following set of eigenvalues:

$$\begin{aligned} \sigma_{VA} &\stackrel{!}{=} 25\sigma \\ &= \{-6.7568; -0.9259; 0.3846; -0.0866; -0.0790; -0.0335; -0.0190\} \end{aligned} \tag{9.80}$$

This eigenvalue assignment is accomplished by the feedback matrix

$$\boldsymbol{M} = \begin{pmatrix} -12.31 & -16.05 & 77.63 & 0.15 & 5.11 & 0.40 & -3.71 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 13.39 & -0.01 & 5770 & 90.07 & -23.71 & 178.06 & 15.41 \\ 17.18 & -0.06 & 7332 & 23.26 & -31.85 & 39.14 & 25.31 \\ -1.48 & -0.01 & -642.30 & -2.04 & 2.11 & -3.43 & -1.81 \\ -130.19 & -192.04 & 239.73 & 0.75 & 18.61 & 2.21 & -12.85 \end{pmatrix}.$$

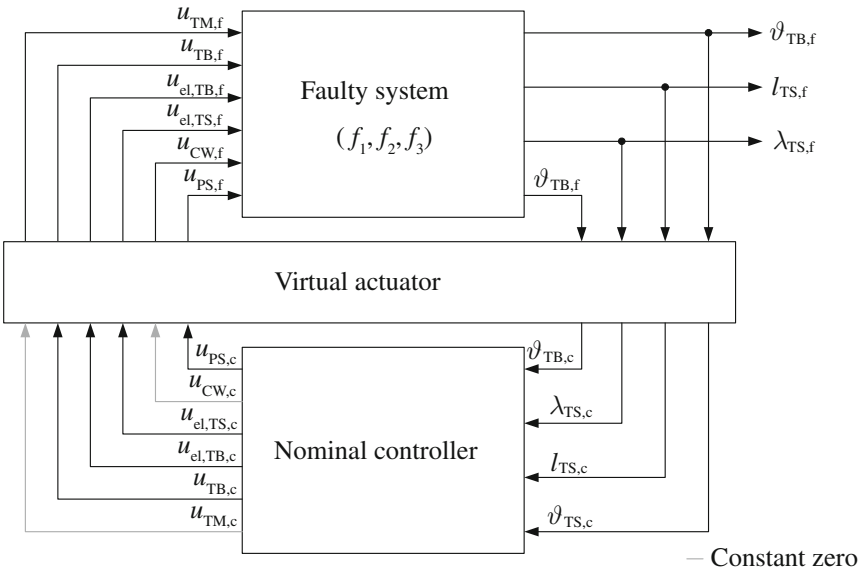


Fig. 9.24 Reconfiguration by means of a virtual actuator

It is possible, because the pair $(\boldsymbol{A}, \boldsymbol{B}_{\mathrm{fl}})$ is completely controllable. The eigenvalues are chosen with respect to the eigenvalues of the plant. They make the virtual actuator much quicker than the plant. The zero row of the matrix $\boldsymbol{M}$ ensures that the failed valve is no longer used for feedback control. Due to the separation property of the virtual actuator, the overall closed-loop system has the eigenvalues of the nominal closed-loop system and the eigenvalues given in Eq. (9.80) for the virtual actuator. Hence, the reconfigured system is stable.

Figure 9.25 approves this result. The two bars placed at time $t = 350\,\mathrm{s}$ mark the time instant at which the valve is blocked and the controller reconfigured. The temperature ϑ_{TS} and the level l_{TS} remain at the set-points, whereas the conductivity cannot follow precisely the set-point change at time $t = 300\,\mathrm{s}$ marked by the dashed line. This is due to the proportional controller used.

Figure 9.26 shows the six control inputs. After the valve V_{TB} is blocked, the signal u_{TB} shown in the top-right corner of the figure does no longer change. The virtual actuator uses the input signals u_{TS}, u_{TB} and u_{PS} which are also used by the nominal controller. In addition to this, the virtual actuator exploits the input u_{CW} to the cold water Valve V_{CW}, whereas the other additional input u_{TM} is not used.

The choice how to distribute the effect of the blocked valve over the remaining actuators is made implicitly by the virtual actuator. No selection procedure, with a possible involvement of a human control designer, is necessary. Therefore, the concept of the virtual actuator can be applied completely automatically.

The second experiment concerns the aim to bring all variables to be controlled back to their set-points. Here the "complete" virtual actuator with the two parameter

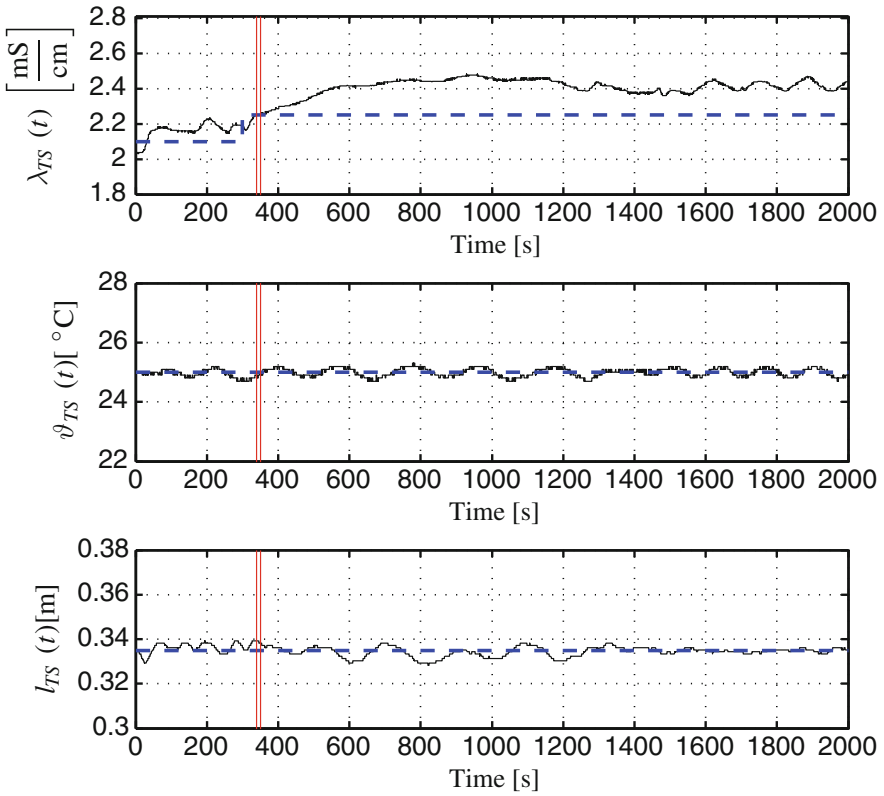


Fig. 9.25 Reconfiguration in case of the valve V_{TB}-failure with $\boldsymbol{N} = \boldsymbol{O}$

matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ is used. Besides the matrix $\boldsymbol{M}$ given above, the direct feedthrough is chosen as

$$\begin{aligned}\boldsymbol{N} &= \left(\boldsymbol{C}(\boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{M})^{-1}\boldsymbol{B}_f\right)^{-1}\left(\boldsymbol{C}(\boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{M})^{-1}\boldsymbol{B}\right)\\ &= \begin{pmatrix} 1 & 0.291 & -0.016 & 0.053 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & -0.588 & 0.031 & -0.037 & 1 & 0\\ 0 & -4.250 & -0.012 & -0.004 & 0 & 1 \end{pmatrix},\end{aligned}$$

which ensures set-point following, because the reconfigured closed-loop system has the same static reinforcement as the nominal control loop.

The reconfiguration result is depicted in Fig. 9.27. For the same experiment as before now all three control outputs are moved back to their set-points.

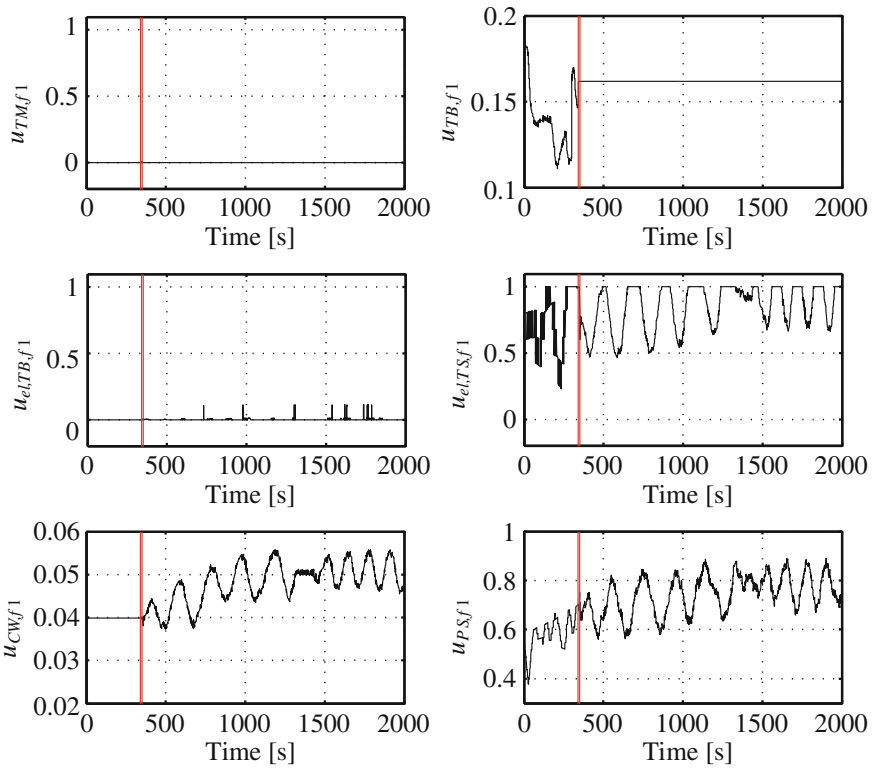


Fig. 9.26 Absolute values of the control inputs after the reconfiguration in case of the valve V_{TB}-failure

As Fig. 9.28 shows, the virtual actuator uses now the additional inputs u_{CW} and u_{TM}. The reconfiguration is completely successful including the restoration of the set-point.

9.3 Fault Recovery by Nominal Trajectory Tracking

Active fault-tolerant control implements control laws that are specific to the diagnosed fault and to the system objective to be achieved. Model-matching and the pseudo-inverse method were first introduced in flight control systems with the objective to minimise the differences between the dynamics of the healthy and the faulty systems, so as to allow pilots to keep faulty systems at hand. However, in some situations, rather than requiring the faulty system dynamics to mimic the nominal system dynamics, it is sensible to require that the faulty system follows (a best approximation of) the nominal system trajectory. Nominal trajectory tracking is of interest, for

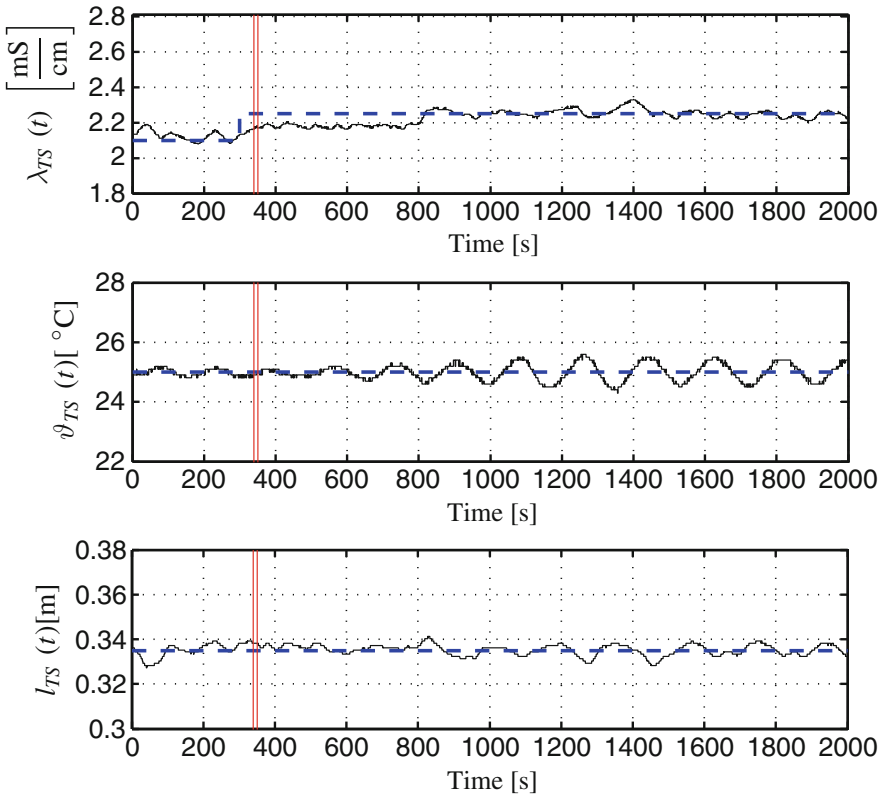


Fig. 9.27 Reconfiguration after valve V_{TB}-failure with feedthrough $\boldsymbol{N} \neq \boldsymbol{O}$

example, when unmanned vehicles are used in space missions where a rescheduling of the whole set of trajectories is impossible. Nominal trajectory tracking is the goal of the approach presented in this section.

9.3.1 Problem Setting

Nominal system. Let

$$\dot{\boldsymbol{x}}_{\text{n}}(t) = \boldsymbol{A}_{\text{n}}\boldsymbol{x}_{\text{n}}(t) + \boldsymbol{B}_{\text{n}}\boldsymbol{u}_{\text{n}}(t) \tag{9.81}$$

be the LTI model of the nominal system, where

$$\boldsymbol{u}_{\text{n}}(t) = \boldsymbol{K}_{\text{n}}\boldsymbol{x}_{\text{n}}(t) \tag{9.82}$$

is the nominal state feedback, that results in the closed-loop behaviour

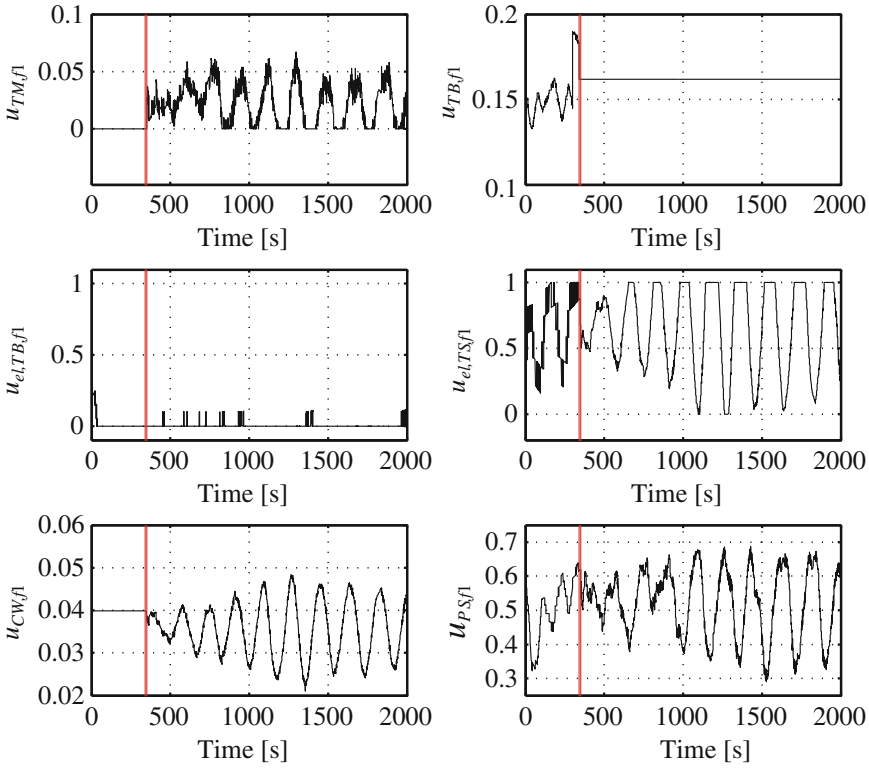


Fig. 9.28 Control input after the reconfiguration for valve V_{TM}-failure

$$\dot{\boldsymbol{x}}_{\mathrm{n}}(t) = (\boldsymbol{A}_{\mathrm{n}} + \boldsymbol{B}_{\mathrm{n}}\boldsymbol{K}_{\mathrm{n}})\boldsymbol{x}_{\mathrm{n}}(t) = \boldsymbol{M}_{\mathrm{n}}\boldsymbol{x}_{\mathrm{n}}(t), \tag{9.83}$$

where $\boldsymbol{M}_{\mathrm{n}}$ is chosen so as to satisfy some nominal requirements including stability. For example, choosing

$$\boldsymbol{M}_{\mathrm{n}} = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P},$$

where $\boldsymbol{P}$ is the solution of some Riccati equation associated with the linear quadratic problem setting gives the nominal system an optimal LQ behaviour.

Faulty system. Assume a fault occurs at time t_{f} such that the faulty system can still be described by a model associated with the pair $(\boldsymbol{A}_{\mathrm{f}},\ \boldsymbol{B}_{\mathrm{f}})$ and the control law is changed to $\boldsymbol{u}_{\mathrm{f}}(t) = \boldsymbol{K}_{\mathrm{f}}\boldsymbol{x}_{\mathrm{f}}(t)$. In fault accommodation, the pair $(\boldsymbol{A}_{\mathrm{f}},\ \boldsymbol{B}_{\mathrm{f}})$ is estimated by the fault estimation module, while in system reconfiguration, it is known from the new model that results from switching off the faulty components that have been isolated by the fault isolation module.

Let $[t_{\mathrm{f}},\ t_0[$ be the time interval during which the detection, isolation, fault estimation and accommodation takes place. The post-fault trajectory satisfies:

$$t \in [t_f,\ t_0[\ :\ \dot{\boldsymbol{x}}_f(t) = (\boldsymbol{A}_f + \boldsymbol{B}_f \boldsymbol{K}_n)\boldsymbol{x}_f(t)$$
$$t \geq t_0\ :\ \dot{\boldsymbol{x}}_f(t) = (\boldsymbol{A}_f + \boldsymbol{B}_f \boldsymbol{K}_f)\boldsymbol{x}_f(t) = \boldsymbol{M}_f \boldsymbol{x}_f(t)$$

It follows that for $t \geq t_0$ the trajectory of the accommodated system is given by

$$\boldsymbol{x}_f(t) = \boldsymbol{\Phi}_f(t - t_0)\boldsymbol{x}_f(t)(t_0)$$

while the trajectory of the nominal system would have been

$$\boldsymbol{x}_n(t) = \boldsymbol{\Phi}_n(t - t_0)\boldsymbol{x}_n(t_0)$$

with $\boldsymbol{\Phi}_i(t - t_0) = \mathrm{e}^{\boldsymbol{M}_i(t-t_0)}$, $(i = f, n)$. While the model-matching approach is concerned with the difference $\boldsymbol{M}_n - \boldsymbol{M}_f$, the nominal trajectory tracking considers the difference $\boldsymbol{\Phi}_n(t-t_0) - \boldsymbol{\Phi}_f(t-t_0)$, and requests the trajectory of the accommodated system to mimic as closely as possible the trajectory of the nominal one in an attempt to rub out the effect of the fault.

Introducing two symmetric matrices $\boldsymbol{Q} \geq 0$, and $\boldsymbol{R} > 0$ and measuring the closeness of the trajectories by means of the quadratic cost

$$J = \frac{1}{2}\int_{t_0}^{\infty} (\boldsymbol{x}_f(t) - \boldsymbol{x}_n(t))^{\mathrm{T}}\, \boldsymbol{Q}\, (\boldsymbol{x}_f(t) - \boldsymbol{x}_n(t)) + (\boldsymbol{u}_f(t) - \boldsymbol{u}_n(t))^{\mathrm{T}}\, \boldsymbol{R}\, (\boldsymbol{u}_f(t) - \boldsymbol{u}_n(t))\ \mathrm{d}t \tag{9.84}$$

provides a problem setting that allows to achieve a compromise between the discrepancies of the accommodated to nominal trajectory and the accommodated to nominal control signal.

9.3.2 Solution

Optimality condition. From the classical theory of optimal control, one gets the following set of necessary conditions

$$\dot{\boldsymbol{x}}_f(t) = \boldsymbol{A}_f \boldsymbol{x}_f(t) + \boldsymbol{B}_f \boldsymbol{u}_f(t) \tag{9.85}$$

$$\dot{\boldsymbol{p}}_f(t) = \boldsymbol{Q}\, (\boldsymbol{x}_f(t) - \boldsymbol{x}_n(t)) - \boldsymbol{A}_f^{\mathrm{T}} \boldsymbol{p}_f(t) \tag{9.86}$$

$$\boldsymbol{O} = \boldsymbol{R}\, (\boldsymbol{u}_f(t) - \boldsymbol{K}_n \boldsymbol{x}_n(t)) - \boldsymbol{B}_f^{\mathrm{T}} \boldsymbol{p}_f(t), \tag{9.87}$$

where $\boldsymbol{p}_f$ is the adjoint state vector. From (9.87), the accommodated control is

$$\boldsymbol{u}_f(t) = \boldsymbol{K}_n \boldsymbol{x}_n(t) + \boldsymbol{R}^{-1} \boldsymbol{B}_f^{\mathrm{T}} \boldsymbol{p}_f(t). \tag{9.88}$$

Following a classical approach, the adjoint state is assumed to have the form

$$\boldsymbol{p}_{\mathrm{f}}(t) = \boldsymbol{H}\boldsymbol{x}_{\mathrm{f}}(t) + \boldsymbol{G}\boldsymbol{x}_{\mathrm{n}}(t),$$

where $\boldsymbol{H}$ and $\boldsymbol{G}$ are two matrices to be determined. Making use of (9.81), (9.85) and (9.88) one gets

$$\begin{aligned}\dot{\boldsymbol{p}}_{\mathrm{f}}(t) &= \boldsymbol{H}\left(\boldsymbol{A}_{\mathrm{f}} + \boldsymbol{B}_{\mathrm{f}}\boldsymbol{R}^{-1}\boldsymbol{B}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{H}\right)\boldsymbol{x}_{\mathrm{f}}(t)\\ &+ \left(\boldsymbol{H}\boldsymbol{B}_{\mathrm{f}}\boldsymbol{K}_{\mathrm{n}} + \boldsymbol{B}_{\mathrm{f}}\boldsymbol{R}^{-1}\boldsymbol{B}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{G} + \boldsymbol{G}\left(\boldsymbol{A}_{\mathrm{n}} + \boldsymbol{B}_{\mathrm{n}}\boldsymbol{K}_{\mathrm{n}}\right)\right)\boldsymbol{x}_{\mathrm{n}}(t)\end{aligned}$$

From (9.86) it follows that

$$\dot{\boldsymbol{p}}_{\mathrm{f}}(t) = \left(\boldsymbol{Q} - \boldsymbol{A}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{H}\right)\boldsymbol{x}_{\mathrm{f}}(t) - \left(\boldsymbol{Q} + \boldsymbol{A}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{G}\right)\boldsymbol{x}_{\mathrm{n}}(t)$$

holds and, therefore,

$$\begin{aligned}\left(\boldsymbol{Q} - \boldsymbol{A}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{H} - \boldsymbol{H}\boldsymbol{A}_{\mathrm{f}} - \boldsymbol{H}\boldsymbol{B}_{\mathrm{f}}\boldsymbol{R}^{-1}\boldsymbol{B}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{H}\right)\boldsymbol{x}_{\mathrm{f}}(t) &= (+\boldsymbol{H}\boldsymbol{B}_{\mathrm{f}}\boldsymbol{K}_{\mathrm{n}} + \cdots\\ \cdots + \boldsymbol{H}\boldsymbol{B}_{\mathrm{f}}\boldsymbol{R}^{-1}\boldsymbol{B}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{G} + \boldsymbol{G}\left(\boldsymbol{A}_{\mathrm{n}} + \boldsymbol{B}_{\mathrm{n}}\boldsymbol{K}_{\mathrm{n}}\right) + \boldsymbol{A}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{G})\boldsymbol{x}_{\mathrm{n}}(t)\end{aligned}$$

so that $\boldsymbol{H}$ and $\boldsymbol{G}$ must satisfy the relations

$$\boldsymbol{A}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{H} + \boldsymbol{H}\boldsymbol{A}_{\mathrm{f}} + \boldsymbol{H}\boldsymbol{B}_{\mathrm{f}}\boldsymbol{R}^{-1}\boldsymbol{B}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{H} - \boldsymbol{Q} = \boldsymbol{O} \tag{9.89}$$

$$\boldsymbol{Q} + \boldsymbol{H}\boldsymbol{B}_{\mathrm{f}}\boldsymbol{K}_{\mathrm{n}} + \boldsymbol{G}\left(\boldsymbol{A}_{\mathrm{n}} + \boldsymbol{B}_{\mathrm{n}}\boldsymbol{K}_{\mathrm{n}}\right) + \left(\boldsymbol{H}\boldsymbol{B}_{\mathrm{f}}\boldsymbol{R}^{-1}\boldsymbol{B}_{\mathrm{f}}^{\mathrm{T}} + \boldsymbol{A}_{\mathrm{f}}^{\mathrm{T}}\right)\boldsymbol{G} = \boldsymbol{O}. \tag{9.90}$$

Equation (9.89) is a classical algebraic Riccati equation and (9.90) a Lyapunov equation that is easily solved once $\boldsymbol{H}$ has been found.

Stability. From (9.87) one gets

$$\boldsymbol{u}_{\mathrm{f}}(t) = \boldsymbol{u}_{\mathrm{n}}(t) + \boldsymbol{R}^{-1}\boldsymbol{B}_{\mathrm{f}}^{\mathrm{T}}\left(\boldsymbol{H}\boldsymbol{x}_{\mathrm{f}}(t) + \boldsymbol{G}\boldsymbol{x}_{\mathrm{n}}(t)\right) \tag{9.91}$$

and, therefore, the accommodated control is obtained by adding the compensating term $\boldsymbol{R}^{-1}\boldsymbol{B}_{\mathrm{f}}^{\mathrm{T}}\left(\boldsymbol{H}\boldsymbol{x}_{\mathrm{f}}(t) + \boldsymbol{G}\boldsymbol{x}_{\mathrm{n}}(t)\right)$ to the nominal control, leading to the accommodated dynamics:

$$\dot{\boldsymbol{x}}_{\mathrm{f}} = \left(\boldsymbol{A}_{\mathrm{f}} + \boldsymbol{B}_{\mathrm{f}}\boldsymbol{R}^{-1}\boldsymbol{B}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{H}\right)\boldsymbol{x}_{\mathrm{f}}(t) + \boldsymbol{B}_{\mathrm{f}}\left(\boldsymbol{K}_{\mathrm{n}} + \boldsymbol{R}^{-1}\boldsymbol{B}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{G}\right)\boldsymbol{x}_{\mathrm{n}}(t). \tag{9.92}$$

Let $\boldsymbol{z}^{\mathrm{T}}(t) = \left(\boldsymbol{x}_{\mathrm{n}}(t)^{\mathrm{T}}\ \boldsymbol{x}_{\mathrm{f}}(t)^{\mathrm{T}}\right)$. Then from Eqs. (9.83) and (9.92) one gets $\dot{\boldsymbol{z}}(t) = \boldsymbol{M}\boldsymbol{z}(t)$ with

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{A}_\mathrm{n} + \boldsymbol{B}_\mathrm{n}\boldsymbol{K}_\mathrm{n} & \boldsymbol{O} \\ \boldsymbol{B}_\mathrm{f}\left(\boldsymbol{K}_\mathrm{n} + \boldsymbol{R}^{-1}\boldsymbol{B}_\mathrm{f}^\mathrm{T}\boldsymbol{G}\right) & \boldsymbol{A}_\mathrm{f} + \boldsymbol{B}_\mathrm{f}\boldsymbol{R}^{-1}\boldsymbol{B}_\mathrm{f}^\mathrm{T}\boldsymbol{H} \end{pmatrix}.$$

Since $\boldsymbol{K}_\mathrm{n}$ is chosen such that the nominal closed-loop matrix $\boldsymbol{A}_\mathrm{n}+\boldsymbol{B}_\mathrm{n}\boldsymbol{K}_\mathrm{n}$ is stable, the stability of the accommodated system follows from the stability of $\boldsymbol{A}_\mathrm{f}+\boldsymbol{B}_\mathrm{f}\boldsymbol{R}^{-1}\boldsymbol{B}_\mathrm{f}^\mathrm{T}H$, which is well known to be achieved by a unique solution $\boldsymbol{H}$ provided that the pair $(\boldsymbol{A}_\mathrm{f}, \boldsymbol{B}_\mathrm{f})$ is still controllable and that the pair $(\boldsymbol{C}, \boldsymbol{A}_\mathrm{f})$ is observable with $\boldsymbol{Q} = \boldsymbol{C}^\mathrm{T}\boldsymbol{C}$.

Admissibility. Let $(\boldsymbol{A}_\mathrm{f}, \boldsymbol{B}_\mathrm{f})$ be a fault such that $(\boldsymbol{A}_\mathrm{f}, \boldsymbol{B}_\mathrm{f})$ is controllable and $(\boldsymbol{C}, \boldsymbol{A}_\mathrm{f})$ is observable, then there exists a unique pair $(\boldsymbol{H}, \boldsymbol{G})$ such that the accommodated control $\boldsymbol{u}_\mathrm{f}(t) = \boldsymbol{u}_\mathrm{n}(t)+\boldsymbol{R}^{-1}\boldsymbol{B}_\mathrm{f}^\mathrm{T}\left(\boldsymbol{H}\boldsymbol{x}_\mathrm{f}(t) + \boldsymbol{G}\boldsymbol{x}_\mathrm{n}(t)\right)$ stabilises the faulty system and is optimal with respect to the cost (9.84). However, not any such fault is recoverable, because although minimal, the cost (9.84) might be too high for the accommodated behaviour to be accepted as close enough to the nominal one.

Let $\epsilon_s(t) = \boldsymbol{x}_\mathrm{f}(t) - \boldsymbol{x}_\mathrm{n}(t)$ and $\epsilon_u(t) = \boldsymbol{u}_\mathrm{f}(t) - \boldsymbol{u}_\mathrm{n}(t)$ be the differences between the faulty and the nominal system behaviour. Using Eq. (9.91) one gets

$$\epsilon_s^\mathrm{T}(t)\,\boldsymbol{Q}\epsilon_s(t) + \epsilon_u^\mathrm{T}(t)\boldsymbol{R}\epsilon_u(t) = z^\mathrm{T}(t)\boldsymbol{S}z(t)$$

with

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{Q} + \boldsymbol{G}^\mathrm{T}\boldsymbol{B}_\mathrm{f}\boldsymbol{R}^{-1}\boldsymbol{B}_\mathrm{f}^\mathrm{T}\boldsymbol{G} & -\boldsymbol{Q} + \boldsymbol{G}^\mathrm{T}\boldsymbol{B}_\mathrm{f}\boldsymbol{R}^{-1}\boldsymbol{B}_\mathrm{f}^\mathrm{T}\boldsymbol{H} \\ -\boldsymbol{Q} + \boldsymbol{H}^\mathrm{T}\boldsymbol{B}_\mathrm{f}\boldsymbol{R}^{-1}\boldsymbol{B}_\mathrm{f}^\mathrm{T}\boldsymbol{G} & \boldsymbol{Q} + \boldsymbol{H}^\mathrm{T}\boldsymbol{B}_\mathrm{f}\boldsymbol{R}^{-1}\boldsymbol{B}_\mathrm{f}^\mathrm{T}\boldsymbol{H} \end{pmatrix}.$$

The cost can now be easily computed. Since $\boldsymbol{M}$ is stable, there is a symmetric negative definite matrix $\boldsymbol{P}$ such that

$$\boldsymbol{M}^\mathrm{T}\boldsymbol{P} + \boldsymbol{P}\boldsymbol{M} = \boldsymbol{S}$$

It follows that

$$\frac{\mathrm{d}}{\mathrm{d}t}z^\mathrm{T}(t)\boldsymbol{P}z(t) = z^\mathrm{T}(t)\boldsymbol{S}z(t)$$

and

$$J = \frac{1}{2}\int_{t_0}^{\infty}\frac{\mathrm{d}}{\mathrm{d}t}z^\mathrm{T}(t)\boldsymbol{P}z(t)\mathrm{d}t = -\frac{1}{2}z^\mathrm{T}(t_0)\boldsymbol{P}z(t_0).$$

As already seen, different admissibility conditions can be stated. For example, one can define a constant admissibility limit η, resulting in recoverable faults that satisfy the inequality

$$-\frac{1}{2}z^\mathrm{T}(t_0)\boldsymbol{P}z(t_0) \leq \eta \tag{9.93}$$

or a quadratic admissibility limit $-\frac{1}{2}z^{\mathrm{T}}(t_0)\boldsymbol{P}_{\min}z(t_0)$ resulting in the recoverable faults if

$$\boldsymbol{P} - \boldsymbol{P}_{\min} \geq 0 \tag{9.94}$$

holds.

Remark 9.2 The state discrepancy in the time window $[t_{\mathrm{f}}, t_0[$ is not taken into account in the cost (9.84) since it depends on the fault only and the control has not yet been accommodated.

Remark 9.3 (*Recoverability versus accommodation delay*) The larger the fault the larger is the initial state difference $\epsilon_s(t_0) = \boldsymbol{x}_{\mathrm{f}}(t_0) - \boldsymbol{x}_{\mathrm{n}}(t_0)$. It follows that, depending on the admissibility condition that is defined, the recoverability of a fault depends on the fault itself (for example faults that result in the loss of controllability of unstable systems are not recoverable), but also partly on the delay introduced by the diagnosis and accommodation processes, as Eq. (9.93) suggests.

Remark 9.4 The model-matching approach can in no case provide any optimal solution to the trajectory tracking problem, since it results in the trajectories $\dot{\boldsymbol{x}}_{\mathrm{f}}(t) = \boldsymbol{M}_{\mathrm{f}}\boldsymbol{x}_{\mathrm{f}}(t)$ obtained by synthesising a matrix $\boldsymbol{M}_{\mathrm{f}}$ closest to $\boldsymbol{M}_{\mathrm{n}}$. Whatever way $\boldsymbol{M}_{\mathrm{f}}$ is computed, the input $\boldsymbol{x}_{\mathrm{n}}(t)$ is never taken into account, as (9.91) shows it should be. □

Example 9.5 Nominal trajectory tracking
Consider, a set of second order systems $(\boldsymbol{A},\ \boldsymbol{B}(\theta))$, where $\theta \in [0, 1]$ is a parameter, such that $\theta = 0$ characterises the nominal system $\boldsymbol{A}_{\mathrm{n}} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$, $\boldsymbol{B}_{\mathrm{n}} = \begin{pmatrix} 1 \\ 5 \end{pmatrix}$ and $\theta > 0$ is associated with the faulty system $\boldsymbol{A}_{\mathrm{f}} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$, $\boldsymbol{B}_{\mathrm{f}} = \begin{pmatrix} 1 - 2\theta \\ 5 - 4\theta \end{pmatrix}$. Note that for $\theta = 0.5$ the faulty system is not controllable.

Under the state feedback control $u = k_1 x_1 + k_2 x_2$ the closed-loop matrix is

$$\boldsymbol{M}(k_1, k_2, \theta) = \begin{pmatrix} k_1(1 - 2\theta) - 1 & k_2(1 - 2\theta) \\ k_1(5 - 4\theta) & k_2(5 - 4\theta) - 1 \end{pmatrix}.$$

Assuming that the control objective is to obtain the behaviour associated with the reference model $\boldsymbol{M}_{\mathrm{n}} = \begin{pmatrix} -2 & 0 \\ -5 & -1 \end{pmatrix}$, the pseudo-inverse method results in the feedback gains:

$$\begin{aligned} k_1(\theta) &= \frac{22\theta - 26}{(1 - 2\theta)^2 + (5 - 4\theta)^2} \\ k_2(\theta) &= 0. \end{aligned}$$

It is easily seen that the Frobenius norm $\|\boldsymbol{M}(k_1, k_2, \theta) - \boldsymbol{M}_{\mathrm{n}}\|_{\mathrm{F}}$ can be zeroed to obtain an exact model-matching result only in the nominal case $\theta = 0$ and this minimum is associated with the nominal control $\boldsymbol{u}_{\mathrm{n}}(t) = (-1 \ \ 0)\boldsymbol{x}_{\mathrm{n}}(t)$. For $\theta \neq 0$, the pseudo-inverse method results in the closed-loop matrix:

$$\boldsymbol{M}_{\mathrm{f}}^{\mathrm{PIM}}(\theta)=\begin{pmatrix}\frac{-64\theta^2+118\theta-52}{20\theta^2-44\theta+26} & 0\\ \frac{-88\theta^2+214\theta-130}{20\theta^2-44\theta+26} & -1\end{pmatrix}$$

whose eigenvalues are

$$\lambda_1(\theta)=-1\ \lambda_2(\theta)=\frac{-64\theta^2+118\theta-52}{20\theta^2-44\theta+26}.$$

It can be checked that the pseudo-inverse method provides an unstable solution for all faults wiht $\theta > 0.728$.

Let us now investigate the nominal trajectory tracking approach, using $\boldsymbol{Q}=\boldsymbol{I}_2$ and $\boldsymbol{R}=1$. The post-fault optimal control is obtained as $\boldsymbol{u}_{\mathrm{f}}(t)=\boldsymbol{H}\boldsymbol{x}_{\mathrm{f}}(t)+\boldsymbol{G}\boldsymbol{x}_{\mathrm{n}}(t)$ provided that $\theta\neq 0.5$. With

$$\boldsymbol{W}(\theta)=\begin{pmatrix}(1-2\theta)^2 & (1-2\theta)(5-4\theta)\\ (5-4\theta)(1-2\theta) & (5-4\theta)^2\end{pmatrix}$$

the matrix $\boldsymbol{H}$ is given by

$$\boldsymbol{H}\boldsymbol{W}(\theta)\boldsymbol{H}-2\boldsymbol{H}-\boldsymbol{I}_2=\boldsymbol{O}$$

while $\boldsymbol{G}$ is the solution of

$$\boldsymbol{I}_2+\boldsymbol{H}\begin{pmatrix}-(1-2\theta) & 0\\ -(5-4\theta) & 0\end{pmatrix}+\boldsymbol{G}\begin{pmatrix}-2 & 0\\ -5 & -1\end{pmatrix}+(\boldsymbol{W}(\theta)-\boldsymbol{I}_2)\boldsymbol{G}=\boldsymbol{O}.$$

Stable case. Let us first illustrate the case where the pseudo-inverse method (PIM) provides a stable closed loop by assuming the fault $\theta = 0.6$. The PIM solution is

$$\boldsymbol{u}_{\mathrm{f}}(t)^{\mathrm{PIM}}=-1.88235\boldsymbol{x}_1^{\mathrm{PIM}}$$

which gives the closed-loop matrix

$$\boldsymbol{M}_{\mathrm{f}}^{\mathrm{PIM}}=\begin{pmatrix}-0.6235 & 0\\ -4.8941 & -1\end{pmatrix}.$$

The nominal trajectory tracking optimal control $\boldsymbol{u}_{\mathrm{f}}(t)$ is defined by the pair

$$\boldsymbol{H}=\begin{pmatrix}-0.4986 & -0.0181\\ -0.0181 & -0.2650\end{pmatrix},\quad \boldsymbol{G}=\begin{pmatrix}0.2789 & 0.0181\\ -0.1258 & -0.2650\end{pmatrix}$$

that satisfies the equations

$$\boldsymbol{H}\begin{pmatrix}0.04 & -0.52\\ -0.52 & 6.76\end{pmatrix}\boldsymbol{H}-2\boldsymbol{H}-\boldsymbol{I}_2=\boldsymbol{O}$$

$$\boldsymbol{H}\begin{pmatrix}0.2 & 0\\ -2.6 & 0\end{pmatrix}+\boldsymbol{I}_2+\boldsymbol{G}\begin{pmatrix}-2 & 0\\ -5 & -1\end{pmatrix}+\begin{pmatrix}-0.96 & -0.52\\ -0.52 & 5.76\end{pmatrix}\boldsymbol{G}=\boldsymbol{O}.$$

Figure 9.29 shows the nominal $\boldsymbol{x}_{\mathrm{n}}(t)$, PIM $\boldsymbol{x}_{\mathrm{f}}^{\mathrm{PIM}}(t)$ and NTT (nominal trajectory tracking) $\boldsymbol{x}_{\mathrm{f}}(t)$ state trajectories, assuming that during the first 2 s, the faulty system is still controlled by the nominal control. As a result, the $\boldsymbol{x}_{\mathrm{f}}^{\mathrm{PIM}}(t)$ and $\boldsymbol{x}_{\mathrm{f}}(t)$ trajectories are identical for $t \in [10, 12[$, and $\boldsymbol{x}_{\mathrm{f}}(t)$ shows a behaviour closer to $\boldsymbol{x}_{\mathrm{n}}(t)$ only after $t = 12$. Figure 9.30 shows the significant improvements in the quadratic costs associated with the discrepancies $\left(\boldsymbol{x}_{\mathrm{n}}(t) - \boldsymbol{x}_{\mathrm{f}}^{\mathrm{PIM}}(t),\ \boldsymbol{u}_{\mathrm{n}}(t) - \boldsymbol{u}_{\mathrm{f}}(t)^{\mathrm{PIM}}\right)$ and $(\boldsymbol{x}_{\mathrm{n}}(t) - \boldsymbol{x}_{\mathrm{f}}(t),\ \boldsymbol{u}_{\mathrm{n}}(t) - \boldsymbol{u}_{\mathrm{f}}(t))$ again for a 2 s delay.

Unstable case. Let us now consider the case $\theta = 1$ in which the PIM control $\boldsymbol{u}_{\mathrm{f}}(t)^{\mathrm{PIM}} = -2\boldsymbol{x}_1$ gives the closed-loop matrix $\boldsymbol{M}_{\mathrm{f}}^{\mathrm{PIM}} = \begin{pmatrix} 1 & 0 \\ -2 & -1 \end{pmatrix}$, which is unstable. The modified PIM approach gives the control law $\boldsymbol{u}_{\mathrm{f}}(t)^{\mathrm{MPIM}} = -0.8\boldsymbol{x}_1$ which results in the stable matrix

$$\boldsymbol{M}_{\mathrm{f}}^{\mathrm{MPIM}} = \begin{pmatrix} -0.2 & 0 \\ -0.8 & -1 \end{pmatrix}.$$

The nominal control $\boldsymbol{u}_{\mathrm{f}}(t)$ is defined by the pair

$$\boldsymbol{H} = \begin{pmatrix} -0.433 & -0.067 \\ -0.067 & -0.433 \end{pmatrix}, \quad \boldsymbol{G} = \begin{pmatrix} 2.134 & 0.5 \\ 1.8 & -0.5 \end{pmatrix}$$

that satisfies

$$\boldsymbol{H}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}\boldsymbol{H} - 2\boldsymbol{H} - \boldsymbol{I}_2 = \boldsymbol{O}$$

$$\boldsymbol{H}\begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix} + \boldsymbol{G}\begin{pmatrix} -2 & 0 \\ -5 & -1 \end{pmatrix} + \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}\boldsymbol{G} + \boldsymbol{I}_2 = \boldsymbol{O}.$$

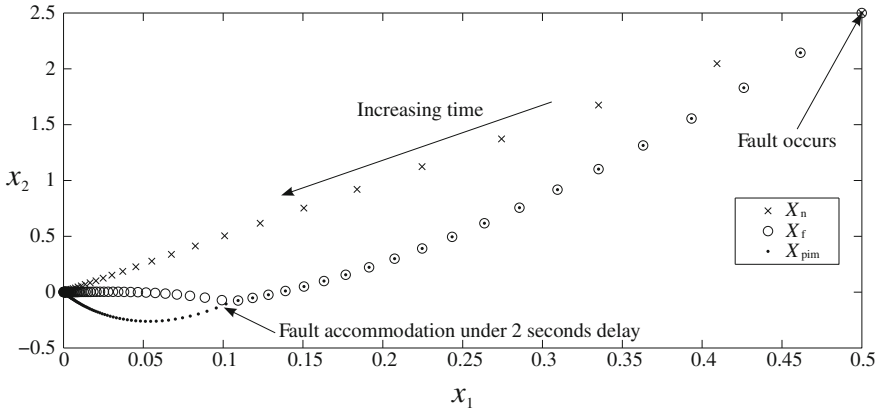


Fig. 9.29 Nominal, PIM and NTT state trajectories

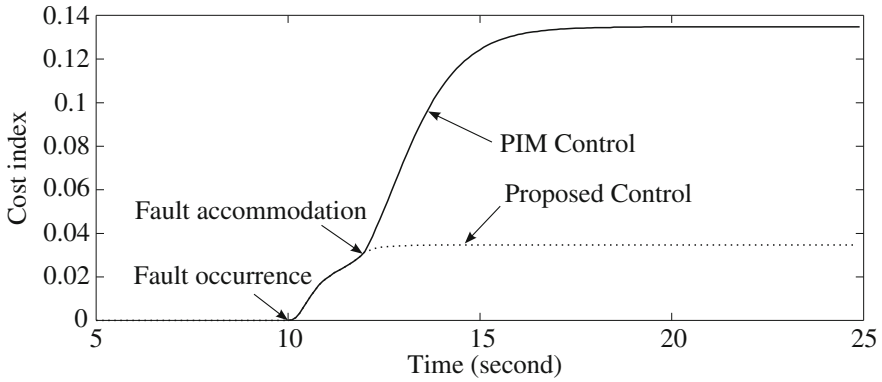


Fig. 9.30 PIM versus NTT costs

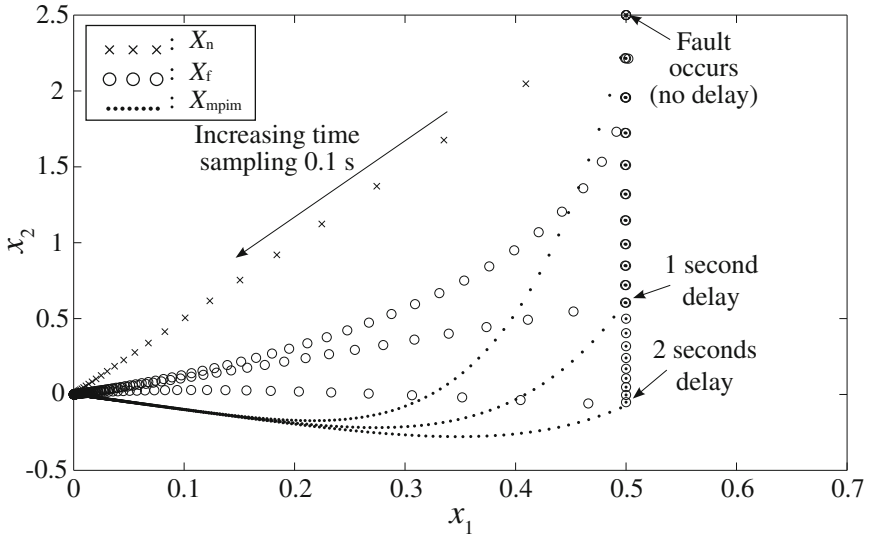


Fig. 9.31 Nominal, MPIM and NTT state trajectories

Figure 9.31 shows the state $\boldsymbol{x}_{\mathrm{n}}(t)$, modified PIM $\boldsymbol{x}_{\mathrm{f}}(t)^{\mathrm{MPIM}}$ and nominal trajectory tracking $\boldsymbol{x}_{\mathrm{f}}(t)$ trajectories, for three different fault detection, isolation, diagnosis and accommodation delays, while Fig. 9.32 shows the quadratic costs associated with the discrepancies $\left(\boldsymbol{x}_{\mathrm{n}}(t)-\boldsymbol{x}_{\mathrm{f}}(t)^{\mathrm{MPIM}}, \boldsymbol{u}_{\mathrm{n}}(t)-\boldsymbol{u}_{\mathrm{f}}(t)^{\mathrm{MPIM}}\right)$ and $(\boldsymbol{x}_{\mathrm{n}}(t)-\boldsymbol{x}_{\mathrm{f}}(t), \boldsymbol{u}_{\mathrm{n}}(t)-\boldsymbol{u}_{\mathrm{f}}(t))$ for the 2 s delay case.

In order to illustrate Remark 9.3, Fig. 9.33 shows how the trajectory tracking cost increases with the diagnosis and accommodation delay. It follows that for small delays the fault may be recoverable, while it becomes non-recoverable for larger ones, because the cost becomes inadmissible. □

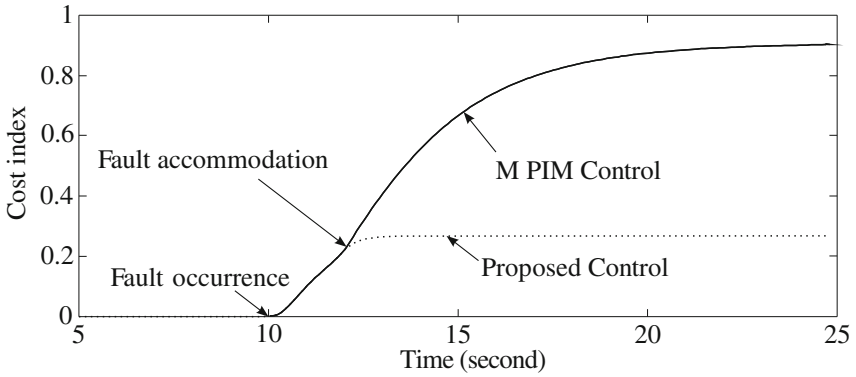


Fig. 9.32 MPIM versus NTT costs

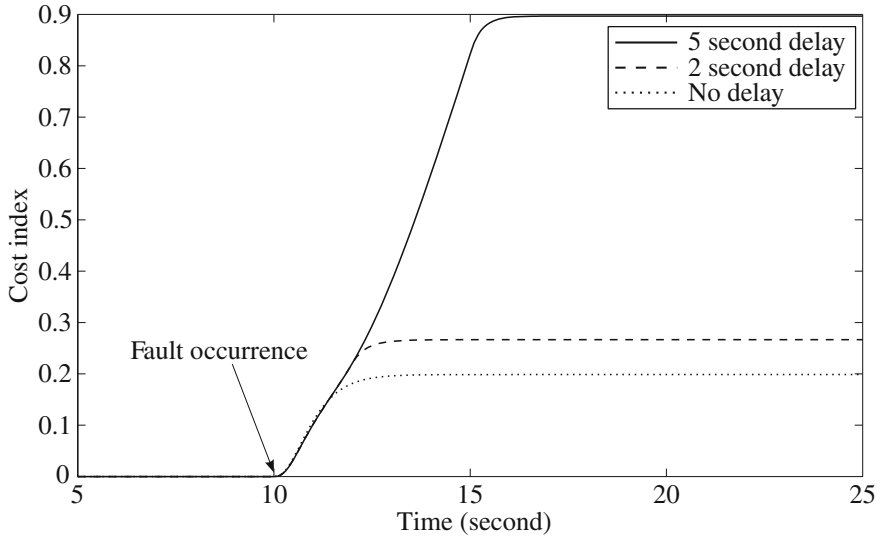


Fig. 9.33 Trajectory tracking cost versus fault accommodation delay

9.4 Fault-Tolerant $\mathcal{H}_\infty$ Design

This section introduces fault-tolerant control strategies that can be applied in a general fault case. It starts with a characterisation of all controllers that stabilise a linear system and also satisfy $\mathcal{H}_2$ or $\mathcal{H}_\infty$ norm conditions. This characterisation makes it possible to evaluate the severity of the fault with respect to the control aims and to find methods for redesign the controller automatically.

The complete description of all stabilising controllers is given by the Youla-Kucera or Q-parametrisation. The Youla parametrisation was originally defined by

using coprime factorisation. However, it is simple to give an equivalent description of the Youla-Kucera parametrisation as a state-space formulation, where the parametrisation turns out to be an observer-based controller.

One of the facilities by using this parametrisation is that the closed-loop transfer function turns out to be affine in the controller parameters. This affine structure is very useful in connection with design of controllers using optimisation methods. Therefore, the method also fits well to solve the control problem arising when we wish to make a redesign for a controller when a system is in a faulty state.

The salient feature offered by the Youla-Kucera parametrisation is that it offers an elegant and very fast solution to the redesign problem for some classes of faults that leave the system stable with the existing controller but make it unable to meet the required performance.

9.4.1 System Description

Consider the plant

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}_1\boldsymbol{w}(t) + \boldsymbol{B}_2\boldsymbol{u}(t) \\ \boldsymbol{z}(t) &= \boldsymbol{C}_1\boldsymbol{x}(t) + \boldsymbol{D}_{11}\boldsymbol{w}(t) + \boldsymbol{D}_{12}\boldsymbol{u}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}_2\boldsymbol{x}(t) + \boldsymbol{D}_{21}\boldsymbol{w}(t) + \boldsymbol{D}_{22}\boldsymbol{u}(t), \end{aligned} \tag{9.95}$$

where $\boldsymbol{x} \in |\mathcal{R}^n$ is the state, $\boldsymbol{u} \in |\mathcal{R}^r$, is the control input, $\boldsymbol{w} \in |\mathcal{R}^k$ is the external input or disturbance, $\boldsymbol{z} \in |\mathcal{R}^l$ is the controlled output and $\boldsymbol{y} \in |\mathcal{R}^m$ is the measurement output. For brevity, it is common to denote this system by the shorter notation

$$\boldsymbol{G}(s) = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{C}_1 & \boldsymbol{C}_2 \\ \boldsymbol{B}_1 & \boldsymbol{D}_{11} & \boldsymbol{D}_{21} \\ \boldsymbol{B}_2 & \boldsymbol{D}_{12} & \boldsymbol{D}_{22} \end{pmatrix},$$

It is assumed that $(\boldsymbol{A}, \boldsymbol{B}_2)$ is stabilisable and $(\boldsymbol{C}_2, \boldsymbol{A})$ is detectable.

Described in transfer function form

$$\begin{pmatrix} \boldsymbol{x}(s) \\ \boldsymbol{y}(s) \end{pmatrix} = \begin{pmatrix} \boldsymbol{G}_{11}(s) & \boldsymbol{G}_{12}(s) \\ \boldsymbol{G}_{21}(s) & \boldsymbol{G}_{22}(s) \end{pmatrix} \begin{pmatrix} \boldsymbol{w}(s) \\ \boldsymbol{u}(s) \end{pmatrix},$$

the transfer function matrix $\boldsymbol{G}(s)$ is decomposed as

$$\boldsymbol{G}(s) = \begin{pmatrix} \boldsymbol{G}_{11}(s) & \boldsymbol{G}_{12}(s) \\ \boldsymbol{G}_{21}(s) & \boldsymbol{G}_{22}(s) \end{pmatrix}.$$

Further, in order to later study the design of diagnosis in conjunction with closed-loop control, let the plant (9.95) be controlled by an output feedback controller

$$\boldsymbol{u}(t) = \boldsymbol{K}\boldsymbol{y}(t).$$

We have now the following definition of stabilisation by output feedback.

Definition 9.3 A proper system $\boldsymbol{G}(s)$ is said to be stabilisable by output feedback if there exists a proper controller $\boldsymbol{K}(s)$ internally stabilising $\boldsymbol{G}(s)$. Moreover, a proper controller $\boldsymbol{K}(s)$ is said to be admissible if it internally stabilises $\boldsymbol{G}(s)$.

Next, we have the following result for the existence of a stabilising controller $\boldsymbol{K}(s)$ for the system $\boldsymbol{G}(s)$, where here and in the following

$$\boldsymbol{K}(s) = \left[\begin{array}{c|c} \boldsymbol{M} & \boldsymbol{N} \\ \hline \boldsymbol{P} & \boldsymbol{Q} \end{array}\right]$$

is used as abbreviation of

$$\boldsymbol{K}(s) = \boldsymbol{P}(s\boldsymbol{I} - \boldsymbol{M})^{-1}\boldsymbol{N} + \boldsymbol{Q}.$$

Lemma 9.4 *There exists a proper controller $\boldsymbol{K}(s)$ achieving internal stability of the closed-loop system if and only if $(\boldsymbol{A},\ \boldsymbol{B}_2)$ is stabilisable and $(\boldsymbol{C}_2,\ \boldsymbol{A})$ is detectable. Further, let $\boldsymbol{F}$ and $\boldsymbol{L}$ be two matrices such that $\boldsymbol{A} + \boldsymbol{B}_2\boldsymbol{F}$ and $\boldsymbol{A} + \boldsymbol{L}\boldsymbol{C}_2$ are stable, then an observer-based stabilising controller is given by*

$$\boldsymbol{K}(s) = \left[\begin{array}{c|c} \boldsymbol{A} + \boldsymbol{B}_2\boldsymbol{F} + \boldsymbol{L}\boldsymbol{C}_2 + \boldsymbol{L}\boldsymbol{D}_{22}\boldsymbol{F} & \boldsymbol{F} \\ \hline -\boldsymbol{L} & \boldsymbol{O} \end{array}\right].$$

It is important to note that the stabilising controller for $\boldsymbol{G}(s)$ depends only on $\boldsymbol{G}_{22}(s)$. We need, therefore, only to look at $\boldsymbol{G}_{22}(s)$ when we are looking for stabilising controllers. This is also the case when we are using the Youla-Kucera parametrisation. In the following the argument s of transfer functions will often be omitted.

9.4.2 Youla-Kucera Parameterisation in Coprime Factorisation Form

First, let us consider two polynomials $m(s)$ and $n(s)$ with real coefficients. m and n are said to be coprime, if their greatest common divisor is 1 (they have no common zeros). It follows from Euclid's algorithm that f and g are coprime if and only if there exists polynomials $x(s)$ and $y(s)$ such that

$$mx + ny = 1. \tag{9.96}$$

This equation is called a Bezout identity. Similarly, the two stable transfer functions m and n are said to be coprime if there exists stable x and y such that Eq. (9.96) is satisfied.

Generally, two stable matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ are right coprime if they have equal number of columns and there exists stable matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ such that

$$(\boldsymbol{X} \quad \boldsymbol{Y}) \begin{pmatrix} \boldsymbol{M} \\ \boldsymbol{N} \end{pmatrix} = \boldsymbol{X}\boldsymbol{M} + \boldsymbol{Y}\boldsymbol{N} = \boldsymbol{I}.$$

This is equivalent to saying that the matrix $\begin{pmatrix} \boldsymbol{M} \\ \boldsymbol{N} \end{pmatrix}$ is stable left invertible.

Similarly, two stable matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ are left coprime if they have equal number of rows and there exists stable $\boldsymbol{X}$ and $\boldsymbol{Y}$ such that

$$(\boldsymbol{M} \quad \boldsymbol{N}) \begin{pmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \end{pmatrix} = \boldsymbol{M}\boldsymbol{X} + \boldsymbol{N}\boldsymbol{Y} = \boldsymbol{I}$$

holds. Equivalently, $(\boldsymbol{M} \quad \boldsymbol{N})$ is stable right invertible.

Now, let $\boldsymbol{G}_{22}(s)$ be a proper real-rational matrix. A right coprime factorisation of $\boldsymbol{G}_{22}(s)$ is a factorisation $\boldsymbol{G}_{22} = \boldsymbol{N}\boldsymbol{M}^{-1}$, where $\boldsymbol{N}$ and $\boldsymbol{M}$ are stable right coprime matrices. Similarly, a left coprime factorisation has the form $\boldsymbol{G}_{22} = \tilde{\boldsymbol{M}}^{-1}\tilde{\boldsymbol{N}}$, where $\tilde{\boldsymbol{N}}$ and $\tilde{\boldsymbol{M}}$ are left coprime. Note that, in these definitions, it is required that the matrices $\boldsymbol{M}$ and $\tilde{\boldsymbol{M}}$ are square and non-singular.

Based on the above, there exists the following result.

Lemma 9.5 *For each proper real-rational matrix $\boldsymbol{G}_{22}(s)$ there exists eight stable matrices satisfying the equations*

$$\begin{gathered} \boldsymbol{G}_{22} = \boldsymbol{N}\boldsymbol{M}^{-1} = \tilde{\boldsymbol{M}}^{-1}\tilde{\boldsymbol{N}} \\ \begin{pmatrix} \tilde{\boldsymbol{X}} & \tilde{\boldsymbol{Y}} \\ -\tilde{\boldsymbol{N}} & \tilde{\boldsymbol{M}} \end{pmatrix} \begin{pmatrix} \boldsymbol{M} & -\boldsymbol{Y} \\ \boldsymbol{N} & \boldsymbol{X} \end{pmatrix} = \boldsymbol{I}. \end{gathered}$$

This lemma defines a double coprime factorisation of $\boldsymbol{G}_{22}(s)$. It should be noted that it is always possible to make a coprime factorisation, if the system is stabilisable and detectable.

Now, let $\tilde{\boldsymbol{K}}(s)$ be a stabilising controller for $\boldsymbol{G}_{22}(s)$ and let $\tilde{\boldsymbol{K}}$ have the following factorisation

$$\tilde{\boldsymbol{K}} = \boldsymbol{U}\boldsymbol{V}^{-1} = \tilde{\boldsymbol{V}}^{-1}\tilde{\boldsymbol{U}}.$$

A feedback system with positive feedback is stable if and only if

$$\begin{pmatrix} \boldsymbol{I} & -\tilde{\boldsymbol{K}} \\ -\boldsymbol{G}_{22} & \boldsymbol{I} \end{pmatrix}^{-1} \text{ is stable.}$$

Using the coprime factorisation of $\tilde{\boldsymbol{K}}$ we get the following conditions for internal stability.

Lemma 9.6 *Let $\boldsymbol{G}_{22}(s)$ be a proper real-rational matrix and*

$$\boldsymbol{G}_{22} = \boldsymbol{N}\boldsymbol{M}^{-1} = \tilde{\boldsymbol{M}}^{-1}\,\tilde{\boldsymbol{N}}$$

be the stable right and left coprime factorisation. Then, there exists a controller

$$\tilde{\boldsymbol{K}}_0 = \boldsymbol{U}_0\boldsymbol{V}_0^{-1} = \tilde{\boldsymbol{V}}_0^{-1}\,\tilde{\boldsymbol{U}}_0$$

with $\boldsymbol{U}_0$, $\boldsymbol{V}_0$, $\tilde{\boldsymbol{U}}_0$ and $\tilde{\boldsymbol{V}}_0$ stable such that

$$\begin{pmatrix} \tilde{\boldsymbol{V}}_0 & -\tilde{\boldsymbol{U}}_0 \\ -\tilde{\boldsymbol{N}} & \tilde{\boldsymbol{M}} \end{pmatrix} \begin{pmatrix} \boldsymbol{M} & \boldsymbol{U}_0 \\ \boldsymbol{N} & \boldsymbol{V}_0 \end{pmatrix} = \boldsymbol{I}.$$

Based on the above results, it is now possible to give a parametrisation of all controllers that stabilise $\boldsymbol{G}_{22}(s)$.

Theorem 9.6 *Let $\boldsymbol{G}_{22}(s)$ be a proper real-rational matrix and*

$$\boldsymbol{G}_{22} = \boldsymbol{N}\boldsymbol{M}^{-1} = \tilde{\boldsymbol{M}}^{-1}\,\tilde{\boldsymbol{N}}$$

be the stable right and left coprime factorisation. Then, the set of all proper controllers achieving internal stability is parameterised either by

$$\begin{aligned} \boldsymbol{K} &= (\boldsymbol{U}_0 + \boldsymbol{M}\boldsymbol{Q}_{\mathrm{r}})(\boldsymbol{V}_0 + \boldsymbol{N}\boldsymbol{Q}_{\mathrm{r}})^{-1} \\ &\det(\boldsymbol{I} + \boldsymbol{V}_0^{-1}\boldsymbol{N}\boldsymbol{Q}_{\mathrm{r}})(\infty) \neq 0 \end{aligned} \tag{9.97}$$

for stable $\boldsymbol{Q}_{\mathrm{r}}$ or by

$$\begin{aligned} \boldsymbol{K} &= (\tilde{\boldsymbol{V}}_0 + \boldsymbol{Q}_l\tilde{\boldsymbol{N}})^{-1}(\tilde{\boldsymbol{U}}_0 + \boldsymbol{Q}_l\tilde{\boldsymbol{M}}) \\ &\det(\boldsymbol{I} + \boldsymbol{Q}_l\tilde{\boldsymbol{N}}\tilde{\boldsymbol{V}}_0^{-1})(\infty) \neq 0 \end{aligned} \tag{9.98}$$

for stable $\boldsymbol{Q}_l$, where $\boldsymbol{U}_0$, $\boldsymbol{V}_0$, $\tilde{\boldsymbol{U}}_0$ and $\tilde{\boldsymbol{V}}_0$ stable satisfied the Bezout identities:

$$\begin{aligned} \tilde{\boldsymbol{V}}_0\boldsymbol{M} - \tilde{\boldsymbol{U}}_0\boldsymbol{N} &= \boldsymbol{I} \\ \tilde{\boldsymbol{M}}\boldsymbol{V}_0 - \tilde{\boldsymbol{N}}\boldsymbol{U}_0 &= \boldsymbol{I}. \end{aligned}$$

Moreover, if $\boldsymbol{U}_0$, $\boldsymbol{V}_0$, $\tilde{\boldsymbol{U}}_0$ and $\tilde{\boldsymbol{V}}_0$ are chosen such that

$$\begin{pmatrix} \tilde{\boldsymbol{V}}_0 & -\tilde{\boldsymbol{U}}_0 \\ -\tilde{\boldsymbol{N}} & \tilde{\boldsymbol{M}} \end{pmatrix} \begin{pmatrix} \boldsymbol{M} & \boldsymbol{U}_0 \\ \boldsymbol{N} & \boldsymbol{V}_0 \end{pmatrix} = \boldsymbol{I}.$$

Then we have

$$\begin{aligned} \boldsymbol{K} &= \left(\boldsymbol{U}_0 + \boldsymbol{M}\boldsymbol{Q}_\mathrm{r}\right)\left(\boldsymbol{V}_0 + \boldsymbol{N}\boldsymbol{Q}_\mathrm{r}\right)^{-1} \\ &= \left(\tilde{\boldsymbol{V}}_0 + \boldsymbol{Q}_\mathrm{r}\tilde{\boldsymbol{N}}\right)^{-1}\left(\tilde{\boldsymbol{U}}_0 + \boldsymbol{Q}_\mathrm{r}\tilde{\boldsymbol{M}}\right) \\ &= \mathcal{F}_l(\boldsymbol{J}_\mathrm{r}, \boldsymbol{Q}_\mathrm{r}), \end{aligned} \tag{9.99}$$

where

$$\boldsymbol{J}_\mathrm{r} = \begin{pmatrix} \boldsymbol{U}_0\boldsymbol{V}_0^{-1} & \tilde{\boldsymbol{V}}_0^{-1} \\ \boldsymbol{V}_0^{-1} & -\boldsymbol{V}_0^{-1}\boldsymbol{N} \end{pmatrix}$$

and $\boldsymbol{Q}_\mathrm{r}$ *is stable and satisfies that* $(\boldsymbol{I} + \boldsymbol{V}_0^{-1}\boldsymbol{N}\boldsymbol{Q}_y)(\infty)$ *is invertible.*

The Youla-Kucera parametrisation is shown in Fig. 9.34.

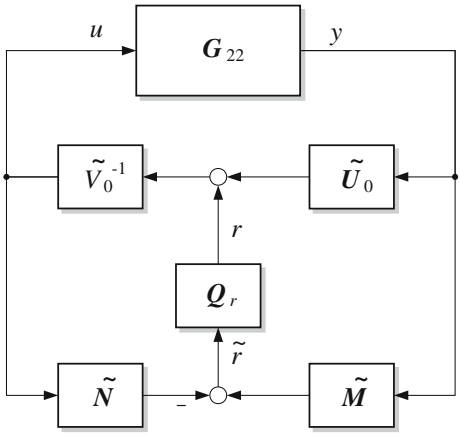


Fig. 9.34 Controller structure for the Youla-Kucera parametrisation

9.4.3 Parametrisation in the State-Space Form

The Youla-Kucera parametrisation derived in the above section was based on coprime factorisation, which may not be the form in which a particular fault-tolerant control problem is described. Further, popular toolboxes support a state-space description (e.g. MATLAB). Therefore, a state-space description will be given in this section together with a representation of the closed-loop transfer function as function of the free stable parameter $\boldsymbol{Q}$.

For the coprime factorisation in a state-space form using state feedback and observers, the following result is available: Let two coprime factorisations of $\boldsymbol{G}_{22}(s)$ be given by

$$\begin{pmatrix} \boldsymbol{M} & \boldsymbol{U}_0 \\ \boldsymbol{N} & \boldsymbol{V}_0 \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F} & \boldsymbol{F} & \boldsymbol{C}_2+\boldsymbol{D}_{22}\boldsymbol{F} \\ \boldsymbol{B}_2 & \boldsymbol{I} & \boldsymbol{D}_{22} \\ -\boldsymbol{L} & \boldsymbol{O} & \boldsymbol{I} \end{pmatrix}$$
$$\begin{pmatrix} \tilde{\boldsymbol{V}}_0 & -\tilde{\boldsymbol{U}}_0 \\ -\tilde{\boldsymbol{N}} & \tilde{\boldsymbol{M}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}+\boldsymbol{L}\boldsymbol{C}_2 & \boldsymbol{F} & \boldsymbol{C}_2 \\ -(\boldsymbol{B}_2+\boldsymbol{L}\boldsymbol{D}_{22}) & \boldsymbol{I} & -\boldsymbol{D}_{22} \\ \boldsymbol{L} & \boldsymbol{O} & \boldsymbol{I} \end{pmatrix},$$

where $\boldsymbol{F}$ and $\boldsymbol{L}$ are chosen such that $\boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F}$ and $\boldsymbol{A}+\boldsymbol{L}\boldsymbol{C}_2$ are both stable. It is now quite simple to give a state-space realisation of the $\boldsymbol{Q}$-parametrisation of all internal stabilising controllers. From Theorem 9.6 we have the linear fractional transformation formulation of all stabilising controllers. Using the state-space description of the coprime factorisation in $\boldsymbol{J}_y$ we get the following result.

Theorem 9.7 *Let $\boldsymbol{F}$ and $\boldsymbol{L}$ be such that $\boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F}$ and $\boldsymbol{A}+\boldsymbol{L}\boldsymbol{C}_2$ are stable. Then all controllers that internally stabilise $\boldsymbol{G}(s)$ can be parameterised as the transfer matrix from $\boldsymbol{y}$ to $\boldsymbol{u}$ given by $\mathcal{F}_l(\boldsymbol{J}_y, \boldsymbol{Q})$, where*

$$\boldsymbol{J}_y = \begin{pmatrix} \boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F}+\boldsymbol{L}\boldsymbol{C}_2+\boldsymbol{L}\boldsymbol{D}_{22}\boldsymbol{F} & \boldsymbol{F} & -(\boldsymbol{C}_2+\boldsymbol{D}_{22}\boldsymbol{F}) \\ -\boldsymbol{L} & \boldsymbol{O} & \boldsymbol{I} \\ \boldsymbol{B}_2+\boldsymbol{L}\boldsymbol{D}_{22} & \boldsymbol{I} & -\boldsymbol{D}_{22} \end{pmatrix}$$

with any $\boldsymbol{Q} \in \mathcal{RH}_\infty$ and $\boldsymbol{I}+\boldsymbol{D}_{22}\boldsymbol{Q}(\infty)$ is non-singular.

The controller given in the theorem is sometimes called the $\boldsymbol{Q}$-observer-based controller. For $\boldsymbol{Q}=\boldsymbol{O}$ the nominal controller turns out to be a standard full-order observer-based controller. Moreover, it can be shown that the separation between the design of the state feedback gain $\boldsymbol{F}$ and the observer gain $\boldsymbol{L}$ is still valid as well as a separation between the nominal controller and the $\boldsymbol{Q}$ parameter. This can be shown by setting up a state-space description of the controller together with the $\boldsymbol{Q}$ parameter and use the state vector $\bar{\boldsymbol{x}} = \begin{pmatrix} \boldsymbol{x} & \boldsymbol{x}-\hat{\boldsymbol{x}} & \boldsymbol{x}_q \end{pmatrix}$, where $\boldsymbol{x}_q$ is the state vector for $\boldsymbol{Q}$.

Next, let us look at the closed-loop transfer function when we have applied a $\boldsymbol{Q}$-parameterised controller as given in Theorem 9.7. The closed-loop transfer function is given by the following linear fractional transformation:

$$\boldsymbol{z} = \mathcal{F}_l(\boldsymbol{G}, \boldsymbol{K})\boldsymbol{w} = \mathcal{F}_l(\boldsymbol{G}, \mathcal{F}_l(J_y, \boldsymbol{Q}))\boldsymbol{w} = \mathcal{F}_l(\boldsymbol{T}, \boldsymbol{Q})\boldsymbol{w}.$$

We need now just to give a state-space description of $\boldsymbol{T}$. By using straightforward and tedious algebra, we get the following result.

Theorem 9.8 *Let $\boldsymbol{F}$ and $\boldsymbol{L}$ be such that $\boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F}$ and $\boldsymbol{A}+\boldsymbol{L}\boldsymbol{C}_2$ are stable. Then, the set of a closed-loop transfer matrices from $\boldsymbol{w}$ to $\boldsymbol{z}$ achievable by an stabilising proper controller is equal to*

$$\mathcal{F}_l(\boldsymbol{T}, \boldsymbol{Q}) = \boldsymbol{T}_{11}+\boldsymbol{T}_{12}\boldsymbol{Q}\boldsymbol{T}_{21}, \quad \boldsymbol{Q} \in \mathcal{RH}_\infty, \quad \boldsymbol{I}+\boldsymbol{D}_{22}\boldsymbol{Q}(\infty),$$

where $\boldsymbol{T}$ *is given by*

$$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{T}_{11} & \boldsymbol{T}_{12} \\ \boldsymbol{T}_{21} & \boldsymbol{T}_{22} \end{pmatrix} = \left(\begin{array}{cc|cc} \boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F} & -\boldsymbol{B}_2\boldsymbol{F} & \boldsymbol{B}_1 & \boldsymbol{B}_2 \\ \boldsymbol{O} & \boldsymbol{A}+\boldsymbol{L}\boldsymbol{C}_2 & \boldsymbol{B}_1+\boldsymbol{L}\boldsymbol{D}_{21} & \boldsymbol{O} \\ \hline \boldsymbol{C}_1+\boldsymbol{D}_{12}\boldsymbol{F} & -\boldsymbol{D}_{12}\boldsymbol{F} & \boldsymbol{D}_{11} & \boldsymbol{D}_{12} \\ \boldsymbol{O} & \boldsymbol{C}_2 & \boldsymbol{D}_{21} & \boldsymbol{O} \end{array}\right).$$

It is important to note that the closed-loop transfer matrix $\boldsymbol{T}$ is an affine function of the controller parameter matrix $\boldsymbol{Q}$, since $\boldsymbol{T}_{22} = \boldsymbol{O}$. This is the reason why the $\boldsymbol{Q}$-parametrisation is so useful, particularly in connection with optimisation of controllers by using numerical tools.

9.4.4 Simultaneous Design of the Controller and the Residual Generator

In the closed loop, there is an interaction between the sensitivity of the residual generated by a fault detection filter and the natural suppression of any fault within a closed loop. The design of closed-loop control and residual generator can, therefore, be considered an integrated design problem. Consider, the simultaneous design of the feedback controller and the residual generator. The design setup is illustrated in Fig. 9.35 (left). It uses the standard problem philosophy.

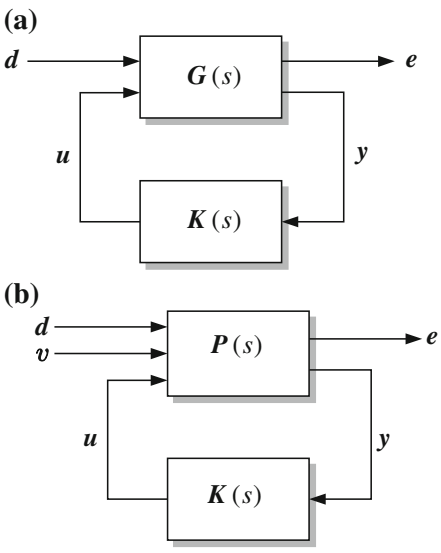


Fig. 9.35 Control system in standard configuration (*left*) and in generalised setup for fault-tolerant control (*right*)

As stated earlier, the standard design provides a controller $\boldsymbol{K}(s)$ for which the closed loop is internally stable and a suitable norm of the closed-loop transfer function from $\boldsymbol{w}$ to $\boldsymbol{z}$ is minimised or made smaller than a pre-specified level.

Instead of using a standard controller as shown in Fig. 9.35 (left), a controller with two outputs can be employed:

$$\begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{a} \end{pmatrix} = \begin{pmatrix} \boldsymbol{K}_1 \\ \boldsymbol{K}_2 \end{pmatrix} \boldsymbol{y}.$$

The additional output signal $\boldsymbol{a}$ is a diagnostic signal, which will be applied to derive an estimate of faults in the controlled system.

Let the open-loop transfer function be given by:

$$\begin{pmatrix} \boldsymbol{e} \\ \boldsymbol{y} \end{pmatrix} = \begin{pmatrix} \boldsymbol{G}_{\mathrm{ed}} & \boldsymbol{G}_{\mathrm{ef}} & \boldsymbol{G}_{\mathrm{eu}} \\ \boldsymbol{G}_{\mathrm{yd}} & \boldsymbol{G}_{\mathrm{yf}} & \boldsymbol{G}_{\mathrm{yu}} \end{pmatrix} \begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{f} \\ \boldsymbol{u} \end{pmatrix} \tag{9.100}$$

To obtain a good estimation of the individual faults, fault models are included in the generalised system as frequency weightings on the faults signals

$$\boldsymbol{f} = \boldsymbol{W}_{\mathrm{f}}(s)\boldsymbol{v},$$

where $\boldsymbol{v}$ is a signal that is anticipated to have a flat power spectrum. The generalised setup is shown in Fig. 9.35 (right).

Now we need to formulate the design setup in Fig. 9.35 (right) as a standard design problem as illustrated in Fig. 9.35 (left). For doing this, define an additional output $\boldsymbol{r}$ as the fault estimation error:

$$\boldsymbol{r} = \boldsymbol{f} - \boldsymbol{a}. \tag{9.101}$$

This is the standard way of formulating a filter design problem in the standard setup. The generalised system $\boldsymbol{P}(s)$ is then given by:

$$\begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{r} \\ \boldsymbol{y} \end{pmatrix} = \boldsymbol{P}(s) \begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{v} \\ \boldsymbol{u} \end{pmatrix} \tag{9.102}$$

with

$$\boldsymbol{P}(s) = \left(\begin{array}{cc|cc} \boldsymbol{G}_{\mathrm{ed}} & \boldsymbol{G}_{\mathrm{ef}}\boldsymbol{W}_{\mathrm{f}} & \boldsymbol{G}_{\mathrm{eu}} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{W}_{\mathrm{f}} & \boldsymbol{O} & -\boldsymbol{I} \\ \hline \boldsymbol{G}_{\mathrm{yd}} & \boldsymbol{G}_{\mathrm{yf}}\boldsymbol{W}_{\mathrm{f}} & \boldsymbol{G}_{\mathrm{yu}} & \boldsymbol{O} \end{array}\right).$$

Using the system setup in (9.102) and the controller

$$\boldsymbol{u} = \boldsymbol{K}(s)\boldsymbol{y}$$

we get the following closed-loop transfer function

$$\begin{pmatrix} \boldsymbol{e} \\ \boldsymbol{r} \end{pmatrix} = \boldsymbol{T}_{\mathrm{cl}}(s) \begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{v} \end{pmatrix}$$

with

$$\begin{aligned} \boldsymbol{T}_{\mathrm{cl}}(s) = &\begin{pmatrix} \boldsymbol{G}_{\mathrm{ed}} & \boldsymbol{G}_{\mathrm{ef}}\boldsymbol{W}_{\mathrm{f}} \\ \boldsymbol{O} & \boldsymbol{W}_{\mathrm{f}} \end{pmatrix} + \\ &\begin{pmatrix} \boldsymbol{G}_{\mathrm{eu}} & \boldsymbol{O} \\ \boldsymbol{O} & -\boldsymbol{I} \end{pmatrix} \boldsymbol{K}(s)(\boldsymbol{I} - \begin{pmatrix} \boldsymbol{G}_{\mathrm{yu}} & \boldsymbol{O} \end{pmatrix} \boldsymbol{K}(s))^{-1} \begin{pmatrix} \boldsymbol{G}_{\mathrm{yd}} & \boldsymbol{G}_{\mathrm{yf}}\boldsymbol{W}_{\mathrm{f}} \end{pmatrix}. \end{aligned}$$

For simplicity, assume that $\boldsymbol{G}(s)$ is open-loop stable (the unstable case can be dealt with as well in this methodology, but is computationally more difficult). Then, the Youla-Kucera parameterisation of all stabilising controllers can be obtained by making the substitutions

$$\begin{aligned} \boldsymbol{Q}(s) &= \boldsymbol{K}(s)(\boldsymbol{I} - \begin{pmatrix} \boldsymbol{G}_{\mathrm{yu}} & \boldsymbol{O} \end{pmatrix} \boldsymbol{K}(s))^{-1} \\ \boldsymbol{K}(s) &= \boldsymbol{Q}(s)(\boldsymbol{I} + \begin{pmatrix} \boldsymbol{G}_{\mathrm{yu}} & \boldsymbol{O} \end{pmatrix} \boldsymbol{Q}(s))^{-1}, \end{aligned} \tag{9.103}$$

where $\boldsymbol{Q}(s)$ is a stable proper transfer function, namely the Youla parameter. Further, let $\boldsymbol{Q}(s)$ be partitioned as:

$$\boldsymbol{Q}(s) = \begin{pmatrix} \boldsymbol{Q}_1(s) \\ \boldsymbol{Q}_2(s) \end{pmatrix}.$$

Then, the following equation for the closed-loop transfer function $\boldsymbol{T}_{\mathrm{cl}}$ is obtained:

$$\boldsymbol{T}_{\mathrm{cl}}(s) = \begin{pmatrix} \boldsymbol{G}_{\mathrm{ed}} + \boldsymbol{G}_{\mathrm{eu}}\boldsymbol{Q}_1\boldsymbol{G}_{\mathrm{yd}} & \boldsymbol{G}_{\mathrm{ef}}\boldsymbol{W}_{\mathrm{f}} + \boldsymbol{G}_{\mathrm{eu}}\boldsymbol{Q}_1\boldsymbol{G}_{\mathrm{yf}}\boldsymbol{W}_{\mathrm{f}} \\ -\boldsymbol{Q}_2\boldsymbol{G}_{\mathrm{yd}} & \boldsymbol{W}_{\mathrm{f}} - \boldsymbol{Q}_2\boldsymbol{G}_{\mathrm{yf}}\boldsymbol{W}_{\mathrm{f}} \end{pmatrix}. \tag{9.104}$$

Note that $\boldsymbol{Q}_1$ only appears in the first row of $\boldsymbol{T}_{\mathrm{cl}}$ and $\boldsymbol{Q}_2$ only in the second row. A separation between the design of $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ has, therefore, been obtained, which is a salient feature of this design approach.

Calculating $\boldsymbol{K}(s)$ directly from (9.103) results in the following equation:

$$\begin{aligned} \boldsymbol{K}(s) &= \begin{pmatrix} \boldsymbol{Q}_1(s)(\boldsymbol{I} + \boldsymbol{G}_{\mathrm{yu}}\boldsymbol{Q}_1(s))^{-1} \\ \boldsymbol{Q}_2(s)(\boldsymbol{I} + \boldsymbol{G}_{\mathrm{yu}}\boldsymbol{Q}_1(s))^{-1} \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{Q}_1(s)(\boldsymbol{I} + \boldsymbol{G}_{\mathrm{yu}}\boldsymbol{Q}_1(s))^{-1} \\ \boldsymbol{Q}_2(s)(\boldsymbol{I} - \boldsymbol{G}_{\mathrm{yu}}\boldsymbol{K}_1(s)) \end{pmatrix}. \end{aligned} \tag{9.105}$$

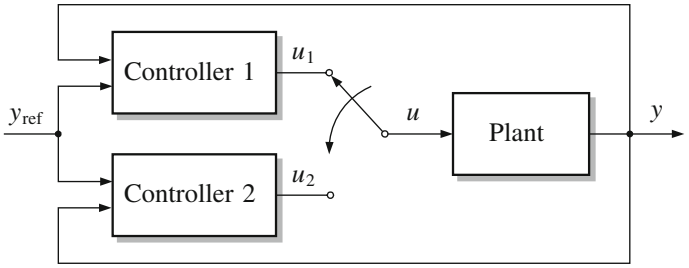


Fig. 9.36 Two-controller scheme

The result indicates that also the original controller structure is separated in a design of the feedback controller $\boldsymbol{K}_1(s)$ and a design of the fault detection filter $\boldsymbol{K}_2(s)$, which depends upon the controller $\boldsymbol{K}_1(s)$.

This result is essential for proper design of residual generators working in closed loop. It is also important for the redesign problem since the effect that the redesigned controller has on the diagnostic filters cannot be ignored.

9.5 Handling the Fault Recovery Transients

9.5.1 Switching Between Controllers

In the previous sections, controller reconfiguration or accommodation often amounts to switching from the nominal controller to a newly designed controller or from one to another element of a bank of controllers. When the considered control laws are of the state feedback or output feedback type, no precaution is required to switch between controllers with the same reference input. However, the situation is different for dynamical controllers. Indeed, the state of the controller which is not in the loop has to be initialised properly before this controller is introduced in the loop, in order to avoid bumps in the system response. One method to achieve this goal is presented here. It amounts to feeding back to each controller, be it active in the loop or not, the manipulated variable actually applied to the process (namely the process input). This mechanism is similar to an anti-windup strategy, which is normally used to handle actuator saturation in a control loop.

Without loss of generality, a situation with two controllers is considered here, so that one controller, say controller 1, is active in the loop, and controller 2 is the controller towards which switching occurs in the fault case (Fig. 9.36). Both controllers are supposed to be described by a linear state-space model of the form

$$\begin{aligned} \dot{\boldsymbol{x}}_{ci}(t) &= \boldsymbol{A}_{ci}\boldsymbol{x}_{ci}(t) + \boldsymbol{B}_{ci}\boldsymbol{y}_{\text{ref}}(t) + \boldsymbol{E}_{ci}\,\boldsymbol{y}(t) \quad \boldsymbol{x}_{ci}(0) = \boldsymbol{x}_{ci}^0 \\ \boldsymbol{u}_i(t) &= \boldsymbol{C}_{ci}\boldsymbol{x}_{ci}(t) + \boldsymbol{D}_{ci}\boldsymbol{y}_{\text{ref}}(t) + \boldsymbol{F}_{ci}\boldsymbol{y}(t) \quad i = 1, 2, \end{aligned} \tag{9.106}$$

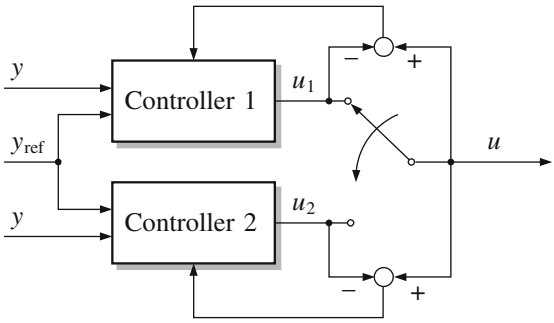


Fig. 9.37 Two-controller scheme with anti-windup mechanism

where $\boldsymbol{x}_{ci}(t), \boldsymbol{u}_i(t), \boldsymbol{y}(t)$ and $\boldsymbol{y}_{ref}(t)$ are respectively the controller state, the controller output, the measured plant output and the reference.

As explained in the previous paragraph, to obtain a smooth switching towards controller 2, the state of this controller must be properly initialised. This can be achieved thanks to an observer-based anti-windup mechanism. It amounts to feeding back the difference $\boldsymbol{u}(t) - \boldsymbol{u}_2(t)$ between the plant input and the output of controller 1 towards controller 2:

$$\begin{aligned} \dot{\boldsymbol{x}}_{c2}(t) &= \boldsymbol{A}_{c2}\boldsymbol{x}_{c2}(t) + \boldsymbol{B}_{c2}\boldsymbol{y}_{ref}(t) + \boldsymbol{E}_{c2}\,\boldsymbol{y}(t) + \boldsymbol{L}_2(\boldsymbol{u}(t) - \boldsymbol{u}_2(t)) \\ \boldsymbol{u}_2(t) &= \boldsymbol{C}_{c2}\boldsymbol{x}_{c2}(t) + \boldsymbol{D}_{c2}\boldsymbol{y}_{ref}(t) + \boldsymbol{F}_{c2}\boldsymbol{y}(t). \end{aligned} \tag{9.107}$$

Substituting the output equation for $\boldsymbol{u}_2(t)$ in the state equation of (9.107) yields

$$\begin{aligned} \dot{\boldsymbol{x}}_{c2}(t) &= (\boldsymbol{A}_{c2} - \boldsymbol{L}_2\boldsymbol{C}_{c2})\boldsymbol{x}_{c2}(t) + (\boldsymbol{B}_{c2} - \boldsymbol{L}_2\boldsymbol{D}_{c2})\boldsymbol{y}_{ref}(t) \\ &\quad + (\boldsymbol{E}_{c2} - \boldsymbol{L}_2\boldsymbol{F}_{c2})\,\boldsymbol{y}(t) + \boldsymbol{L}_2\boldsymbol{u}(t) \\ \boldsymbol{u}_2(t) &= \boldsymbol{C}_{c2}\boldsymbol{x}_{c2}(t) + \boldsymbol{D}_{c2}\boldsymbol{y}_{ref}(t) + \boldsymbol{F}_{c2}\boldsymbol{y}(t) \end{aligned} \tag{9.108}$$

and shows that $\boldsymbol{L}_2$ should be chosen in such a way that $(\boldsymbol{A}_{c2} - \boldsymbol{L}_2\boldsymbol{C}_{c2})$ has all its eigenvalues inside the open left-half plane in order for $\boldsymbol{x}_{c2}(t)$ to reach a steady-state value when controller 2 is not inserted in the loop, in the absence of change in $\boldsymbol{y}_{ref}(t)$, $\boldsymbol{y}(t)$ and $\boldsymbol{u}(t)$. Possible options consist in choosing $\boldsymbol{L}_2$ so that all eigenvalues lie at the origin, or to use $\boldsymbol{L}_2 = \boldsymbol{B}_{c2}\boldsymbol{D}_{c2}^{-1}$, which corresponds to the so-called conditioning technique. The latter approach requires a square full-rank matrix $\boldsymbol{D}_{c2}$, although this conditions can be weakened. It also requires that the zeros of the controller lie in the open left-half plane.

Obviously, for reason of symmetry, to allow switching from controller 2 to controller 1, the latter controller must be provided with a similar anti-windup feature. Its state-space equation is thus written like (9.108), with index 1 substituted for 2. The resulting block diagram is given in Fig. 9.37.

9.5.2 Progressive Fault Accommodation

In the ideal fault-tolerant linear quadratic problem described in Sect. 7.2 the fault detection, isolation and estimation process take no time. However, it has been already noted that in practice, three time periods exist:

Time window	System situation	System	Controller
$[0, t_f[$	Nominal operation	$(\boldsymbol{A}, \boldsymbol{B})$	$\boldsymbol{u} = -\boldsymbol{R}^{-1}\boldsymbol{B}'P\boldsymbol{x}$
$[t_f, t_a[$	Fault detection, isolation and estimation process, fault accommodation process delay	$(\boldsymbol{A}, \boldsymbol{B}_f)$	$\boldsymbol{u} = -\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}\boldsymbol{x}$
$[t_a, \infty)$	Fault is accommodated	$(\boldsymbol{A}, \boldsymbol{B}_f)$	$\boldsymbol{u}_f = -\boldsymbol{R}^{-1}\boldsymbol{B}_f^T\boldsymbol{P}_f\boldsymbol{x}$

During the time period $[t_f, t_a[$ the faulty system $(\boldsymbol{A}, \boldsymbol{B}_f)$ is still controlled by the nominal control $\boldsymbol{u} = -\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}\boldsymbol{x}$. This control is optimal for $(\boldsymbol{A}, \boldsymbol{B})$ and the closed loop $\boldsymbol{A} - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}$ is stable, but no guarantee can be given when $\boldsymbol{B}$ is replaced by $\boldsymbol{B}_f$ and the closed loop $\boldsymbol{A} - \boldsymbol{B}_f\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}$ may be unstable. If $t_a - t_f$ is not small enough, although the new control law $\boldsymbol{u}_f$ will recover the system stability and provide the best possible performance when applied, the system state may violate some physical limits or it may lead to a non-admissible value of the system cost. Note that physical limits are not formalised in the standard LQ problem setting, but they are usually taken into account by an appropriate choice of the weighting matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$.

Therefore, to solve practical problems the fault detection, isolation and estimation process delay as well as the fault accommodation process delay have to be made as small as possible. As far as fault accommodation is concerned, this can be obtained by two complementary strategies:

- Design an algorithm that computes the accommodated control $\boldsymbol{u}_f$ (i.e. that solves the algebraic Riccati equation) in minimum time,
- Design an algorithm that computes a sequence of controls that will eventually converge to $\boldsymbol{u}_f$ and will stabilise the system as soon as possible. Such an algorithm belongs to the family of "anytime" algorithms, which means that the result of any iteration is acceptable, and it will improve as the number of iterations increases. This is the *progressive accommodation* strategy.

Newton-Raphson iteration scheme for solving the algebraic Riccati equation. The Newton-Raphson iteration scheme has been proposed in the literature as an effective way of solving the algebraic Riccati equation. Let $\boldsymbol{P}_i$ be the unique solution of the Lyapunov equation

$$\boldsymbol{P}_i(\boldsymbol{A} - \boldsymbol{B}_f\boldsymbol{F}_{i-1}) + (\boldsymbol{A} - \boldsymbol{B}_f\boldsymbol{F}_{i-1})^T\boldsymbol{P}_i = -\boldsymbol{Q} - \boldsymbol{F}_{i-1}^T\boldsymbol{R}\boldsymbol{F}_{i-1}, \tag{9.109}$$

where

$$\boldsymbol{F}_i = \boldsymbol{R}^{-1}\boldsymbol{B}_{\mathrm{f}}\boldsymbol{P}_i \tag{9.110}$$

for all $i = 1, 2, \ldots$ and the initial $\boldsymbol{F}_0$ is given. If $\boldsymbol{A} - \boldsymbol{B}_{\mathrm{f}}\boldsymbol{F}_0$ is stable, then all matrices $\boldsymbol{P}_i$ are positive definite, and one has the convergence result

$$\begin{array}{l}(1)\ \boldsymbol{P}_0 \geq \boldsymbol{P}_1 \geq \cdots \geq \boldsymbol{P}_i \geq \boldsymbol{P}_{i+1} \geq \cdots \geq \boldsymbol{P}_{\mathrm{f}}, \quad i = 1, 2, \ldots \\ (2)\ \lim_{i\to\infty} \boldsymbol{P}_i = \boldsymbol{P}_{\mathrm{f}},\end{array} \tag{9.111}$$

where $\boldsymbol{P}_{\mathrm{f}}$ is the solution of the algebraic Riccati equation

$$\boldsymbol{P}_{\mathrm{f}}(\boldsymbol{A} - \boldsymbol{B}_{\mathrm{f}}\boldsymbol{F}_{\mathrm{f}}) + (\boldsymbol{A} - \boldsymbol{B}_{\mathrm{f}}\boldsymbol{F}_{\mathrm{f}})^{\mathrm{T}}\boldsymbol{P}_{\mathrm{f}} = -\boldsymbol{Q} - \boldsymbol{F}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{F}_{\mathrm{f}}.$$

Progressive Accommodation (PA) scheme. The PA scheme is based on the Newton-Raphson algorithm. Assume that iteration i takes a time Δ_i, and consider the sequence

$$t_i = t_{\mathrm{init}} + \sum_{j=1}^{i} \Delta_j, \quad i = 1, 2, \ldots,$$

where t_{init} is the time at which the Newton-Raphson algorithm is initialised after the fault has been detected, isolated and estimated (note that $t_{\mathrm{f}} < t_{fdi} < t_{\mathrm{init}}$). t_i is the time at which the result $\boldsymbol{F}_i$ becomes available (note that the constancy of Δ_i is not necessary, the scheme can, therefore, be employed whatever the tasks scheduling strategy of the FTC computer). The idea of progressive accommodation is to apply the feedback control law $\boldsymbol{u}_i = -\boldsymbol{F}_i\boldsymbol{x}$ on the time interval $\left[t_i, t_{i+1}\right[$. As a result, the system behaviour after the fault occurrence is

$$\dot{\boldsymbol{x}}(t) = (\boldsymbol{A} - \boldsymbol{B}_{\mathrm{f}}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P})\boldsymbol{x}(t), \quad t \in [t_{\mathrm{f}}, t_{\mathrm{init}}[\tag{9.112}$$

$$\dot{\boldsymbol{x}} = (\boldsymbol{A} - \boldsymbol{B}_{\mathrm{f}}\boldsymbol{F}_0)\boldsymbol{x}, \quad t \in [t_{\mathrm{init}}, t_1[\tag{9.113}$$

$$\dot{\boldsymbol{x}} = (\boldsymbol{A} - \boldsymbol{B}_{\mathrm{f}}\boldsymbol{F}_i)\boldsymbol{x}, \quad t \in [t_i, t_{i+1}[\quad i = 1, 2, \ldots, \tag{9.114}$$

where $\boldsymbol{F}_0$ is the Newton-Raphson initialisation at time t_{init}. It can be shown from (9.111) that if the system $(\boldsymbol{A}, \boldsymbol{B}_{\mathrm{f}})$ is stabilised by $\boldsymbol{F}_0$, then it is stabilised by all $\boldsymbol{F}_i$, and each $\boldsymbol{F}_i$ is better than the previous one with respect to the LQ cost. Moreover, PA results in a lower cost than the one associated with controlling the system by the nominal control until the Newton-Raphson algorithm has converged (which means the accommodated solution is computed) and then applying the accommodated control. Figure 9.38 shows the fault-tolerant system architecture using the PA scheme.

$\boldsymbol{A} - \boldsymbol{B}_{\mathrm{f}}\boldsymbol{F}_0$ being stable is only a sufficient condition for the PA procedure to converge. Convergence may be obtained in some cases, even when the initial feedback does not stabilise the system. This happens in the following example.

Example 9.6 Progressive accommodation of a first-order system
Consider the following LQ problem

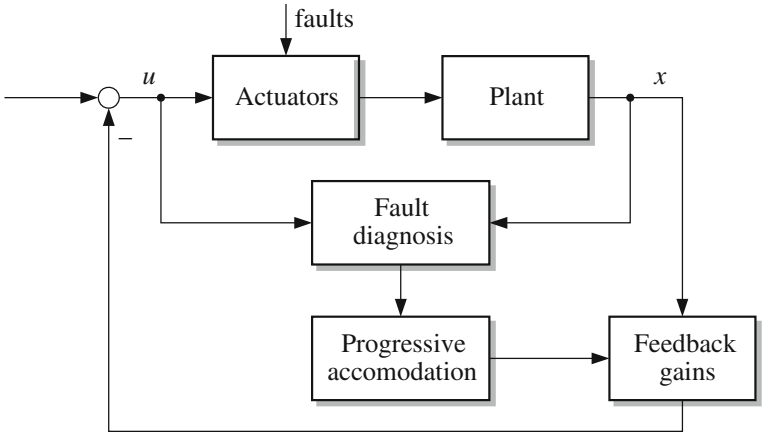


Fig. 9.38 Progressive accommodation scheme

$$\dot{\boldsymbol{x}}(t) = -\boldsymbol{x}(t) + \boldsymbol{u}(t), \quad \boldsymbol{x}(0) = 4$$
$$\boldsymbol{J} = \int_0^\infty [\boldsymbol{x}^2(t) + \boldsymbol{u}^2(t)]\mathrm{d}t.$$

The nominal Algebraic Riccati Equation is $\boldsymbol{P}^2 + 2\boldsymbol{P} - 1 = \boldsymbol{O}$ leading to the optimal control $\boldsymbol{u}(t) = (1 - \sqrt{2})\boldsymbol{x}(t)$, and closed-loop behaviour $\dot{\boldsymbol{x}} = -\sqrt{2}\boldsymbol{x}$. Let the faulty system be

$$\dot{\boldsymbol{x}}(t) = -\boldsymbol{x} - 2\sqrt{2}\boldsymbol{u}(t) \qquad t \geq 1$$

Under the nominal control, the faulty system behaviour is $\dot{\boldsymbol{x}}(t) = (3 - 2\sqrt{2})\boldsymbol{x}(t)$ which is unstable. The new Algebraic Riccati Equation is $8\boldsymbol{P}_\mathrm{f}^2 + 2\boldsymbol{P}_\mathrm{f} - 1 = \boldsymbol{O}$ whose stable solution gives the optimal control of the faulty system $\boldsymbol{u}_\mathrm{f} = \frac{\sqrt{2}}{2}\boldsymbol{x}$ and the closed-loop behaviour $\dot{\boldsymbol{x}}(t) = -3\boldsymbol{x}(t)$. The Newton-Raphson algorithm results in

$$\boldsymbol{P}_i = \frac{1 + 8\boldsymbol{P}_{i-1}^2}{2(1 + 8\boldsymbol{P}_{i-1})},$$

which converges to the solution of the new Algebraic Riccati Equation. The table below shows the evolution of $\boldsymbol{P}_i$, while Fig. 9.39 shows the evolution of the system state when fault accommodation is applied after convergence of the Newton-Raphson scheme (which takes 3 iterations, with $\Delta = 1s$) (classical approach, dashed line), and using the progressive accommodation scheme (continuous line).

i	0	1	2	3	4	5
$k_i \times 10^2$	41.42	27.5	25.08	25	25	25

□

Example 9.7 Progressive accommodation in nominal trajectory tracking
Let us consider again the second-order system of the nominal trajectory tracking example, under the fault

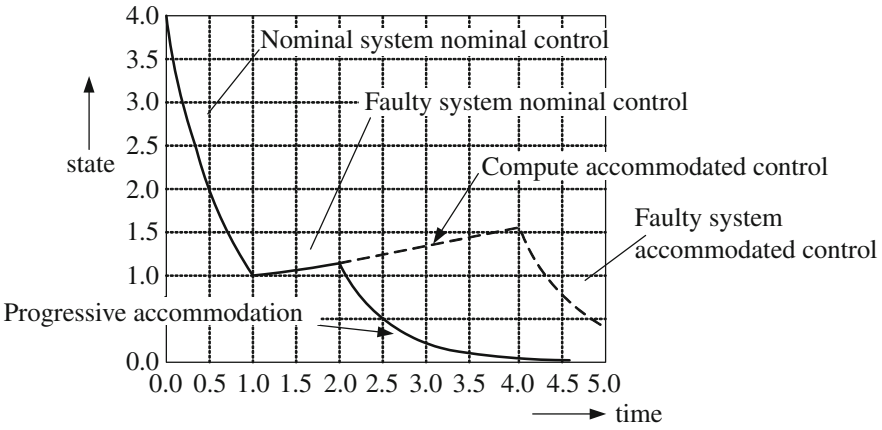


Fig. 9.39 Comparison of the classical and the progressive accommodation schemes

$$\boldsymbol{A}_{\mathrm{f}} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}, \quad \boldsymbol{B}_{\mathrm{f}} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

associated with the fault parameter $\theta = 1$. Remember that in this case, the pseudo-inverse method leads to an unstable system, and the modified pseudo-inverse method (MPIM) has to be applied. The alternative approach based on optimal nominal trajectory tracking gives the control

$$\boldsymbol{u}_{\mathrm{f}}(t) = \left(\boldsymbol{K}_n + \boldsymbol{R}^{-1}\boldsymbol{B}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{G}\right)\boldsymbol{x}_{\mathrm{n}}(t) + \boldsymbol{R}^{-1}\boldsymbol{B}_{\mathrm{f}}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x}_{\mathrm{f}}(t)$$

where

$$\boldsymbol{H} = \begin{pmatrix} -0.433 & -0.067 \\ -0.067 & -0.433 \end{pmatrix} \quad \text{and} \quad \boldsymbol{G} = \begin{pmatrix} 2.134 & 0.5 \\ 1.800 & -0.5 \end{pmatrix}$$

satisfy the Riccati equation

$$\boldsymbol{H}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}\boldsymbol{H} - 2\boldsymbol{H} - \boldsymbol{I}_2 = \boldsymbol{O} \tag{9.115}$$

and the Lyapunov equation

$$\boldsymbol{H}\begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix} + \boldsymbol{G}\begin{pmatrix} -2 & 0 \\ -5 & -1 \end{pmatrix} + \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}\boldsymbol{G} + \boldsymbol{I}_2 = \boldsymbol{O}. \tag{9.116}$$

Figure 9.31 that shows the state $\boldsymbol{x}_{\mathrm{n}}$, MPIM $\boldsymbol{x}_{\mathrm{f}}^{\mathrm{MPIM}}$ and nominal trajectory tracking $\boldsymbol{x}_{\mathrm{f}}$ trajectories, for three different fault detection, isolation, diagnosis and accommodation delays is recalled here as Fig. 9.40.

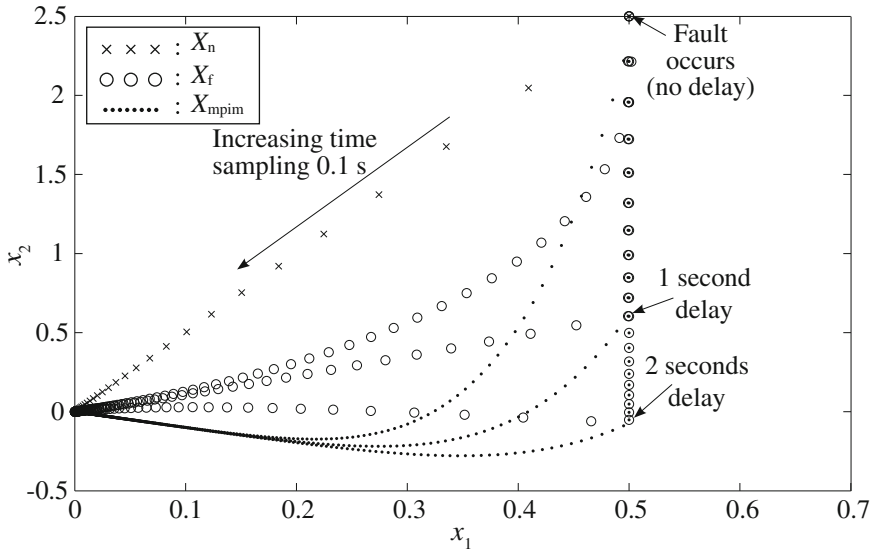


Fig. 9.40 Nominal, MPIM and NTT state trajectories

Assume a 2 s accommodation delay, which is composed of 1 s for the detection, isolation and diagnosis procedure, which ends with an estimate of matrix $\boldsymbol{B}_f$, and 1 s for control accommodation that solves Eqs. (9.115) and (9.116) based on this estimate. Let us now illustrate how much the efficiency of the fault accommodation scheme is improved by using the Progressive Accommodation approach. Solving the Riccati equation takes five Newton-Ralphson iterations, according to the following sequence:

$$\boldsymbol{H}_1 = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \boldsymbol{H}_2 = \begin{pmatrix} -0.7 & 0.2 \\ 0.2 & -0.7 \end{pmatrix}$$
$$\boldsymbol{H}_3 = \begin{pmatrix} -0.4839 & -0.0161 \\ -0.0161 & -0.4839 \end{pmatrix}, \quad \boldsymbol{H}_4 = \begin{pmatrix} -0.4357 & -0.0643 \\ -0.0643 & -0.4357 \end{pmatrix}$$
$$\boldsymbol{H}_5 = \begin{pmatrix} -0.4330 & -0.0670 \\ -0.0670 & -0.4330 \end{pmatrix}.$$

For this system, using the first iteration value $\boldsymbol{H}_1$ (which is obtained after 0, 2 s) instead of waiting the Riccati equation solution $\boldsymbol{H}_5$ for 1 s, allows to stabilise the system much sooner, and hence gives improved results.

Figure 9.41 compares the direct accommodation control and the progressive accommodation one. It is seen that progressive accommodation practically *rubs out* the effect of the accommodation delay: the resulting trajectories are quite close to the ones associated with a 1 s delay. □

9.6 Exercises

Exercise 9.1 Reconfiguration by model-matching techniques
Consider a stable plant

$$\begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{pmatrix} = \begin{pmatrix} -\frac{1}{T_1} & 0 \\ \frac{1}{T_2} & -\frac{1}{T_2} \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix}$$
$$y(t) = (1 \quad 1) \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}$$

and the stabilising proportional controller

$$\begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix} = \begin{pmatrix} -k_1 \\ 0 \end{pmatrix} y(t).$$

If the actuator 1 fails, the control loop should be closed with the help of the redundant control input $u_2(t)$. Does the model-matching approach yield a stable closed-loop system? Is the performance of the closed-loop system improved with respect to the nominal loop if the Markov parameter approach is used? □

Exercise 9.2 Fault-tolerant control of the three-tank system
Consider the three-tank system introduced in Sect. 2.2, where in the first part of the exercise the redundant hardware is switched off. Use a continuous PI-controller for the level $h_1(t)$ of the left tank and assume that the sensor used in this control loop fails. What is the result of the model-matching approach to this problem if the level $h_2(t)$ of the second tank is continuously measured and used for the controller of the left tank?

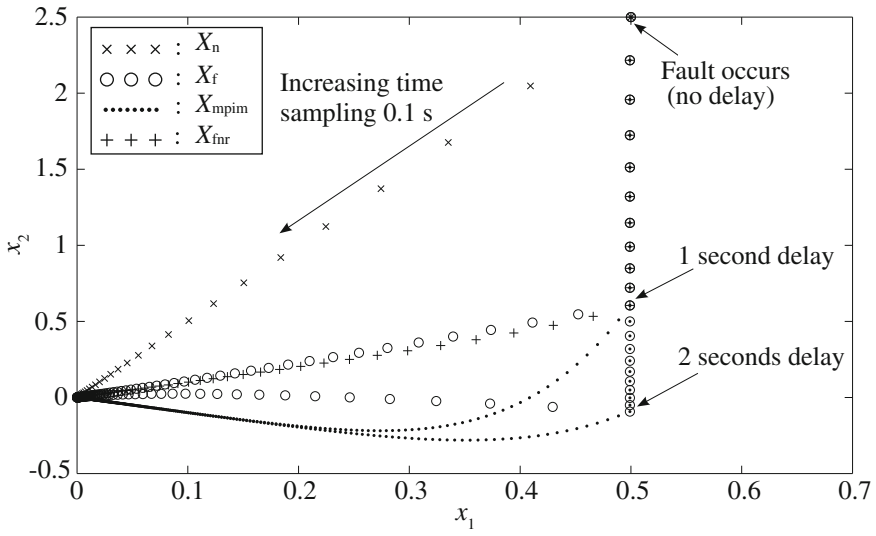


Fig. 9.41 Progressive accommodation in the nominal trajectory tracking state trajectories

Consider now a continuous level controller for Tank T_2 and assume that the actuator V_{12H} fails. The level $h_2(t)$ should be controlled by switching on the Tank T_3 and using the set-point $h_{3\text{ref}}(t)$ of the level controller of this tank as the control input to bring the level $h_2(t)$ of Tank T_2 towards the setpoint $h_{2\text{ref}}(t)$. Apply the existence conditions for model-matching and the virtual actuator to this problem. How can the failure of the control loop in Tank T_2 be compensated? □

Exercise 9.3 Virtual actuator
For the unstable system

$$\begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} 2 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix}$$
$$y(t) = \begin{pmatrix} 2 & 1 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}$$

a proportional controller

$$\begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix} = - \begin{pmatrix} 0 \\ k_2 \end{pmatrix} y(t).$$

should be found that stabilises the system. In case of the actuator failure a virtual actuator should be used to stabilise the system with the nominal controller. Find reasonable parameters of the virtual actuator and prove that the reconfigured closed-loop system is stable. □

Exercise 9.4 Nominal and model-matching control for a single-axis satellite
This exercise is a continuation of Exercises 5.3, 6.3 and 6.4. The objective is to perform attitude control for a single axis of a satellite. In this exercise, actuator dynamics need be considered and two states x_3 and x_4 have been added to describe actuator dynamics.

A state-space model for the nominal plant is given by:

$$\begin{aligned} \dot{x}_1(t) &= I^{-1}(x_3(t) + x_4(t) + d(t)) \\ \dot{x}_2(t) &= x_1(t) \\ \tau_1 \dot{x}_3(t) &= -x_3(t) + b_1 u_1(t) \\ \tau_2 \dot{x}_4(t) &= -x_4(t) + b_3 u_2(t) \\ y_1(t) &= x_1(t) + w_1(t) \\ y_2(t) &= x_2(t) + w_2(t) \\ y_3(t) &= x_3(t) + w_3(t). \end{aligned}$$

The system has actuator 1 (the input to which is $u_1(t)$) as the primary actuator. A second actuator (with input $u_2(t)$) is intended for secondary actuation should the primary one fail. The secondary actuator has a time constant that is larger than that of actuator 1.

Parameters. The nominal values of parameters are

$$\begin{aligned} \tau_1 &= 0.5 \text{ s} \\ \tau_2 &= 2.5 \text{ s} \\ b_1 &= 1.0 \\ b_2 &= 1.0 \end{aligned}$$

1. Design a nominal controller that use $sy_1(t)$ and $y_2(t)$ as measurements and $u_1(t)$ as actuator input,

$$u_1(t) = -(l_1 y_1(t) + l_2 y_2(t)). \tag{9.117}$$

The closed-loop system should have two real eigenvalues at $s = -0.5 \frac{\text{rad}}{\text{s}}$. Disregard the actuator that is not in use.

A fault on actuator 1 renders this actuator useless and it is needed to use actuator 2 instead.

2. Investigate whether ideal model matching is possible with this form of controller when this fault happens.
3. Design a dynamical controller that will provide model matching in the frequency domain. □

9.7 Bibliographical Notes

Fault-tolerant model-matching design. [151] is one of the earliest papers on controller reconfiguration by model-matching. [117] describes an improvement of the pseudo-inverse method for the ensurance of stability. A survey of the methods are given in [216]. A proof of Lemma 9.4 can be found in [412].

Further extensions of the pseudo-inverse method that result in the *admissible model-matching approach* have been recently given in [322], with extensions to the linear quadratic problem in [40, 328] and aerospace applications in [39].

Control reconfiguration for actuator or sensor failures. The ideas of the virtual sensor and virtual actuator have been developed in [220]. A thorough treatment can be found in the monographs [288, 340]. These concepts have been experimentally tested at a laboratory process [289], a two-degrees-of-freedom helicopter model [221] and a fuel cell [282]. The generalised version of the virtual actuator, which is explained in this chapter, has been proposed in [213]. Design methods for virtual sensors and virtual actuators can be found in [216, 290, 294, 310, 311] with extensions to nonlinear systems in [291, 293]. The conceptual similarities and differences of the virtual actuator and the dual observer are described in [292]. Alternative method that likewise use the fault-hiding principle are described, for example, in [202].

Fault-tolerant H_∞ design. Controller redesign based on the Youla-Kucera parametrisation is described in [188, 251, 359]. In [248], the Youla-Kucera parametrisation has been applied in connection with tuning controllers. The exact, the almost exact and the optimal design problems for $\boldsymbol{Q}$ have been considered in detail in [296].

The results in Sect. 9.4 are based on [344, 345] which focus on the use of fault estimation within a reliable control framework [375]. The methods for reconfiguration design are new within the fault-tolerant control domain. A few schemes have come into real application. Predetermined design for accommodation was demonstrated for a satellite in [42, 43].

The design methods considered in Sect. 9.5 are based on the same conditions as the methods described in [317]. Further results on using the Youla-Kucera parameterisation for fault-tolerant control in the additive fault, multiplicative fault and parameter fault cases can be found in [243, 247, 250, 251, 346]. An architecture for fault-tolerant control, based on joint controller and FDI design was presented in [241].

Theorem 9.6 has been proved in [412].

Handling of the fault recovery transients. The link between mastering the transient of controller switching and handling actuator saturation has been recognised for a long time. Indeed, both problems involve the discrepancy between the controller output and the process input, which might lead to performance degradation and even instability of the closed-loop. Anti-windup methods have thus been developed with a view to handle both problems (see [9] for the observer-based approach, and [140, 141, 276] for the conditioning technique). In [134, 406], they are explicitly introduced in multi-controller schemes such as found in hybrid or switched-mode systems in order to avoid undesirable switching transients. The anti-windup mechanisms used in [134] are high-gain feedback loops around each idle controller, which force the controller outputs to track the process input, while in [406] each controller is augmented with dynamics identical to that of the plant in order to allow the controller state to evolve in an appropriate way when the controller is not connected to the plant input. The latter scheme is cumbersome when the number of controllers is large.

Dedicated methods have also been studied for handling transients in controller switching. In [133], the authors recast the problem in an associated tracking problem, where the standby controller is viewed as a dynamical system of which the output should track the manipulated signal (plant input) by means of a two-degree-of-freedom controller. In [397], a simple new realisation of a set of linear SISO controllers is described that inherently assures bumpless transient upon switching between controllers.

Reference [396] made a rigorous analysis of controller switching and suggested an adaptive method to obtain bounded signals with a nominal controller from the instant a fault is detected until controller reconfiguration is made.

Progressive accommodation was first introduced in [402] to handle aircraft actuator faults, and the general approach was presented in [403]. In [66] "anytime algorithms" are used as an interesting tool to address fault recovery transients, since they produce solutions that are improving as the number of iterations increases, while any current solution can be applied before complete convergence is achieved.

In [334] the fault-tolerant linear quadratic problem is extended to the trajectory tracking problem which arises when a pre-designed system trajectory is to be followed as closely as possible (for example in space rendez-vous missions) instead of recomputing an optimal trajectory associated with the current configuration.

Chapter 10
Distributed Fault Diagnosis and Fault-Tolerant Control

Abstract Distributed systems are formed by the interconnection of several subsystems or autonomous agents. Each entity is equipped with a local computing device that runs the whole or a part of the diagnosis and fault-tolerant control algorithms. This chapter explains the specific features of such systems and provides tools for the design and the coordination of distributed algorithms that achieve the overall diagnosis and control specifications, under given communication structures and local computing power limitations.

10.1 Introduction

The need for distributed control directly follows from the growing dimensions of complex, large-scale, multi-agent systems. Star architectures that connect all field devices (sensors and actuators) to one single computer running all the control laws are unpractical for large-scale applications. Using several computers and hubs to implement the control laws and connect the field devices is possible, thanks to local area networks that transfer the needed measurements to the computing devices and the generated control signals to the system actuators. The development of multi-agent systems (teams of robots, fleets of unmanned vehicles) also heavily rests on data transmissions between the individual entities and on local decision making.

Assuming a distributed control architecture, the implementation of a global diagnoser may be an unpractical option because of the amount of needed communication that sometimes makes it technically impossible. The diagnosis algorithms must then also be distributed, by assigning a part of the global fault detection and isolation task to each subsystem.

Fault-tolerant distributed systems have been considered for long in the software community to cope with hardware, software and communication faults. More recently, specific problems have been considered in the control community for fault-tolerant estimation, diagnosis and control of large-scale systems. As far as control is concerned, distributed systems introduce an *information pattern*, meaning that different data sets are available to different controllers, as opposed to the conventional design where all controllers share the same information. The information pattern

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8_10

plays also a very important role in distributed diagnosis, since the amount of known data available to each subsystem is a key parameter for its detection and isolation capabilities.

In this chapter, distributed systems are first presented. Distributed diagnosis is then addressed in reference with the structural fault detection and isolation capability of the overall system. According to the locally available model and data, the local diagnosers provide more or less powerful conclusions that must be coordinated (or aggregated) into a system-level overall diagnosis. Distribution algorithms are then considered, based on information patterns that take into account the specificities of the communication architecture. The constraints associated with possible local computing power limitations are also considered. The second part of the chapter addresses fault-tolerant distributed control. Since the solvability of the control design problem depends on the information pattern that is implemented, it follows that a fault that is not recoverable under a given information pattern might be recoverable under another one. The reconfiguration of the information patterns appears therefore as a powerful tool to achieve fault tolerance.

10.2 Distributed Systems

10.2.1 System Decomposition

Consider a system Σ equipped with a set I of m actuators, and a set J of p sensors. Its behaviour is described by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), t) \tag{10.1}$$

$$\boldsymbol{y}(t) = \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), t), \tag{10.2}$$

where $\boldsymbol{x} \in |\mathcal{R}^n$ is the state, $\boldsymbol{u} \in |\mathcal{R}^m$ is the control vector, $\boldsymbol{y} \in |\mathcal{R}^p$ is the measurement vector and $\boldsymbol{d} \in |\mathcal{R}^q$ is some disturbance vector.

Let $\{\boldsymbol{u}_k,\ k = 1, \ldots, s\}$, $\{\boldsymbol{y}_k,\ k = 1, \ldots, s\}$ and $\{\boldsymbol{x}_k,\ k = 1, \ldots, s\}$ be partitions of $\boldsymbol{u}$, $\boldsymbol{y}$ and $\boldsymbol{x}$ into $s \geq 1$ subvectors, and let

$$\dot{\boldsymbol{x}}_k(t) = \boldsymbol{f}_k(\boldsymbol{x}_k(t), \overline{\boldsymbol{x}}_k(t), \boldsymbol{u}_k(t), \overline{\boldsymbol{u}}_k(t), \boldsymbol{d}(t), t) \tag{10.3}$$

$$\boldsymbol{y}_k = \boldsymbol{g}_k(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), t) \tag{10.4}$$

($k = 1, \ldots, s$) be the resulting decomposition of Eqs. (10.1) and (10.2), where $\overline{\boldsymbol{x}}_k$ gathers all the components of $\boldsymbol{x}$ except $\boldsymbol{x}_k$.

Each equation in (10.3), (10.4) can be interpreted as describing the behaviour of a subsystem Σ_k with $\boldsymbol{u}_k \in |\mathcal{R}^{m_k}$ the local control vector associated with a subset I_k of the actuators, $\boldsymbol{y}_k \in |\mathcal{R}^{p_k}$ the local measurement vector associated with a subset J_k of the sensors and $\boldsymbol{x}_k \in |\mathcal{R}^{n_k}$ the local state. Note that $\mathcal{I} = \{I_k,\ k = 1, \ldots, s\}$ and $\mathcal{J} = \{J_k,\ k = 1, \ldots, s\}$ are partitions of I and J.

The functions $\boldsymbol{f}_k(\boldsymbol{x}_k, \overline{\boldsymbol{x}}_k, \boldsymbol{u}_k, \overline{\boldsymbol{u}}_k, \boldsymbol{d}, t)$ can take different forms. A specific case occurs when $\boldsymbol{f}_k(\boldsymbol{x}_k, \overline{\boldsymbol{x}}_k, \boldsymbol{u}_k, \overline{\boldsymbol{u}}_k, \boldsymbol{d}, t)$ is decomposable, namely it is the sum of two functions

$$\boldsymbol{f}_k(\boldsymbol{x}_k, \overline{\boldsymbol{x}}_k, \boldsymbol{u}_k, \overline{\boldsymbol{u}}_k, \boldsymbol{d}, t) = \boldsymbol{f}_k^{\text{self}}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{d}, t) + \boldsymbol{f}_k^{\text{coupled}}(\overline{\boldsymbol{x}}_k, \overline{\boldsymbol{u}}_k, \boldsymbol{d}, t),$$

where

- $\boldsymbol{f}_k^{\text{self}}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{d}, t)$ describes the self-dynamics of subsystem Σ_k and
- $\boldsymbol{f}_k^{\text{coupled}}(\overline{\boldsymbol{x}}_k, \overline{\boldsymbol{u}}_k, \boldsymbol{d}, t)$ describes the coupled dynamics with respect to the other subsystems (meaning the influence of the other subsystems on subsystem Σ_k).

Note that other decompositions of Σ could be defined, from the system global model (10.1), (10.2) by changing the value of s and the partitions of $\boldsymbol{u}$, $\boldsymbol{y}$ and $\boldsymbol{x}$. In practice, however, there is usually a natural decomposition into subprocesses associated with the global process to be controlled (then, each subsystem describes a given subprocess) or with the control system. Indeed, in large-scale processes, the control system is composed of several computing devices, each of them running some part of the real-time control algorithms (diagnosis, supervision, management, etc.), and the sensors and actuators are connected to the distributed control system through hubs and communication networks. In order to address distributed systems, the simple network architecture in which each subsystem Σ_k performs a part of the overall control and a part of the overall diagnosis is considered, as illustrated in Fig. 10.1.

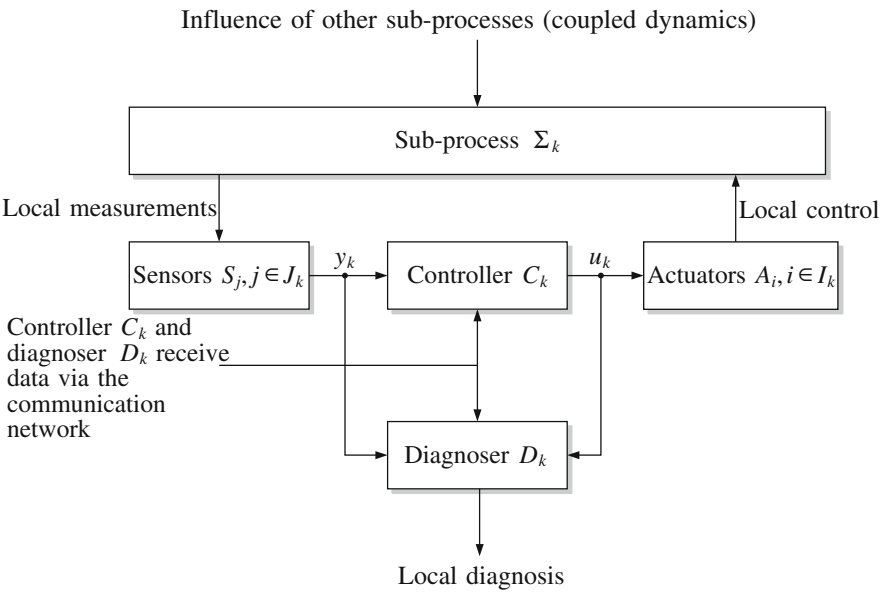


Fig. 10.1 Local controller and diagnoser

Example 10.1 System decomposition
Consider the sixth-order linear time invariant system

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t)$$

with

$$\boldsymbol{x}^{\mathrm{T}} = (x_1, x_2, x_3, x_4, x_5, x_6),$$

controlled by a set of 5 actuators ($I = \{1, 2, 3, 4, 5\}$) whose control signals are components of the vector

$$\boldsymbol{u}^{\mathrm{T}} = (u_1, u_2, u_3, u_4, u_5).$$

There are $2^6 - 2$ different ways to decompose this system into two subsystems. Indeed, each decomposition is obtained by considering a non-empty subset of the six states as the local state of the first subsystem and the remaining subset (provided it is non-empty) as the local state of the second subsystem.

More generally, the number of possible decompositions into s subsystems is the number of partitions of the state variables into s non-empty classes. If covers are considered instead of partitions, the result is a decomposition into overlapping subsystems, a case we shall not consider here. For example, with the matrices

$$\boldsymbol{A} = \begin{pmatrix} -1 & 2 & 1 & -1 & 0 & 1 \\ 0 & 1 & 2 & 0 & 0 & 0 \\ 0.5 & -0.5 & -2 & 0 & 1 & 1 \\ 1 & -1 & -1 & -2 & 0.4 & 0 \\ 0 & 0 & 0 & 1 & -3 & 1 \\ 2 & 0 & -1 & -2 & 0 & -4 \end{pmatrix}, \quad \boldsymbol{B} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix} \tag{10.5}$$

and the decomposition (x_1, x_2), (x_3, x_4, x_5, x_6), the two subsystems are, respectively,

$$\Sigma_1 : \quad \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -1 & 2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 0.5 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + \begin{pmatrix} 1 & -1 & 0 & 1 \\ 2 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix}$$

and

$$\Sigma_2 : \quad \begin{cases} \begin{pmatrix} \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{pmatrix} = \begin{pmatrix} -2 & 0 & 1 & 1 \\ -1 & -2 & 0.4 & 0 \\ 0 & 1 & -3 & 1 \\ -1 & -2 & 0 & -4 \end{pmatrix} \begin{pmatrix} x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} u_3 \\ u_4 \\ u_5 \end{pmatrix} \\ \qquad + \begin{pmatrix} 0.5 & -0.5 \\ 1 & -1 \\ 0 & 0 \\ 2 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}. \end{cases}$$

Note that for linear systems, the state equations are decomposable whatever the partition of the state that is considered, provided that the control signals do not simultaneously act on different subsystems. In this example, actuator 4 directly influences the state variables x_4 and x_5 and, therefore, decompositions in which these variables would belong to different subsystems would not enjoy the property that their state equations are decomposable. Associating a controller with each subsystem results in the determination of the control signals $u_1,\ u_2$ by Σ_1 and the determination of $u_3,\ u_4,\ u_5$ by Σ_2. Note that, unlike the control decomposition, the diagnosis decomposition is not implied by the system decomposition. Indeed, in addition to the computation of $u_1,\ u_2$, the computing device of subsystem Σ_1 could be assigned any part of some overall diagnosis algorithm (provided it is fed with the appropriate data and has enough computing power), and the same applies of course to subsystem Σ_2.

In the sequel, this example will be continued under the assumption that there is some physical reason to distinguish four subsystems based on the partition of the state (x_1, x_2), (x_3), (x_4, x_5), (x_6) and the partition of the actuators $I_1 = \{1,\ 2\}, I_2 = \{3\}, I_3 = \{4\}, I_4 = \{5\}$. The considered system decomposition is

$$\Sigma_1 : \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -1 & 2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 0.5 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + \cdots + \begin{pmatrix} 1 & -1 & 0 & 1 \\ 2 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix}$$

$$\Sigma_2 : \quad \dot{x}_3 = -2x_3 + u_3 + \begin{pmatrix} 0.5 & -0.5 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix}$$

$$\Sigma_3 : \quad \begin{pmatrix} \dot{x}_4 \\ \dot{x}_5 \end{pmatrix} = \begin{pmatrix} -2 & 0.4 \\ 1 & -3 \end{pmatrix} \begin{pmatrix} x_4 \\ x_5 \end{pmatrix} + \begin{pmatrix} 2 \\ 1 \end{pmatrix} u_4 + \cdots + \begin{pmatrix} 1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_6 \end{pmatrix}$$

$$\Sigma_4 : \quad \dot{x}_6 = -4x_6 + 2u_5 + \begin{pmatrix} 2 & 0 & -1 & -2 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}. \square$$

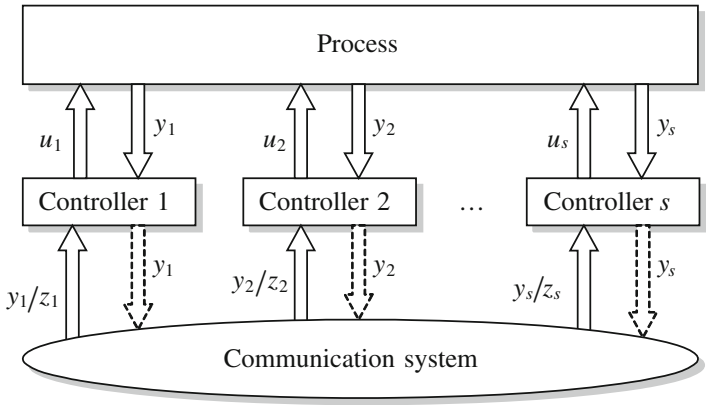


Fig. 10.2 Distributed system architecture

10.2.2 Distributed Control

Local controllers. Assuming that each subsystem Σ_k to be equipped with its own controller means that the overall control (i.e. the determination of vector $\boldsymbol{u}$) is distributed among the s controllers and each of them is in charge of computing the subvector $\boldsymbol{u}_k$. The design of efficient control algorithms might need some controllers to use more measurements than the locally available ones. This is possible, thanks to the existence of a communication network such that local controllers can use the measurements $\boldsymbol{z}_k \in |\mathcal{R}^{\pi_k}$ provided by a subset of sensors Z_k. Since the local measurements are always available to subsystem Σ_k, the relation $J_k \subseteq Z_k \subseteq J$ holds and $Z_k \setminus J_k$ is the set of remote sensors whose measurements are made available to Σ_k over the communication network. Since $\mathcal{J} = \{J_k,\ k = 1, \ldots, s\}$ is a partition of J, it follows that $\{Z_k,\ k = 1, \ldots, s\}$ is a cover of J, i.e. one has $Z_k \neq \emptyset$, $(k = 1, \ldots, s)$ and $\cup_{k=1,\ldots,s} Z_k = J$. Figure 10.2 displays the corresponding architecture (dotted arrows mean that the variables may, or may not, be communicated). In the sequel, for the sake of conciseness, we use the same notation for the sensors and the signals they deliver (should they be ordered as vectors or not), for example $\boldsymbol{y}_k \subseteq \boldsymbol{z}_k \subseteq \boldsymbol{y}$, $\boldsymbol{z}_k \setminus \boldsymbol{y}_k$, etc.

Information pattern. Given a system decomposition, the s-tuple

$$\mathcal{Z} = \{\boldsymbol{z}_k,\ k = 1, \ldots, s\}$$

is an *information pattern*. The *full information pattern* is

$$\mathcal{Z}_{\max} = \{\boldsymbol{z}_k = \boldsymbol{y},\ k = 1, \ldots, s\},$$

meaning that all the measurement signals $\boldsymbol{y}$ are available to each local controller. Note that this is nothing but the centralised control architecture, when $s = 1$, and a

distributed implementation of the centralised control when $s > 1$. On the contrary, under the *local information pattern*

$$\mathcal{Z}_{\min} = \left\{ z_k = \boldsymbol{y}_k,\ k = 1, \ldots, s \right\},$$

only locally produced measurements are used by each local controller, which characterises the decentralised control scheme.

Example 10.2 Local controllers
Assume that the system of Example 10.1 is equipped with 4 sensors $J = \{1,\ 2,\ 3,\ 4\}$ providing the measurement signals

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 0\ 1\ 0\ 0\ 0\ 0 \\ 0\ 0\ 1\ 0\ 0\ 0 \\ 0\ 0\ 0\ 1\ 0\ 0 \\ 0\ 0\ 0\ 0\ 0\ 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix}$$

and each controller is interfaced with one of them as follows:

$$J_1 = \{1\},\ J_2 = \{2\},\ J_3 = \{3\},\ J_4 = \{4\}.$$

Assume that output feedback is investigated, for the sake of simplicity. Consider for example the information pattern $\mathcal{Z} = \{(y_1, y_2), y_2, (y_1, y_3), (y_2, y_4)\}$. It needs the signal y_1 to be communicated from Σ_1 to Σ_3 and the signal y_2 to be communicated from Σ_2 to Σ_1 and Σ_4, and it allows to use the controllers

$$\begin{aligned} u_1(t) &= k_{11} y_1(t) + k_{12} y_2(t) \\ u_2(t) &= k_{22} y_2(t) \\ u_3(t) &= k_{31} y_1(t) + k_{33} y_3(t) \\ u_4(t) &= k_{42} y_2(t) + k_{44} y_4(t), \end{aligned}$$

where the k_{ij} are the real output feedback gains. By comparison, the full information pattern $\mathcal{Z}_{\max} = \{\boldsymbol{y},\ \boldsymbol{y},\ \boldsymbol{y},\ \boldsymbol{y}\}$ allows the design $\boldsymbol{u}(t) = \boldsymbol{K}\boldsymbol{y}(t)$ with $K \in |\mathcal{R}^{5\times 4}$, but needs all the measurements to be communicated, while in the local information pattern

$$\mathcal{Z}_{\min} = \{y_1,\ y_2,\ y_3,\ y_4\}$$

associated with decentralised control, no variable at all is communicated, but the output feedback must satisfy the constraints $u_i(t) = k_{ii} y_i(t)$ where the k_{ii} are real numbers. □

10.2.3 Distributed Diagnosis

Whatever the way they have been designed (analytical redundancy relations, observers, identification-based designs), a centralised diagnoser evaluates all the residuals using the data available to it through its connection with the system sensors and controllers. In a distributed architecture, each subsystem Σ_k, $(k = 1, \ldots, s)$ runs its own local diagnoser, defined by a pair $(\boldsymbol{r}_k, \delta_k)$ where $\boldsymbol{r}_k$ are the residuals it has been assigned and δ_k is a decision procedure on the residuals $\boldsymbol{r}_k$. Let $\boldsymbol{z}_k^a \subseteq \boldsymbol{u}$ and $\boldsymbol{z}_k^s \subseteq \boldsymbol{y}$ be the control and measurement signals whose knowledge is needed to run the residuals $\boldsymbol{r}_k$ that have been assigned to subsystem Σ_k ($\boldsymbol{z}_k^a$ and $\boldsymbol{z}_k^s$ are determined by the computation form of the residuals in $\boldsymbol{r}_k$). Then, the information pattern that allows the local diagnosers to perform their task is

$$\mathcal{Z} = \left\{ \left(\boldsymbol{z}_k^a, \boldsymbol{z}_k^s \right), k = 1, \ldots, s \right\}.$$

The full information pattern is $\mathcal{Z}_{\max} = \{(\boldsymbol{u}, \boldsymbol{y}), k = 1, \ldots, s\}$, while the local information pattern is $\mathcal{Z}_{\min} = \left\{ \left(\boldsymbol{u}_k, \boldsymbol{y}_k \right), k = 1, \ldots, s \right\}$.

10.2.4 Communication Cost

Given an information pattern $\mathcal{Z}$, the sets $\boldsymbol{z}_k \setminus \boldsymbol{y}_k$, $(k = 1, \ldots, s)$ contain those measurement signals that are needed by, but are not locally available to, the controller of subsystem Σ_k. Similarly, the sets $\boldsymbol{z}_k^a \setminus \boldsymbol{u}_k$ and $\boldsymbol{z}_k^s \setminus \boldsymbol{y}_k$ contain the control signals (resp. the measurement signals) that are needed by, but are not locally available to, the diagnoser of subsystem Σ_k. Those data are received through the communication network, that involves some communication cost. Whatever the network and the communication protocol, the communication cost would clearly depend on the variables coding, transmission rate, checking procedures, management strategy, etc., and it would be growing with the number of communicated variables. It is assumed that the communication cost is expressed by a function $com\,(K)$ where K is the set of communicated variables, such that $com\,(\emptyset) = 0$ and $K_1 \subseteq K_2 \Rightarrow com\,(K_1) \leq com\,(K_2)$.

10.2.5 Communication Schemes

Among many available communication schemes, this chapter builds on the publisher/subscriber and the bilateral agreements based ones. The publisher/subscriber scheme is associated with diffusion-based networks and is well suited to factory communication protocols, like communication between intelligent sensors, actuators and subsystems, while the bilateral agreements scheme is well suited to describe the communication between autonomous agents.

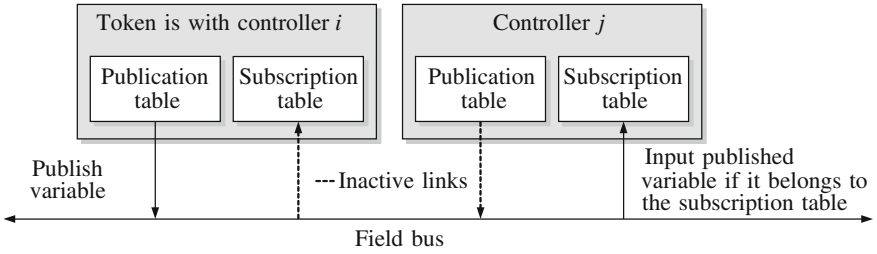


Fig. 10.3 The publisher/subscriber scheme

Diffusion-based networks. In diffusion-based networks,[1] a variable that is published in the communication system is available to all the subsystems that subscribe to it. The process globally works as follows:

- When a controller gets the token, it takes control of the communication bus and publishes the identifier and the value of the variables it is in charge of publishing (they are in its publication table);
- The other controllers recognise the identifier of a variable they have subscribed to (the list is in their subscription table). If recognised, they input its value; and
- The token passes to the next controller.

Figure 10.3 illustrates the publisher/subscriber scheme.

Bilateral agreements. In this scheme, subsystems establish bilateral agreements by which they share their data. Let a be the binary relation such that $a\left(\Sigma_i,\ \Sigma_j\right) = 1$ if Σ_i and Σ_j share their data, $a\left(\Sigma_i,\ \Sigma_j\right) = 0$ otherwise. Note that a being reflexive, symmetric and transitive, its graph $\mathcal{A}$ (which represents the set of agreements) involves a partition of all subsystems $\{\Sigma_k,\ k = 1, \ldots, s\}$ into equivalence classes $\mathcal{E}\left(\mathcal{A}\right) = \{E_l,\ l = 1, \ldots, \sigma\}$, with $\sigma = s$ when $\mathcal{A}$ is empty and $\sigma < s$ otherwise. It follows that the same data $z\left(E_l\right)$ are available to all the subsystems that belong to the same class E_l, as illustrated by Fig. 10.4 for a system with five distributed controllers and two equivalence classes.

Other schemes. Teams of autonomous agents most often use wireless communications, which restricts the communication possibilities of each agent to a subset of the other agents in its neighbourhood. In such applications, the network is described by a graph whose nodes $\mathcal{N}_i$ are the individual agents, and an arc between agents $\mathcal{N}_i$ and $\mathcal{N}_j$ indicates that the first can send data to the second. Such communication schemes are not considered in this chapter.

[1] Examples of diffusion-based networks are the Factory Instrumentation Protocol defined by the European Standards EN50170 and the IEC 61158/IEC 61784 Communication Profile Family 5.

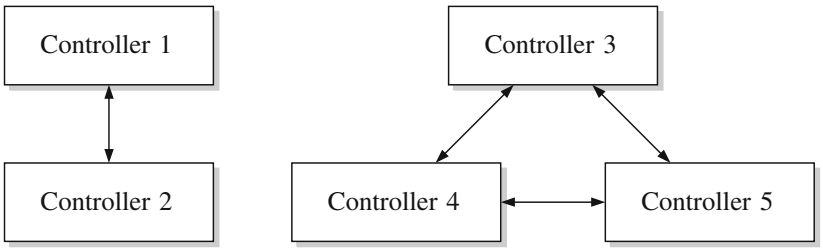


Fig. 10.4 Two agreement classes between 5 controllers

10.3 Distributed Diagnosis Design

In order to design a local diagnoser for each subsystem, two problems are to be solved:

1. characterise the system-level diagnosis that follows from the subsystem-level diagnosis and
2. design the local diagnosers so as to obtain specified results at the global system level.

These problems are addressed in this section and in the next one.

A direct means to evaluate the system-level diagnosis achieved by a set of distributed diagnosers is to compare it with the results that would be obtained with the overall (centralised) diagnoser. In order to develop this comparison, we first highlight the parameters that shape the design of a global diagnoser, namely its structural and its quantitative properties, which were, respectively, presented in Chaps. 5, 6 and 7.

10.3.1 Structural Diagnoser

Remember that from a structural point of view, the dynamical behaviour of a system Σ is described by a set of variables $\mathcal{V}$ and a set of constraints $\mathcal{C}$ that are satisfied when it is healthy. For continuous systems, the constraints $\mathcal{C}$ are algebraic and differential equations, the classical formulation of which is recalled here:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), t\right) \tag{10.6}$$

$$\boldsymbol{y}(t) = \boldsymbol{g}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), t\right), \tag{10.7}$$

where $\boldsymbol{x} \in |\mathcal{R}^n$ is the state, $\boldsymbol{u} \in |\mathcal{R}^m$ and $\boldsymbol{d} \in |\mathcal{R}^q$ are, respectively, the known and unknown inputs, and $\boldsymbol{y} \in |\mathcal{R}^p$ are the known measured outputs. In the sequel, we still use the same notation for sets and vectors of variables, as well as for sets and vectors of constraints, because no confusion is possible. Note that algebraic constraints on the state can easily be introduced via Eq. (10.7) as a subset of sensors whose output is constant and equal to zero. The variables $\mathcal{V}$ are partitioned into known $\mathcal{K} = \boldsymbol{u} \cup \boldsymbol{y}$ and unknown $\mathcal{X} = \boldsymbol{x} \cup \boldsymbol{d}$ variables.

In order to characterise the global diagnoser's properties, we first recall how analytical redundancy relations (ARR) are exhibited from the system canonical decomposition and we present a basic result from the logical theory of diagnosis.

Canonical decomposition. The structural analysis of Σ is the analysis of the bipartite graph $\mathcal{G} = (\mathcal{C}, \mathcal{Z}, \mathcal{E})$ introduced in Sect. 5.2, where $\mathcal{Z}$ is the set of variables, $\mathcal{C}$ is the set of constraints and $\mathcal{E}$ is the set of edges in which each pair $(c_i, z_j) \in \mathcal{E}$ means that the variable z_j appears in the constraint c_i. The DM decomposition of the graph $\mathcal{G}$ is explained in Sect. 5.4.1 that provides three canonical subsystems of Σ, namely an over-constrained, a just-constrained and an under-constrained subsystem. The over-constrained subsystem exhibits more than one solution to the complete matching problem of its unknown variables, while in the just-constrained subsystem, the complete matching is unique, and there is no complete matching in the under-constrained subsystem. Remember that the set $\mathcal{Z}$ of variables is decomposed into the set $\mathcal{K}$ of known variables and the set $\mathcal{X}$ of unknown variables and that a complete matching of the unknown variables allows to express the unknown variables as functions of the known variables, which means that it is possible to eliminate them in any constraint where they appear, simply by replacing them by their expression.

Analytical redundancy relations. The over-constrained subsystem is the monitorable part of Σ. Indeed, the existence of more than one complete matching of its unknown variables implies that a set of compatibility condition must be satisfied by the variables in $\mathcal{K}$ for Eqs. (10.6) and (10.7) to be consistent. These conditions are the analytical redundancy relations (ARR).

The essence of ARR-based diagnosis is to check whether the ARR are satisfied or not by the known data. This is done via a set of *residuals* whose computation involves only known variables, and whose value should be zero in normal operation. Let $\boldsymbol{r}(\mathcal{C}, \mathcal{K})$ be the set of residuals associated with the set of constraints $\mathcal{C}$ and the known variables $\mathcal{K}$.

The computation of each residual $\rho \in \boldsymbol{r}(\mathcal{C}, \mathcal{K})$ involves a subset $\mathcal{K}(\rho) \subseteq \mathcal{K}$ of known variables and a subset $\mathcal{C}(\rho) \subseteq \mathcal{C}$ of constraints. $\mathcal{K}(\rho)$ is known from its *computation form*, and $\mathcal{C}(\rho) \subseteq \mathcal{C}$ defines its *structure*.

Remark 10.1 ARR-based residuals generally call for derivatives of the known variables, which is often argued against them in real-time applications. However, the derivation order can be limited (at the cost of reducing the number of found ARR) and moreover, it is in general possible to design observers whose outputs are equivalent and do not suffer the noise sensitivity issues. □

10.3.2 Logical Theory of Diagnosis

The logical theory of diagnosis is a tool that will be used to explain the coordination of several local diagnoses. It rests on the residuals signatures presented in Chap. 5 that are further analysed here.

Structural detectability. Since a fault changes one (or several) system constraint(s), it follows that there is a contradiction between the two statements:

1. a residual $\rho(t)$ is *falsified* by the data ($\rho(t) \neq 0$) and
2. all the constraints in its structure $\mathcal{C}(\rho)$ hold true.

The structure of a falsified residual is named as *conflict*, meaning it contains at least one faulty (untrue) constraint. It follows that for a faulty constraint to be detectable, it must belong at least to one residual's structure. The set of faults structurally detectable by the residuals $\boldsymbol{r}$ is therefore $\mathcal{D} = \cup_{\rho \in r} \mathcal{C}(\rho)$, while the non-detectable faults are $\overline{\mathcal{D}} = \mathcal{C} \setminus \mathcal{D}$. Remark that this explains the term *monitorable* that applies to the over-constrained subsystem, because it is the only one that produces residuals.

Remark 10.2 It is important to remark that the system canonical decomposition is unique. It follows that the set of detectable faults is also unique. In particular, it cannot be extended by ARR combinations. □

Structural isolability. A constraint $c \in \mathcal{D}$ partitions the residuals $\boldsymbol{r}$ into $\boldsymbol{r}_1(c)$ whose structure contains c and $\boldsymbol{r}_0(c)$ whose structure does not contain c. Let us first consider single faults: when c is faulty, the residuals $\boldsymbol{r}_0(c)$ are satisfied while the residuals $\boldsymbol{r}_1(c)$ are falsified. The *signature* of fault c is the vector $\boldsymbol{s}(c)$ whose jth component gives the status of residual r_j (0 when satisfied, 1 when falsified). The diagnoser is characterised by its *distinguishability* partition $\left\{\mathcal{D}^i, i = 0, 1, 2, \ldots\right\}$ where $\mathcal{D}^0 = \text{OK} \cup \overline{\mathcal{D}}$ are the situations that have the same signature as the healthy system, namely $\boldsymbol{s}\left(\mathcal{D}^0\right)$ such that $\boldsymbol{r}_0(\mathcal{D}^0) = \boldsymbol{r}$ and $\boldsymbol{r}_1\left(\mathcal{D}^0\right) = \emptyset$, and $\mathcal{D}^i$, $(i \neq 0)$ are the faulty situations that have the same signature $\boldsymbol{s}\left(\mathcal{D}^i\right)$.

Assuming that fault cancellations do not occur, a multiple fault

$$C = \{c_i, i = 1, 2, \ldots\}$$

has the signature $\boldsymbol{s}(C)$ such that

$$\boldsymbol{r}_1(C) = \cup_{c \in C} \boldsymbol{r}_1(c)$$

and

$$\boldsymbol{r}_0(C) = \boldsymbol{r} \setminus \boldsymbol{r}_1(C).$$

Note that a special case of multiple faults is addressed in the partition

$$\left\{\mathcal{D}^i, i = 0, 1, 2, \ldots\right\}$$

because the signature $\boldsymbol{s}\left(\mathcal{D}^i\right)$ characterises any subset of faults that belong to the same class $\mathcal{D}^i$, $(i \neq 0)$. The set of fault signatures can be studied by considering every single and multiple faults. However, it is simpler to rely on the following result from the logical theory of diagnosis.

Minimal hitting sets and diagnosis. Let $\boldsymbol{r} = \boldsymbol{r}_\mathrm{s} \cup \boldsymbol{r}_\mathrm{f}$ be a partition of the residuals $\boldsymbol{r}$ into satisfied residuals $\boldsymbol{r}_\mathrm{s}$ and falsified residuals $\boldsymbol{r}_\mathrm{f}$. A minimal subset of constraints Δ^i whose faults result in this very partition is a *possible diagnosis*. Since more than one such subset may exist, the *overall diagnosis* is the set of possibilities $\Delta = \left\{\Delta^i, i = 1, 2, \ldots\right\}$. Note that this definition automatically includes simple and multiple faults.

Theorem 10.1 (Minimal hitting set) *Let* $\{\mathcal{C}(\rho), \rho \in \boldsymbol{r}_\mathrm{f}\}$ *be the set of conflicts associated with the partition of the residuals into* $\boldsymbol{r} = \boldsymbol{r}_\mathrm{s} \cup \boldsymbol{r}_\mathrm{f}$. *A possible diagnosis is a minimal hitting set of* $\{\mathcal{C}(\rho), \rho \in \boldsymbol{r}_\mathrm{f}\}$.

A subset of constraints H is a hitting set of $\{\mathcal{C}(\rho), \rho \in \boldsymbol{r}_\mathrm{f}\}$ if the two relations

- $H \subseteq \cup_{\rho \in \boldsymbol{r}_\mathrm{f}} \mathcal{C}(\rho)$ and
- $H \cap \mathcal{C}(\rho) \neq \emptyset, \forall \rho \in \boldsymbol{r}_\mathrm{f}$

hold. H is minimal if no proper subset of H satisfies these two conditions. In words, H is a minimal subset of constraints such that

- each of them belongs to at least one conflict and
- a corresponding multiple fault falsifies every residual in $\boldsymbol{r}_\mathrm{f}$.

Example 10.3 Ship with dual measurements
The simplified non-linear model of a ship steering system with dual measurements was considered in Chap. 5. The unknown variables are the heading angle ψ, the turn rate ω and the rudder angle δ. There are four known variables $\{y_1, y_2, y_3, y_4\}$. From the state and measurement equations

$$\begin{array}{l} c_1: \\ c_2: \end{array} \begin{pmatrix} \dot{\omega} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} \eta_1 \omega + \eta_3 \omega^3 + \delta \\ \omega \end{pmatrix}$$

$$\begin{array}{l} m_1: \\ m_2: \\ m_3: \\ m_4: \end{array} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} \psi \\ \psi \\ \dot{\psi} \\ \delta \end{pmatrix},$$

one finds three residuals whose computation forms and structures are, respectively, given by Eq. (10.8) and Table 10.1:

$$\begin{aligned} \rho_1 &= y_2 - y_1 \\ \rho_2 &= \dot{y}_1 - y_3 \\ \rho_3 &= \dot{y}_3 - \eta_1 y_3 - \eta_3 y_3^3 - y_4. \end{aligned} \tag{10.8}$$

A fault in any constraint is detectable since there is at least a "1" in each column of the signature table. Some faults are not isolable, since they have identical signatures. The resulting equivalence classes are $\mathcal{D}^1 = \{m_1\}$, $\mathcal{D}^2 = \{m_2\}$, $\mathcal{D}^3 = \{m_3\}$ and $\mathcal{D}^4 = \{m_4, c_1, c_2\}$, which gives the distinguishability Table 10.2 (*OK* has been re-labelled as $\mathcal{D}^0$):

Table 10.1 Structures of the ship example residuals

	OK	m_1	m_2	m_3	m_4	c_1	c_2
ρ_1	0	1	1	0	0	0	0
ρ_2	0	1	0	1	0	0	0
ρ_3	0	0	0	1	1	1	1

Table 10.2 Distinguishability table of the ship example

	$\mathcal{D}^0$	$\mathcal{D}^1$	$\mathcal{D}^2$	$\mathcal{D}^3$	$\mathcal{D}^4$
ρ_1	0	1	1	0	0
ρ_2	0	1	0	1	0
ρ_3	0	0	0	1	1

Assume residual ρ_1 is satisfied by the real-time measurements, residuals ρ_2, ρ_3 are falsified and the signature 011 directly leads to the diagnosis $\mathcal{D}^3$. Now, suppose that although there is no signature 111 in the table, all three residuals are falsified by the real-time measurements. Using the notation $\mathcal{D}^i \times \mathcal{D}^j$ for a double fault diagnosis in which the first fault is a constraint in $\mathcal{D}^i$ and the second fault a constraint in $\mathcal{D}^j$, it is seen that this is indeed possible as the result of the multiple faults $\mathcal{D}^1 \times \mathcal{D}^3 \cup \mathcal{D}^1 \times \mathcal{D}^4 \cup \mathcal{D}^2 \times \mathcal{D}^3$, as it can be visually checked on Fig. 10.5 where $\mathcal{D}^1 \times \mathcal{D}^3$, $\mathcal{D}^1 \times \mathcal{D}^4$ and $\mathcal{D}^2 \times \mathcal{D}^3$ are minimal hitting sets of $\{\mathcal{C}(\rho_1), \mathcal{C}(\rho_2), \mathcal{C}(\rho_3)\}$.

Considering all possible signatures (note that the signature 010 can never be obtained) gives the diagnosis Table 10.3.

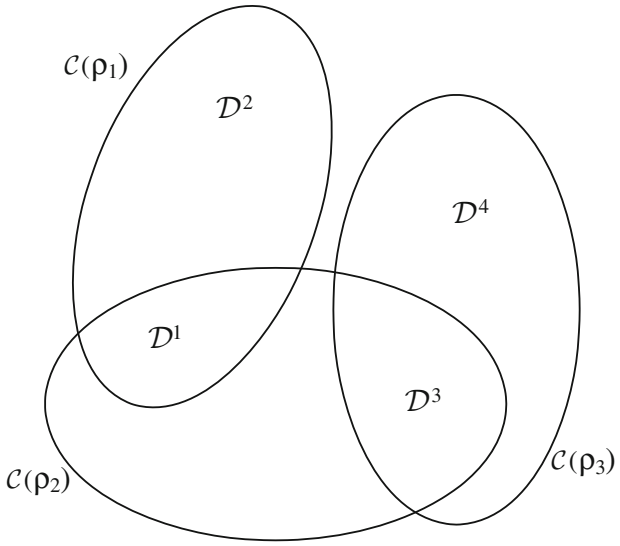


Fig. 10.5 The three conflicts associated with the signature 111

Table 10.3 Diagnosis table of the ship example

$\rho_1\,\rho_2\,\rho_3$	Diagnosis
000	$\mathcal{D}^0$
001	$\mathcal{D}^4$
010	cannot happen
011	$\mathcal{D}^3$
100	$\mathcal{D}^2$
101	$\mathcal{D}^2 \times \mathcal{D}^4$
110	$\mathcal{D}^1$
111	$\mathcal{D}^1 \times \left(\mathcal{D}^3 \cup \mathcal{D}^4\right) \cup \mathcal{D}^2 \times \mathcal{D}^3$

Remark 10.3 The conclusion obtained via the logical theory of diagnosis may contain many possible diagnosis, as it can be seen from the previous example, where there are three sets of possible double faults associated with the signature 111. All conclusions are indeed consistent from a logical point of view. However, in practical applications, one may have to select only one of them. Under the assumption that the joint probabilities of faults occurring in different constraints are known, it seems of course appropriate to select the most probable one. □

10.3.3 Practical Diagnoser and Real-Time Operation

Structural versus actual properties. Structural properties are necessary but not sufficient for actual properties to be true. A residual whose structure does not contain a given fault can by no means allow its detection, but a structurally detectable fault might never be detected in practice because the sensitivity of the residuals is too small, or because the signal/noise ratio does not allow its detection. Similarly, two isolable faults might never be isolated from each other if only a common subset of residuals is sensitive enough to them.

A practical diagnoser is a pair $(\boldsymbol{r}, \delta)$ where $\boldsymbol{r}$ is a set of residuals and δ is a decision procedure that checks the residuals status (satisfied/falsified), using the available knowledge on modelling errors, unknown inputs and measurement noises, in order to reduce false alarms, missed detections, detection delays and mis-isolations, as analysed in Chap. 7.

Real-time operation. The real-time operation of a practical diagnoser follows 4 steps (steps 3 and 4 are optional depending on the application):

1. Compute the value of the residuals $\boldsymbol{r}$ from the values of the known variables $\mathcal{K}$;
2. Fault detection: evaluate the residuals using the decision procedure δ and conclude whether a fault has occurred ($\boldsymbol{r}_{\mathrm{f}} \neq \emptyset$) or if the system can possibly be healthy ($\boldsymbol{r}_{\mathrm{s}} = \boldsymbol{r}$);

3. Fault isolation: find the minimal hitting sets consistent with the observed signature, if a detection has been fired; and
4. Fault estimation: estimate the model of the faulty system.

Whatever the complexity of steps 2, 3 and 4, they apply to the residuals $\boldsymbol{r}$ issued from the structural analysis. Implementing a centralised or a distributed diagnoser is therefore based on implementing a centralised or a distributed computation scheme for the residuals $\boldsymbol{r}$. This is why only the fault detection and isolation properties of structural diagnosers are considered in the sequel.

Centralised or distributed implementation. In a centralised system, the diagnoser is run by a single computing device that is connected with all the sensors and controllers. A centralised diagnosis implementation is based on the assumptions that

- the data involved in the computation form of any residual are available to the central computing device;
- there is no data transmission delay, or if some delay is unavoidable, all the data involved in the computation form of a given residual are available under compatible time stamps.

In a distributed diagnosis scheme, each subsystem Σ_k, $(k = 1, \ldots, s)$ runs its own diagnosis algorithm, and the above assumptions may be no longer satisfied:

- the data available to each subsystem depend on the information pattern that is implemented;
- the communication network may introduce transmission errors, data losses and unacceptable delays; and
- even when not faulty, the communication network may introduce different transmission delays for different variables, due to the network scheduling procedures.

The design of a distributed diagnosis scheme rises two interrelated problems:

- **Problem 1**. Given a residual vector $\boldsymbol{r}$ and a set of subsystems Σ_k, $(k = 1, \ldots, s)$ how to design an information pattern and how to distribute the residual computations between the different subsystems? and
- **Problem 2**. Given a set of local diagnosers how to achieve an overall decision that is consistent with the locally achieved ones?

Because it is needed to understand the coordination procedure in order to design the residuals distribution, we start with the solution of Problem 2.

10.3.4 Local Diagnosers and Their Coordination

As the data available to local diagnosers depend on the information pattern, we will need to manipulate these entities in order to understand the *subsystem-level diagnosis capabilities*. The *system-level coordination* of all the subsystem-level diagnosis will be addressed after.

Information pattern. We first give a formal definition of the set of all information patterns and define an order on this set.

Definition 10.1 (*Information pattern*) An information pattern is a set $\mathcal{Z} = \{z_k, k = 1, \ldots, s\}$, where z_k is a pair $\left(z_k^a, z_k^s\right)$ such that $\boldsymbol{u}_k \subseteq z_k^a \subseteq \boldsymbol{u}$ and $\boldsymbol{y}_k \subseteq z_k^s \subseteq \boldsymbol{y}$.

In other words, an information pattern is a s-tuple whose ith element is the subset of input/output data available to the ith subsystem. The set of information patterns is easily provided with a partial order relation defined as

$$\mathcal{Z}_1 \preceq \mathcal{Z}_2 \Leftrightarrow \forall k = 1, \ldots, s : z_{1,k}^a \subseteq z_{2,k}^a \wedge z_{1,k}^s \subseteq z_{2,k}^s.$$

In this case, $\mathcal{Z}_2$ is said to be *wider* than $\mathcal{Z}_1$—or $\mathcal{Z}_1$ is *narrower* than $\mathcal{Z}_2$. The minimal information pattern

$$\mathcal{Z}_{\min} = \left\{\left(\boldsymbol{u}_k, \boldsymbol{y}_k\right), k = 1, \ldots, s\right\}$$

is narrower than any other information pattern, and the maximal information pattern

$$\mathcal{Z}_{\max} = \{(\boldsymbol{u}, \boldsymbol{y}), k = 1, \ldots, s\}$$

is wider than any other one.

It follows from the definition that any information pattern

$$\mathcal{Z} = \left\{\left(z_k^a, z_k^s\right), k = 1, \ldots, s\right\}$$

is such that $\cup_{k=1,\ldots,s} z_k^a = \boldsymbol{u}$ and $\cup_{k=1,\ldots,s} z_k^s = \boldsymbol{y}$, in other words, the z_k^a, respectively, and the z_k^s are a cover of $\boldsymbol{u}$, respectively, of $\boldsymbol{y}$.

Local diagnosers. The structural analysis of the global system model (10.6), (10.7)

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), t\right)\\ \boldsymbol{y}(t) &= \boldsymbol{g}\left(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), t\right)\end{aligned}$$

results in the set of residuals $\boldsymbol{r}(\mathcal{C}, \mathcal{K})$ where $\mathcal{C} = \boldsymbol{f} \cup \boldsymbol{g}$ and $\mathcal{K} = \boldsymbol{u} \cup \boldsymbol{y}$. Assume that for some reason, we are interested in the structural analysis of the constraints $\mathcal{C}$ when there are less known variables than $\mathcal{K}$, namely $\mathcal{K}^- \subseteq \mathcal{K}$. The monotonicity property

$$\mathcal{K}^- \subseteq \mathcal{K} \Rightarrow \boldsymbol{r}\left(\mathcal{C}, \mathcal{K}^-\right) \subseteq \boldsymbol{r}(\mathcal{C}, \mathcal{K})$$

holds true, with the conclusion that if $\mathcal{Z} = \left\{\left(z_k^a, z_k^s\right), k = 1, \ldots, s\right\}$ is an information pattern by which the known variables available to subsystem Σ_k are $\mathcal{K}_k = \left(z_k^a, z_k^s\right)$, the subset of residuals that can be computed by subsystem Σ_k is $\boldsymbol{r}(\mathcal{C}, \mathcal{K}_k) \subseteq \boldsymbol{r}(\mathcal{C}, \mathcal{K})$.

Similarly, let $\mathcal{C}^- \subseteq \mathcal{C}$ be a subset of constraints, then whatever the known variables $\mathcal{K}$ one has $\boldsymbol{r}\left(\mathcal{C}^-, \mathcal{K}\right) \subseteq \boldsymbol{r}(\mathcal{C}, \mathcal{K})$, with the consequence that another monotonicity

property holds true under the information pattern $\mathcal{Z}$, namely

$$\mathcal{C}_k \subseteq \mathcal{C} \Rightarrow \boldsymbol{r}(\mathcal{C}_k, \mathcal{K}_k) \subseteq \boldsymbol{r}(\mathcal{C}, \mathcal{K}_k) \subseteq \boldsymbol{r}(\mathcal{C}, \mathcal{K}).$$

Definition 10.2 (*Distributed diagnosis scheme*) Let $\mathcal{C} = \{\mathcal{C}_k, k = 1, \ldots, s\}$ be a collection of constraint subsets and consider an information pattern $\mathcal{Z} = (z_k, k = 1, \ldots, s)$. A distributed diagnosis scheme is a set of local diagnosers $(\boldsymbol{r}(\mathcal{C}_k, z_k), \delta_k)$ where δ_k is the decision procedure associated with the evaluation of the residuals $\boldsymbol{r}(\mathcal{C}_k, z_k)$.

The subset of constraints $\mathcal{C}_k$ associated with a local diagnoser need not be the constraints $\boldsymbol{f}_k \cup \boldsymbol{g}_k$ that describe the behavioural model of subsystem Σ_k in Eqs. (10.3), (10.4).

Before we describe different diagnosis schemes associated with different choices of $\mathcal{C}$, let us first investigate the relation between local and global diagnosis.

Local versus global detection. From a structural point of view, each local diagnoser in a distributed diagnosis scheme is characterised by the partition

$$\left\{\mathcal{D}_k^i, i = 0, 1, 2, \ldots\right\}$$

that defines the system situations and it is able to distinguish from its different residual signatures. Note that such a local partition is necessarily wider than the partition $\left\{\mathcal{D}^i, i = 0, 1, 2, \ldots\right\}$ associated with the global diagnoser, since $\mathcal{D}_k^i$ is the union of several subsets $\mathcal{D}^j$ in the global distinguishability table.

In order to be detected, a fault must be detectable by at least one local diagnoser. The set of detectable faults in a distributed diagnosis scheme is therefore $\cup_{k=1,\ldots,s}\mathcal{D}_k$ where $\mathcal{D}_k = \cup_{i\neq 0}\mathcal{D}_k^i$ is the set of faults detectable by the local diagnoser Σ_k. As the global scheme can detect the faults in $\mathcal{D}$, the difference $\mathcal{D} \setminus \cup_{k=1,\ldots,s}\mathcal{D}_k$ characterises the loss of detectability caused by the distributed diagnosis with respect to the global diagnosis.

Example 10.3 (cont.) Ship with dual measurements
Using the three residuals ρ_1, ρ_2, ρ_3, the distinguishability partition associated with the ship example was $\mathcal{D}^0 = \{OK\}$, $\mathcal{D}^1 = \{m_1\}$, $\mathcal{D}^2 = \{m_2\}$, $\mathcal{D}^3 = \{m_3\}$ and $\mathcal{D}^4 = \{m_4, c_1, c_2\}$. Assume a distributed diagnosis where the local diagnoser 1 runs only residual ρ_1. Considering only the first row of Table 10.2, its local distinguishability partition is $\left\{\mathcal{D}_1^0, \mathcal{D}_1^1\right\}$, with $\mathcal{D}_1^0 = \mathcal{D}^0 \cup \mathcal{D}^3 \cup \mathcal{D}^4$ and $\mathcal{D}_1^1 = \mathcal{D}^1 \cup \mathcal{D}^2$. □

Local versus global isolation. In order to evaluate the combined performance of the local diagnosers, one needs a *coordination or aggregation* procedure that provides a global diagnosis from the set of local diagnosis. Without loss of generality, the coordination procedure can be analysed for the case of two local diagnosers.

Theorem 10.2 *Let $\Delta_k = \left\{\Delta_k^i, i \in i_k\right\}$ be the local diagnosis delivered by two local diagnosers ($k = 1, 2$), where $\Delta_k^0 = OK \cup \overline{\mathcal{D}}_k$, $\overline{\mathcal{D}}_k$ are the faults non-detectable*

by diagnoser k and each Δ_k^i, $(i \neq 0)$ *is a minimal hitting set of the conflicts* $\left\{\mathcal{C}(\rho), \rho \in \boldsymbol{r}_f(k)\right\}$. *Consistent diagnosis are obtained as*

$$\Delta_{12} = \left\{\Delta_{12}^{00}, \Delta_{12}^{0i}, \Delta_{12}^{j0}, \Delta_{12}^{ij}, i, j \neq 0\right\}, \tag{10.9}$$

where

$$\Delta_{12}^{00} = OK \cup \left(\overline{\mathcal{D}}_1 \times \overline{\mathcal{D}}_2\right) \tag{10.10}$$

$$i \neq 0 : \begin{cases} \Delta_{12}^{0i} = \overline{\mathcal{D}}_1 \times \left(\mathcal{D}_2^i \cap \overline{\mathcal{D}}_1\right) \\ \Delta_{12}^{i0} = \overline{\mathcal{D}}_2 \times \left(\mathcal{D}_1^i \cap \overline{\mathcal{D}}_2\right) \end{cases} \tag{10.11}$$

$$i, j \neq 0 : \Delta_{12}^{ij} = \mathcal{D}_1^i \times \mathcal{D}_2^j \tag{10.12}$$

under a simplification and a deletion rule:

1. *Simplification rule: a double fault that consists of a pair of identical faults is simplified into a single fault.*
2. *Deletion rule: non-minimal hitting sets are deleted.*

Understanding the coordination procedure is quite simple: let $\boldsymbol{r}(1)$ and $\boldsymbol{r}(2)$ be the residuals run by the local diagnosers. Associated with the signatures $\boldsymbol{r}(1) = \boldsymbol{r}_s(1) \cup \boldsymbol{r}_f(1)$ and $\boldsymbol{r}(2) = \boldsymbol{r}_s(2) \cup \boldsymbol{r}_f(2)$ are the conflicts $\mathcal{C}(1) = \{\mathcal{C}(\rho), \rho \in \boldsymbol{r}_f(1)\}$ and $\mathcal{C}(2) = \{\mathcal{C}(\rho), \rho \in \boldsymbol{r}_f(2)\}$. Four cases can be distinguished, according to the fact that $\mathcal{C}(1)$ and $\mathcal{C}(2)$ are empty or not.

- **Case 1**: $\boldsymbol{r}_s(1) = \boldsymbol{r}(1)$ and $\boldsymbol{r}_s(2) = \boldsymbol{r}(2)$. In this case, there is no conflict, and the two local diagnosis are $\Delta_1^0 = OK \cup \overline{\mathcal{D}}_1$ and $\Delta_2^0 = OK \cup \overline{\mathcal{D}}_2$, where $\overline{\mathcal{D}}_1$ (resp. $\overline{\mathcal{D}}_2$) are the faults non-detectable by Σ_1 (resp. by Σ_2). The global diagnosis consistent with the local ones is $OK \cup \left(\overline{\mathcal{D}}_1 \times \overline{\mathcal{D}}_2\right)$.
- **Case 2**: $\boldsymbol{r}_s(1) = \boldsymbol{r}(1)$ and $\boldsymbol{r}_f(2) \neq \emptyset$. In this case, the first diagnosis is $\Delta_1^0 = OK \cup \overline{\mathcal{D}}_1$, while the second is $\Delta_2^1 = \cup_{i \in i_2} \mathcal{D}^i$ where $\mathcal{D}^i, i \in i_2$ are the faults that have the signature $\boldsymbol{r}_s(2) \cup \boldsymbol{r}_f(2)$. The global diagnosis consistent with the local ones is $\overline{\mathcal{D}}_1 \times \left(\mathcal{D}_2^1 \cap \overline{\mathcal{D}}_1\right)$. Indeed, OK is inconsistent, since the residuals $\boldsymbol{r}_f(2)$ are falsified. Any fault in $\overline{\mathcal{D}}_1$ satisfies $\boldsymbol{r}(1)$ and any fault in $\mathcal{D}_2^1$ falsifies $\boldsymbol{r}_f(2)$. The reason why only faults in $\mathcal{D}_2^1 \cap \overline{\mathcal{D}}_1$ are considered is that faults that falsify $\boldsymbol{r}_f(2)$ must also satisfy $\boldsymbol{r}_s(1)$.
- **Case 3**: $\boldsymbol{r}_f(1) \neq \emptyset, \boldsymbol{r}_s(2) = \boldsymbol{r}(2)$ is similar to case 2.
- **Case 4**: $\boldsymbol{r}_f(1) \neq \emptyset, \boldsymbol{r}_f(2) \neq \emptyset$. In this case, the first diagnosis is $\Delta_1^1 = \cup_{i \in i_1} \mathcal{D}^i$ and the second is $\Delta_2^1 = \cup_{i \in i_2} \mathcal{D}^i$. Any double fault in $\mathcal{D}_1^1 \times \mathcal{D}_2^1$ is indeed possible.

The simplification rule follows from the fact that when the same fault is concluded to be present by each local diagnoser, the "double fault" is in fact a simple one. Finally, each $\Delta_k^i, i \neq 0$ being a minimal hitting set of the conflicts $\{\mathcal{C}(\rho), \rho \in \boldsymbol{r}_f(k)\}$, a pair $\mathcal{D}_1^i \times \mathcal{D}_2^j, i, j \neq 0$ is a hitting set of $\left\{\mathcal{C}(\rho), \rho \in \cup_{k=1,2} \boldsymbol{r}_f(k)\right\}$ and it provides a

possible conclusion as seen above. However, this hitting set may be non-minimal, and in this case it cannot be a possible diagnosis.

Remark 10.4 The coordination unit provides the overall diagnosis consistent with all the local diagnosers' conclusions. It may be implemented in the computing device of any subsystem (we are not discussing here its possible distribution). Technically, it receives the local subsystems' decisions and coordinates them according to the procedure of Theorem 10.2. Note that alternatively, the coordination could also be done by a direct combination of all the locally obtained signatures according to the global diagnoser's distinguishability table. □

Example 10.3 (cont.) Ship with dual measurements
Let us exemplify the coordination procedure in the ship with dual measurements assuming there are two computing devices Σ_1 and Σ_2, which are, respectively, interfaced with the measurement signals y_1, y_2 and y_3, y_4. Under the minimal information pattern, noted $\{(y_1, y_2), (y_3, y_4)\}$, they respectively run the residuals ρ_1 and ρ_3, since the computation form of ρ_2 is available to none of them. The local distinguishability tables are

	$\mathcal{D}^0 \cup \mathcal{D}^3 \cup \mathcal{D}^4$	$\mathcal{D}^1 \cup \mathcal{D}^2$
ρ_1	0	1

	$\mathcal{D}^0 \cup \mathcal{D}^1 \cup \mathcal{D}^2$	$\mathcal{D}^3 \cup \mathcal{D}^4$
ρ_3	0	1

and the application of Theorem 10.2 gives

$\rho_1 \rho_3$	Local diagnosis Δ_1	Local diagnosis Δ_2	Coordinated diagnosis Δ_{12}
00	$\mathcal{D}^0 \cup \mathcal{D}^3 \cup \mathcal{D}^4$	$\mathcal{D}^0 \cup \mathcal{D}^1 \cup \mathcal{D}^2$	$\mathcal{D}^0$
01	$\mathcal{D}^0 \cup \mathcal{D}^3 \cup \mathcal{D}^4$	$\mathcal{D}^3 \cup \mathcal{D}^4$	$\mathcal{D}^3 \cup \mathcal{D}^4$
10	$\mathcal{D}^1 \cup \mathcal{D}^2$	$\mathcal{D}^0 \cup \mathcal{D}^1 \cup \mathcal{D}^2$	$\mathcal{D}^1 \cup \mathcal{D}^2$
11	$\mathcal{D}^1 \cup \mathcal{D}^2$	$\mathcal{D}^3 \cup \mathcal{D}^4$	$\left(\mathcal{D}^1 \cup \mathcal{D}^2\right) \times \left(\mathcal{D}^3 \cup \mathcal{D}^4\right)$

It can be checked that the coordinated diagnosis is the same as the centralised diagnosis based on the two residuals ρ_1 and ρ_3. Indeed, the centralised distinguishability table would be

	$\mathcal{D}^0$	$\mathcal{D}^1 \cup \mathcal{D}^2$	$\mathcal{D}^3 \cup \mathcal{D}^4$
ρ_1	0	1	0
ρ_3	0	0	1

with the diagnosis

$\rho_1\,\rho_3$	Global diagnosis
00	$\mathcal{D}^0$
01	$\mathcal{D}^3 \cup \mathcal{D}^4$
10	$\mathcal{D}^1 \cup \mathcal{D}^2$
11	$\left(\mathcal{D}^1 \cup \mathcal{D}^2\right) \times \left(\mathcal{D}^3 \cup \mathcal{D}^4\right)$

Let us examine other information patterns. The publication of y_3 by Σ_2 and its subscription by Σ_1 allows the distributed diagnosis scheme ρ_1, ρ_2 by Σ_1 and ρ_3 by Σ_2. The local distinguishability tables and the coordination result are given in Tables 10.4, 10.5 and 10.6. □

Table 10.4 Local distinguishability table of Σ_1

	$\mathcal{D}^0 \cup \mathcal{D}^4$	$\mathcal{D}^1$	$\mathcal{D}^2$	$\mathcal{D}^3$
ρ_1	0	1	1	0
ρ_2	0	1	0	1

Table 10.5 Local distinguishability table of Σ_2

	$\mathcal{D}^0 \cup \mathcal{D}^1 \cup \mathcal{D}^2$	$\mathcal{D}^3 \cup \mathcal{D}^4$
ρ_3	0	1

Table 10.6 Coordination table for Σ_1 and Σ_2

$\rho_1\,\rho_2\,\rho_3$	Δ_1	Δ_2	Δ_{12}
000	$\mathcal{D}^0 \cup \mathcal{D}^4$	$\mathcal{D}^0 \cup \mathcal{D}^1 \cup \mathcal{D}^2$	$\mathcal{D}^0$
001	$\mathcal{D}^0 \cup \mathcal{D}^4$	$\mathcal{D}^3 \cup \mathcal{D}^4$	$\mathcal{D}^4$
010	$\mathcal{D}^3$	$\mathcal{D}^0 \cup \mathcal{D}^1 \cup \mathcal{D}^2$	cannot happen
011	$\mathcal{D}^3$	$\mathcal{D}^3 \cup \mathcal{D}^4$	$\mathcal{D}^3 \cup \mathcal{D}^4$
100	$\mathcal{D}^2$	$\mathcal{D}^0 \cup \mathcal{D}^1 \cup \mathcal{D}^2$	$\mathcal{D}^2$
101	$\mathcal{D}^2$	$\mathcal{D}^3 \cup \mathcal{D}^4$	$\mathcal{D}^2 \times \mathcal{D}^4$
110	$\mathcal{D}^1 \cup \left(\mathcal{D}^2 \times \mathcal{D}^3\right)$	$\mathcal{D}^0 \cup \mathcal{D}^1 \cup \mathcal{D}^2$	$\mathcal{D}^1 \cup \mathcal{D}^2$
111	$\mathcal{D}^1 \cup \left(\mathcal{D}^2 \times \mathcal{D}^3\right)$	$\mathcal{D}^3 \cup \mathcal{D}^4$	$\mathcal{D}^1 \times \left(\mathcal{D}^3 \cup \mathcal{D}^4\right) \cup \mathcal{D}^2 \times \mathcal{D}^3$

10.3.5 Distribution Schemes

We now describe different distributed diagnosis schemes, associated with different choices of the collection $\mathcal{C} = \{\mathcal{C}_k, k = 1, \ldots, s\}$.

Global diagnoser in one subsystem. Let $\mathcal{C}_{\max,k}$ be defined by $\mathcal{C}_i = \emptyset, i \neq k$ and $\mathcal{C}_k = \mathcal{C}$, and the information pattern $\mathcal{Z}_{\max,k}$ be such that $z_k = \boldsymbol{u} \cup \boldsymbol{y}$. Then subsystem Σ_k runs the global diagnosis algorithm, while the other subsystems do not perform any diagnosis at all.

Global diagnoser with replicas. Let K be a subset of subsystems, let $\mathcal{C}_{\max,K}$ be defined by $\mathcal{C}_i = \emptyset$, $(i \notin K)$ and $\mathcal{C}_i = \mathcal{C}$, $(i \in K)$ and let $\mathcal{Z}_{\max,K}$ be an information pattern such that $\forall i \in K : z_i = \boldsymbol{u} \cup \boldsymbol{y}$, then each subsystem in K runs a *replica* of the global diagnoser, while the other ones do not perform any diagnosis at all.

Decentralised diagnosers. Under the collection $\mathcal{C}_{\max} = \{\mathcal{C}_k = \mathcal{C}, k = 1, \ldots, s\}$ and the local information pattern $\mathcal{Z}_{\min} = \left\{\mathcal{K}_k = \boldsymbol{u}_k \cup \boldsymbol{y}_k, k = 1, \ldots, s\right\}$, each subsystem runs the residuals whose computation form uses the local variables $\boldsymbol{u}_k \cup \boldsymbol{y}_k$. This scheme needs no data transmission for the computation of the local residuals (but communication is still needed for the coordination task). It may yield weak results, because only a subset of the global residuals is run (for example, it is easy to see that a residual whose computation form needs measurements generated by sensors from different subsystems will not be run at all). It is of course possible to consider an even more reduced scheme with $\mathcal{C}_k \subseteq \mathcal{C}, k = 1, \ldots, s$ (at least one inclusion being strict) under the local information pattern.

Distributed diagnosers. The collection $\mathcal{C} = \{\mathcal{C}_k = \boldsymbol{\varphi_k} \cup \boldsymbol{\gamma_k}, k = 1, \ldots, s\}$ where $\boldsymbol{\varphi_k} \subseteq \boldsymbol{f}$ and $\boldsymbol{\gamma_k} \subseteq \boldsymbol{g}$ is the most general one. Associated with the information pattern $\mathcal{Z} = \left\{\left(z_k^a, z_k^s\right), k = 1, \ldots, s\right\}$, each subsystem Σ_k sees the global state $\boldsymbol{x}$ as $\left(\boldsymbol{\xi_k}, \overline{\boldsymbol{\xi_k}}\right)$, the global control $\boldsymbol{u}$ as $\left(z_k^a, \overline{z}_k^a\right)$ and the global measurements $\boldsymbol{y}$ as $\left(z_k^s, \overline{z}_k^s\right)$:

$$\begin{pmatrix} \dot{\xi}_k \\ \dot{\overline{\xi}}_k \end{pmatrix} = \begin{pmatrix} \boldsymbol{\varphi_k}\left(\boldsymbol{\xi_k}, \overline{\boldsymbol{\xi}_k}, z_k^a, \overline{z}_k^a, \boldsymbol{d}, t\right) \\ \overline{\boldsymbol{\varphi_k}}\left(\boldsymbol{\xi_k}, \overline{\boldsymbol{\xi}_k}, z_k^a, \overline{z}_k^a, \boldsymbol{d}, t\right) \end{pmatrix} \tag{10.13}$$

$$\begin{pmatrix} z_k^s \\ \overline{z}_k^s \end{pmatrix} = \begin{pmatrix} \gamma_k\left(\boldsymbol{x}, z_k^a, \overline{z}_k^a, \boldsymbol{d}, t\right) \\ \overline{\gamma_k}\left(\boldsymbol{x}, z_k^a, \overline{z}_k^a, \boldsymbol{d}, t\right) \end{pmatrix} \tag{10.14}$$

which results in the local residuals $\boldsymbol{r}_k\left(\boldsymbol{\varphi_k} \cup \boldsymbol{\gamma_k}, z_k^a \cup z_k^s\right)$.

Remark 10.5 The diagnosis decomposition needs by no means be identical to the control decomposition (10.3) and (10.4). Taking $\boldsymbol{\varphi_k} = \boldsymbol{f}_k, k = 1, \ldots, s$ is sometimes justified in the literature by the argument that under the decentralised information pattern, the local residuals $\boldsymbol{r}_k\left(\boldsymbol{\varphi_k} \cup \boldsymbol{g}_k, \boldsymbol{u}_k \cup \boldsymbol{y}_k\right)$ are sensitive only to faults in Σ_k, but this is true only if an over-constrained subsystem exists in the decomposition of Σ_k. In general, both the interconnection variables and the consideration

of wider information patterns introduce constraints from other subsystems whose elimination (when possible) results in residual structures that do not contain only local constraints.

Remark 10.6 z_k^a and z_k^s being defined by the given information pattern, the largest set of local residuals is obtained with $\mathcal{C}_k = \boldsymbol{f} \cup \gamma_k$, $k = 1, \ldots, s$, where γ_k is determined by z_k^s. This is nothing but the subset of global residuals whose computation form is available to Σ_k. □

Replicas. Local diagnoser residuals may have non-empty intersections. For example, two subsystems that share data both see a common subset of constraints by means of the same known variables, which results in identical local residuals. The same conclusion holds when two subsystems publish their data, and each of them subscribes to the data published by the other one. It is a design decision to implement several replicas of the same residuals in several local diagnosers. The decision has a cost associated with multiple calculations of the same residuals, but it allows to detect faults that might occur in the computing devices, using a voting scheme. Moreover, the diagnosis remains available under such faults, if the number of replicas is large enough. Note that the local to global coordination rules remain unchanged when replicas are used, as it can be checked from the following example.

Example 10.3 (**cont.**) **Ship with dual measurements**
Assume that Σ_1 runs ρ_1, ρ_2 and Σ_2 runs ρ_2, ρ_3. The local distinguishability partitions become

	$\mathcal{D}^0$	$\mathcal{D}^1$	$\mathcal{D}^2$	$\mathcal{D}^3$	$\mathcal{D}^4$
ρ_1	0	1	1	0	0
ρ_2	0	1	0	1	0

and

	$\mathcal{D}^0$	$\mathcal{D}^1$	$\mathcal{D}^2$	$\mathcal{D}^3$	$\mathcal{D}^4$
ρ_2	0	1	0	1	0
ρ_3	0	0	0	1	1

and they still provide the coordinated diagnosis:

$\rho_1\rho_2\rho_3$	Δ_1	Δ_2	Δ_{12}
000	$\mathcal{D}^0 \cup \mathcal{D}^4$	$\mathcal{D}^0 \cup \mathcal{D}^2$	$\mathcal{D}^0$
001	$\mathcal{D}^0 \cup \mathcal{D}^4$	$\mathcal{D}^4$	$\mathcal{D}^4$
010	$\mathcal{D}^3$	$\mathcal{D}^1$	cannot happen
011	$\mathcal{D}^3$	$\mathcal{D}^3 \cup \mathcal{D}^4 \times \mathcal{D}^1$	$\mathcal{D}^3 \cup \mathcal{D}^4$
100	$\mathcal{D}^2$	$\mathcal{D}^0 \cup \mathcal{D}^2$	$\mathcal{D}^2$
101	$\mathcal{D}^2$	$\mathcal{D}^4$	$\mathcal{D}^2 \times \mathcal{D}^4$
110	$\mathcal{D}^1 \cup \mathcal{D}^2 \times \mathcal{D}^3$	$\mathcal{D}^1$	$\mathcal{D}^1 \cup \mathcal{D}^2$
111	$\mathcal{D}^1 \cup \mathcal{D}^2 \times \mathcal{D}^3$	$\mathcal{D}^3 \cup \mathcal{D}^4 \times \mathcal{D}^1$	$\mathcal{D}^1 \times \left(\mathcal{D}^3 \cup \mathcal{D}^4\right) \cup \mathcal{D}^2 \times \mathcal{D}^3$

10.4 Design of the Local Diagnosers

10.4.1 Specifications

The design of a distributed diagnosis scheme aims at satisfying functional and fault-tolerance specifications, under local computing capacity constraints, at a minimal communication cost.

Functional specifications. The functional specifications encompass the following points:

- The detectability and isolability performances of the global system $(\mathcal{C}, \mathcal{K})$ are entirely defined by the set of residuals $\boldsymbol{r}(\mathcal{C}, \mathcal{K})$. In what follows, it is supposed that the diagnosis performances of the distributed system are wished to be the same as those of the centralised system. However, the approach can be applied whatever the subset of residuals $\boldsymbol{r} \subseteq \boldsymbol{r}(\mathcal{C}, \mathcal{K})$ that are wished to be implemented.
- The computing cost of a subsystem Σ_k which has been assigned the set of residuals $\boldsymbol{r}_k$ is a function $h(\boldsymbol{r}_k)$ assumed to be known. The capacity constraint is expressed as $h(\boldsymbol{r}_k) \leq h_k, k = 1, \ldots, s$.
- The communication cost depends on the information pattern that is implemented, and it is an increasing function of the set of communicated variables. For the sake of simplicity, information patterns are considered first under the publisher/subscriber scheme. The extension to the bilateral agreement scheme is considered next.

Fault-tolerance specifications. Fault-tolerance specifications may be added to the functional specifications. They specify the diagnosis performances that are still to be achieved should faults occur in

- the sensors or in the communication system (they decrease the set of known inputs that can be used by each local diagnoser),
- the process components (they decrease the set of healthy constraints upon which the set of residuals to be used depends), and
- the local computing devices (the local diagnosis from faulty devices cannot be used in the coordination procedure).

10.4.2 Simple Distribution Problem

Let us start with the following simple problem associated with the functional specifications: assuming there is no capacity constraint associated with the subsystems Σ_k, $(k = 1, \ldots, s)$ distribute the residual computations among them so as to obtain the same diagnosis performances as in the centralised scheme, at a minimal communication cost.

Let $\boldsymbol{r}(\mathcal{C}, \mathcal{K})$ be the residuals of the global system, and $\boldsymbol{r}_k(\mathcal{C}, \mathcal{Z})$ be the residuals whose computation form is available to subsystem Σ_k under the information pattern $\mathcal{Z}$. Then, the distributed scheme achieves the same performances as the centralised scheme if and only if

$$\cup_{k=1,\ldots,s} \boldsymbol{r}_k(\mathcal{C}, \mathcal{Z}) = \boldsymbol{r}(\mathcal{C}, \mathcal{K}) \tag{10.15}$$

i.e. the residuals $\boldsymbol{r}_k(\mathcal{C}, \mathcal{Z}), k = 1, \ldots, s$ cover the residuals $\boldsymbol{r}(\mathcal{C}, K)$.

From the monotonicity property $\mathcal{Z}^+ \succeq \mathcal{Z} \Rightarrow \boldsymbol{r}_k\left(\mathcal{C}, \mathcal{Z}^+\right) \supseteq \boldsymbol{r}_k(\mathcal{C}, \mathcal{Z})$, it follows that if Eq. (10.15) is not satisfied under an information pattern $\mathcal{Z}$ it may be satisfied under a wider one $\mathcal{Z}^+$. In the publisher/subscriber scheme, wider information patterns are obtained by publishing more variables. The set of all possible information patterns is therefore the lattice of all publishable variables, namely $L = 2^{\boldsymbol{u} \cup \boldsymbol{y}}$, which is organised into levels L_i that contain subsets of i variables. The algorithm that solves the simple distribution problem is therefore

Algorithm 10.1 *Simple distribution*

Given: a set $\boldsymbol{r}(\mathcal{C}, \mathcal{K})$ of residuals to be covered
a system decomposition into subsystems Σ_k with local known variables $\mathcal{K}_k = \boldsymbol{u}_k \cup \boldsymbol{y}_k$

Initialisation: $E_i = L_i, \quad i = 0, 1, \ldots |\boldsymbol{u} \cup \boldsymbol{y}|$.

Loop: While $E_t \neq \emptyset$

1. for each subset of published variables $\boldsymbol{z} \in E_t$, identify the subsets of residuals $\boldsymbol{r}_k(\mathcal{C}, \mathcal{Z}) \subseteq \boldsymbol{r}(\mathcal{C}, \mathcal{K})$, $(k = 1, \ldots, s)$ whose computation form is available, and update E_t as $E_t \setminus \{\boldsymbol{z}\}$
2. If Eq. (10.15) is satisfied, $\boldsymbol{z}$ solves the problem. List $\boldsymbol{z}$ in the set of solutions Z^* and update E_{t+1} as $E_{t+1} \setminus \mathcal{P}(Z^*)$ where $\mathcal{P}(Z^*) = \cup_{z \in Z^*} \mathcal{P}(\boldsymbol{z})$ and $\mathcal{P}(\boldsymbol{z})$ are the predecessors of $\boldsymbol{z}$ in the lattice L.

Result: List Z^* of minimal subsets of variables to be published in the publisher/subscriber scheme in order for the distributed diagnosis to achieve the same performance as the centralised diagnosis.

Comments.

1. Since any solution $\boldsymbol{z} \in Z^*$ results in the running of all the residuals $\boldsymbol{r}(\mathcal{C}, \mathcal{K})$, any predecessor of $\boldsymbol{z}$ also results in running all the residuals. Because there are more available data to the subsystems, some residuals may be replicated in several subsystems.

2. The process considers wider and wider information patterns, so it must eventually terminate, with a non-empty set of solutions. Indeed, the worst case in which all the publishable variables are published is associated with the information pattern $\mathcal{Z}_{\max}$ that implements the whole set of residuals $\boldsymbol{r}(\mathcal{C}, \mathcal{K})$ in each subsystem.
3. The presentation has been aimed at distributing all the residuals in $\boldsymbol{r}(\mathcal{C}, \mathcal{K})$ but the approach can be applied whatever the subset of residuals $\boldsymbol{r} \subseteq \boldsymbol{r}(\mathcal{C}, \mathcal{K})$ that are wished to be implemented.

Example 10.3 (cont.) Ship with dual measurements
Let us illustrate the simple distribution procedure with the ship example whose results have been presented earlier. Let the two subsystems be Σ_1 with local sensors $\{y_1, y_2\}$ and Σ_2 with local sensors $\{y_3, y_4\}$. The specification is that the distributed diagnosis should be as powerful as the centralised diagnosis.

The procedure starts with the minimal information pattern

$$\mathcal{Z}_{\min} = \{(y_1, y_2), (y_3, y_4)\}$$

associated with the decentralised system.

The first iteration provides the distribution ρ_1 assigned to Σ_1 and ρ_3 assigned to Σ_2 as already seen, which is not admissible because there is a loss of detectability and isolability with respect to the centralised scheme. In the second iteration, the constraints are considered under wider information patterns. Under the publisher/subscriber scheme, four wider information patterns can be obtained by publishing one single variable, namely

$$\begin{aligned}
\mathcal{Z}_1 &= \{(y_1, y_2), (y_1, y_3, y_4)\} \\
\mathcal{Z}_2 &= \{(y_1, y_2), (y_2, y_3, y_4)\} \\
\mathcal{Z}_3 &= \{(y_1, y_2, y_3), (y_3, y_4)\} \\
\mathcal{Z}_4 &= \{(y_1, y_2, y_4), (y_3, y_4)\}\,.
\end{aligned}$$

From the residuals (10.8), the associated decompositions are

Information pattern	Σ_1	Σ_2
$\mathcal{Z}_1$	ρ_1	ρ_2, ρ_3
$\mathcal{Z}_2$	ρ_1	ρ_3
$\mathcal{Z}_3$	ρ_1, ρ_2	ρ_3
$\mathcal{Z}_4$	ρ_1	ρ_3

Only $\mathcal{Z}_1$ and $\mathcal{Z}_3$ improve the diagnosis capability. The diagnosis performances under $\mathcal{Z}_3$ are given in Tables 10.4 and 10.5. They show that there is no loss of detectability/isolability with respect to the global diagnosis scheme. It can be checked from the local tables

	OK	m_1	m_2	m_3	m_4	c_1	c_2
ρ_1	0	1	1	0	0	0	0

and

	OK	m_1	m_2	m_3	m_4	c_1	c_2
ρ_2	0	1	0	1	0	0	0
ρ_3	0	0	0	1	1	1	1

that the same conclusion holds under $\mathcal{Z}_1$, so the two schemes satisfy the diagnosis specifications.

Finally, the information patterns $\mathcal{Z}_1$ and $\mathcal{Z}_3$ are the minimal ones for which the diagnosis specifications can be obtained. Considering wider information patterns is not necessary (unless replications are wished), since the diagnosis performances would not be increased, but the local computing costs could only be increased (because more residuals would be computed in each subsystem). In order to illustrate this point, let us investigate the case where two variables are published. There are six possible information patterns, which lead to six possible distributed schemes, according to the table:

Information pattern	Σ_1	Σ_2
$\mathcal{Z}_{12} = \{(y_1, y_2), (y_1, y_2, y_3, y_4)\}$	ρ_1	ρ_1, ρ_2, ρ_3
$\mathcal{Z}_{13} = \{(y_1, y_2, y_3), (y_1, y_3, y_4)\}$	ρ_1, ρ_2	ρ_2, ρ_3
$\mathcal{Z}_{14} = \{(y_1, y_2, y_4), (y_1, y_3, y_4)\}$	ρ_1	ρ_2, ρ_3
$\mathcal{Z}_{23} = \{(y_1, y_2, y_3), (y_2, y_3, y_4)\}$	ρ_1, ρ_2	ρ_3
$\mathcal{Z}_{24} = \{(y_1, y_2, y_4), (y_2, y_3, y_4)\}$	ρ_1	ρ_3
$\mathcal{Z}_{34} = \{(y_1, y_2, y_3, y_4), (y_3, y_4)\}$	ρ_1, ρ_2	ρ_3

All schemes (except $\mathcal{Z}_{24}$) satisfy the diagnosis specifications since they allow to distribute the computation of all residuals. Note that schemes $\mathcal{Z}_{12}$ and $\mathcal{Z}_{13}$ implement residuals replications. Note also that any information pattern wider than the two minimal patterns $\mathcal{Z}_1$ and $\mathcal{Z}_3$ under which the specifications are satisfied also satisfies the specifications, as shown in Fig. 10.6, where the different information patterns are displayed along with the associated residual distribution. Information patterns under which the distributed diagnosis specifications are satisfied are in white, and the minimal ones have a bold contour. □

10.4.3 Distribution Under Computing Cost Constraints

The simple distribution problem does not take into account the possible limitations in the local computing power of the different subsystems. Assume there is a known function $h_k(\rho)$ associated with each pair (ρ, k), $\rho \in \boldsymbol{r}(\mathcal{C}, \mathcal{K})$, $(k = 1, \ldots, s)$ that evaluates the computing cost of a residual ρ by the computing device of subsystem Σ_k and that subsystem Σ_k can devote only an amount h_k of computing effort to the distributed diagnosis task. Then, assuming that computing costs are additive, the previous distribution problem must take into account the computing cost constraints:

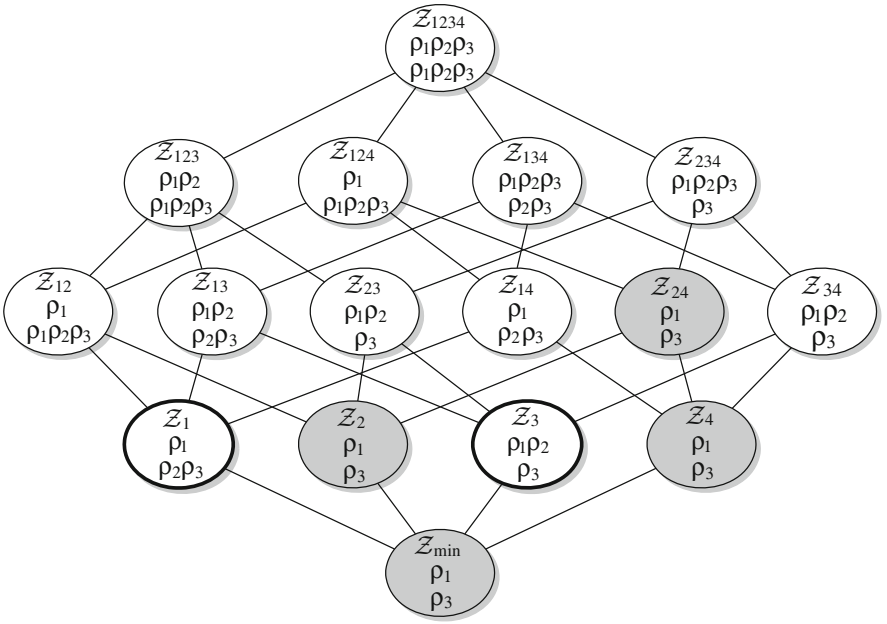


Fig. 10.6 Information patterns and diagnosis distribution in the ship example

$$\sum_{\rho \in \boldsymbol{r}_k} h_k(\rho) \leq h_k. \tag{10.16}$$

Starting with the results of the simple distribution algorithm, it is easily seen that if the set Z^* contains at least one solution that satisfies the computing cost constraints, then the constrained problem is solved, by discarding those solutions that are inadmissible.

Two situations must be distinguished when all solutions in Z^* are inadmissible:

- First, an inadmissible solution can be transformed into an admissible one, if there exists subsets of residuals in the overloaded subsystems whose deletion leads to an admissible computing cost, but does not degrade the diagnosis performance because they are replicas of residuals computed in other—non-overloaded—subsystems.
- If no such possibility exists, non-minimal subsets of publishable data must be considered, in order to provide non-overloaded subsystem with replicas of residuals that could be deleted from overloaded subsystems.

In order to implement this procedure, the previous algorithm is modified as follows.

Algorithm 10.2 *Distribution under cost constraints*

Given: A set $\boldsymbol{r}(\mathcal{C}, \mathcal{K})$ of residuals to be covered
a system decomposition into subsystems Σ_k with local known variables $\mathcal{K}_k = \boldsymbol{u}_k \cup \boldsymbol{y}_k$, known computing costs $h_k(\rho)$, $\rho \in \boldsymbol{r}(\mathcal{C}, \mathcal{K})$, and known computing power limitations h_k

Initialisation: $E_i = L_i, \quad i = 0, 1, \ldots |\boldsymbol{u} \cup \boldsymbol{y}|$.

Loop: While $E_t \neq \emptyset$:

1. For each subset of published variables $z \in E_t$, identify the subsets of residuals $\boldsymbol{r}_k(\mathcal{C}, \mathcal{Z}) \subseteq \boldsymbol{r}(\mathcal{C}, \mathcal{K})$, $(k = 1, \ldots, s)$ whose computation form is available, and update E_t as $E_t \setminus \{z\}$,
2. If Eqs. (10.15) and (10.16) are satisfied, or if Eq. (10.15) is satisfied and Eq. (10.16) is not satisfied but becomes satisfied by deleting the replicated residuals in the overloaded subsystems, z solves the problem. List z in the set of solutions Z^* and update E_{t+1} as $E_{t+1} \setminus \mathcal{P}(Z^*)$ where $\mathcal{P}(Z^*) = \cup_{z \in Z^*} \mathcal{P}(z)$ and $\mathcal{P}(z)$ are the predecessors of z in the lattice L.

Result: List Z^* of minimal subsets of variables to be published in the publisher/subscriber scheme in order for the distributed diagnosis to achieve the same performance as the centralised diagnosis while satisfying the computation cost constraints.

Comment. Since it explores wider and wider information patterns, the algorithm must eventually terminate. However, a solution is not guaranteed to exist. A necessary and sufficient condition for a solution to exist is that it exists under the maximal information pattern. In that case, all subsystems are able to run all residuals $\boldsymbol{r}(\mathcal{C}, \mathcal{K})$, and the deletion of replicated residuals problem boils down to finding a partition of the set $\boldsymbol{r}(\mathcal{C}, \mathcal{K})$ into s classes such that the computing cost constraints are satisfied. Let $\sigma_k(\rho)$ be the binary variables such that $\sigma_k(\rho) = 1$ when residual ρ is assigned to subsystem Σ_k and $\sigma_k(\rho) = 0$ when residual ρ is not assigned to subsystem Σ_k. Then the residual distribution problem under computation cost constraints has a solution if and only if the constraint satisfaction problem,

$$\forall \rho \in \boldsymbol{r}(\mathcal{C}, \mathcal{K}) : \sum_{k=1,\ldots,s} \sigma_k(\rho) = 1 \tag{10.17}$$

$$k = 1, \ldots, s : \sum_{\rho \in \boldsymbol{r}(\mathcal{C}, \mathcal{K})} \sigma_k(\rho)\, h_k(\rho) \leq h_k, \tag{10.18}$$

has a solution, which can easily be checked since it is a classical task allocation

problem (algorithms to solve a version of this problem—namely finding maximal matchings in a bipartite graph—were given in Chap. 5).

10.4.4 The Bilateral Agreements Scheme

In the bilateral agreements scheme, the set of all subsystems is partitioned into equivalence classes such that subsystems in the same class share all their data. Denoting by $\mathcal{K}_{\mathrm{K}}$ the known data available to all subsystems in a class $\{\Sigma_k, k \in K\}$, these bilateral agreements result in the residual assignments $\boldsymbol{r}_k\,(\mathcal{C}, \mathcal{K}_{\mathrm{K}}) = \boldsymbol{r}_{\mathrm{K}}, k \in K$. It follows that for each residual possibly run by subsystem Σ_k, there are $|K|-1$ replicas possibly run by the other subsystems Σ_j, $(j \neq k)$ in the same class.

The following algorithm explores the increasing levels of a hierarchy built on the atomic decomposition Σ_k, $(k = 1, \ldots, s)$. At each level of the hierarchy, two subsystems are merged according to some merging policy, for example, the two subsystems whose merger implies the smallest communication cost, the two subsystems whose merger implies the largest set of computable residuals, the two subsystems with the best efficiency ratio computed from the increase in the communication cost versus the increase in the number of computable residual, etc. The satisfaction of the computing cost constraints is achieved by deleting from the overloaded subsystems those residuals whose replica is present in some underloaded subsystem.

Algorithm 10.3 *Bilateral agreements*

Given: A set $\boldsymbol{r}\,(\mathcal{C}, \mathcal{K})$ of residuals to be covered
a system decomposition into subsystems Σ_k with local known variables $\mathcal{K}_k = \boldsymbol{u}_k \cup \boldsymbol{y}_k$, known computing costs $h_k\,(\rho)$, $\rho \in \boldsymbol{r}\,(\mathcal{C}, \mathcal{K})$, and known computing power limitations h_k.

Initialisation: $E_0 = \left\{E_{0,k} = \Sigma_k, \quad k = 1, \ldots, s\right\}$

Loop: While E_t is not a singleton

1. For each pair of classes $E_{t,i}$ and $E_{t,j}$, evaluate their possible merger in terms of the induced communication cost and of the residuals that become computable, and select the pair whose merger is preferred according to the selected merging policy. Update E_{t+1} by replacing $\left\{E_{t,i}, E_{t,j}\right\}$ in E_t by $E_{t,i} \cup E_{t,j}$.
2. If all residuals are covered and the computing cost constraints are satisfied, or if they become satisfied by deleting the replicated residuals in the overloaded subsystems, the decomposition E_{t+1} solves the problem.

Result: a decomposition of the system into equivalence classes associated with bilateral communication agreements that achieves the same performance as the centralised diagnosis while satisfying the computation cost constraints.

Comments.

1. Since it explores increasing levels of the system hierarchy, the algorithm must eventually terminate. The necessary and sufficient condition for a solution to exist (and therefore to be found) is the same as in the publisher/subscriber scheme, that has been given in Eqs. (10.17) and (10.18).
2. It is well known that hierarchical procedures are by no way optimal, since mergers are performed at each level following a greedy approach (the best merger at a given level is not necessarily the best one from a global point of view), and are never un-merged. However, they are very popular because of their simplicity. They are in general run several times, under several merging policies, which allows for comparison between the results and get a good idea of the main features of the solutions.

Example 10.4 Hierarchical distribution algorithm
In this example, we consider the distribution of a set of nine residuals among four subsystems, under the bilateral communication scheme. The structures of the residuals computation form are

	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	z_9	z_{10}	z_{11}
ρ_1						1		1			
ρ_2		1	1		1						
ρ_3						1	1				
ρ_4		1		1							
ρ_5								1		1	1
ρ_6	1	1									
ρ_7							1	1	1		
ρ_8						1	1			1	
ρ_9	1	1	1	1							

and the local data are given by

Subsystem	Σ_1	Σ_2	Σ_3	Σ_4
Local data	z_1, z_2	z_3, z_4, z_5	z_6, z_7	z_8, z_9, z_{10}, z_{11}

The atomic decomposition associated with the decentralised scheme leads to the computable residuals

E_0	Σ_1	Σ_2	Σ_3	Σ_4
Computable residuals	ρ_6	$\emptyset$	ρ_3	ρ_5

that do not cover the whole set $\rho_1, \ldots, \rho_9$, and therefore the information exchange must be increased. At the first level of the hierarchy, there are six possible bilateral agreements, which give the following results:

Mergers	$\Sigma_1 \cup \Sigma_2$	$\Sigma_1 \cup \Sigma_3$	$\Sigma_1 \cup \Sigma_4$
Computable residuals	$\rho_2\rho_4\rho_6\rho_9$	$\rho_3\rho_6$	$\rho_5\rho_6$
Communication load	$z_1z_2z_3z_4z_5$	$z_1z_2z_6z_7$	$z_1z_2z_8z_9z_{10}z_{11}$

Mergers	$\Sigma_2 \cup \Sigma_3$	$\Sigma_2 \cup \Sigma_4$	$\Sigma_3 \cup \Sigma_4$
Computable residuals	$\rho_3\rho_8$	ρ_5	$\rho_1\rho_3\rho_5\rho_7\rho_8$
Communication load	$z_3z_4z_5z_6z_7$	$z_3z_4z_5z_8z_9z_{10}z_{11}$	$z_6z_7z_8z_9z_{10}z_{11}$

Assume that the merging policy that gives the largest number of computable residuals is chosen. There are two decompositions at level 1 that both allow to compute six residuals, namely

E_{11}	Σ_1	Σ_2	$\Sigma_3 \cup \Sigma_4$
Computable residuals	ρ_6	$\emptyset$	$\rho_1\rho_3\rho_5\rho_7\rho_8$

and

E_{12}	$\Sigma_1 \cup \Sigma_2$	Σ_3	Σ_4
Computable residuals	$\rho_2\rho_4\rho_6\rho_9$	ρ_3	ρ_5

Note that E_{12} needs less communication than E_{11}, but neither E_{11} nor E_{12} covers the whole set of wished residuals, so more communication has to be introduced by considering the second level of the hierarchy.

From aggregating E_{11} and E_{12}, one gets

E_{21}	$\Sigma_1 \cup \Sigma_2$	$\Sigma_3 \cup \Sigma_4$
Computable residuals	$\rho_2\rho_4\rho_6\rho_9$	$\rho_1\rho_3\rho_5\rho_7\rho_8$

that covers all the residuals, and from E_{12} one gets three possibilities, associated with the mergers $\Sigma_3 \cup \Sigma_4$, $\Sigma_1 \cup \Sigma_2 \cup \Sigma_3$ and $\Sigma_1 \cup \Sigma_2 \cup \Sigma_4$ but none of them covers the whole set of residuals. Note that in the solution E_{21}, both Σ_1 and Σ_2 are able to run the residuals $\rho_2\rho_4\rho_6\rho_9$ and both Σ_3 and Σ_4 are able to run the residuals $\rho_1\rho_3\rho_5\rho_7\rho_8$. Assuming each subsystem has a sufficient computing power, the existence of the replicas makes the distributed scheme tolerant to faults in the individual computing devices: for example, in the presence of a complete failure of Σ_1, the residuals $\rho_2\rho_4\rho_6\rho_9$ could still be run by Σ_2 (of course, one should also consider in this case the effects of such a failure on the control functionalities, but this is not the topic of this section).

Assuming limited computing powers that do not allow full duplication, the set of residuals $\rho_2\rho_4\rho_6\rho_9$ should be split into two subsets, respectively, run in Σ_1 and Σ_2, according to the classical task allocation problem under constraints, a version of which has been used in the comment on p. 499 to evaluate the existence of a solution to the distribution problem under constraints. □

10.4.5 Fault-Tolerant Distributed Diagnosis

Fault-tolerant distributed diagnosis considers the effect of faults on the diagnosis capability of a distributed diagnosis system. It will not be developed in detail, since most of the tools that are useful for the analysis and the design of fault-tolerant diagnosis have been presented in this chapter and in previous chapters, as it appears from the following analysis of the different fault consequences.

Faults in the process components. Faults in the process components decrease the set of constraints that can be used to build the residuals. Less constraints means less residuals, which means less detectability and distinguishability. The analysis of the fault tolerance of a given diagnosis system is therefore nothing but the analysis of the subsets of residuals that still allow to perform the desired detection and isolation specifications. Considering subsets of residuals (i.e. residual configurations) is quite similar to considering actuator or sensor configurations as in Chap. 8.

Faults in the sensors or in the communication network. Faults in the sensors or in the communication network decrease the set of known variables that are available to the local computing devices of the distributed system. Less known variables means less residuals, which brings back to the above problem.

Faults in the local computing devices. Faults in the local computing devices result in erroneous local diagnosis. Using replicas of the same residuals in different computing devices allows to detect inconsistencies by means of appropriate voting schemes. The general problem has been thoroughly studied in the computer science community, and the reader is referred to the bibliographical notes for an overview of the main results.

10.5 Fault-Tolerant Control by Information Pattern Reconfiguration

The previous sections have shown the prominent role of the information pattern in the design of a distributed diagnosis algorithm. Similarly, the role of the information pattern is the main feature that distinguishes fault-tolerant control in distributed systems from fault-tolerant control in embedded systems. Other features are that

- due to the interactions between subsystems, the specifications to be satisfied in normal and in faulty operations must be considered at the system level, while the control is designed at the subsystem level; and
- the fault recovery process is desired to be limited to the smallest possible number of subsystems.

These features are now addressed in the frame of reconfiguration-based fault tolerance.

10.5.1 Admissibility and Reconfigurability

Remember that fault tolerance is the property that some specification $\mathcal{P}$ satisfied by the nominal system is also satisfied in the presence of faults (performance degradation may be allowed by introducing a less-demanding specification when faults occur). When distributed systems are considered, an important question to decide is whether each subsystem is responsible for finding an admissible control that achieves the part of the global specification it has been assigned to fulfil (in this case, the specification is said to be *decomposable*), or whether system-level admissibility is to be considered (this is the *non-decomposable* specifications case).

Definition 10.3 (*Decomposable specification*) A specification $\mathcal{P}$ is *decomposable* if it is equivalent to some set $\{\mathcal{P}_k,\ k = 1, \ldots, s\}$ where $\mathcal{P}_k$ is a specification of subsystem Σ_k.

Due to the coupling variables $\overline{\boldsymbol{x}}_k$ in Eq. (10.3), it appears that not all specifications are decomposable. In the sequel of this chapter, we consider non-decomposable specifications. Indeed, there is no interest, when addressing fault-tolerant distributed systems, in considering decomposable specifications: should the specification be decomposable, one could simply address fault tolerance within each subsystem, using the local subsystem model, and treating the interconnection variables as unknown inputs. This would of course result in a distributed recovery algorithm (each subsystem recovers its own faults, independently of the others), but would not bring much new insight to the fault-tolerance problem of the distributed system, since each subsystem would be treated as a system of its own, using the methods presented in the previous chapters.

Example 10.5 Decomposable specifications

- The controllability of the system in Example 10.1 is a structural property that depends on the pair $(\boldsymbol{A}, \boldsymbol{B})$ given in Eq. (10.5). It is in general not equivalent to the controllability of the four subsystems associated with the respective pairs of matrices:

Subsystem	Matrix $\boldsymbol{A}$	Matrix $\boldsymbol{B}$
Σ_1	$\begin{pmatrix} -1 & 2 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & 0.5 \end{pmatrix}$
Σ_2	-2	1
Σ_3	$\begin{pmatrix} -2 & 0.4 \\ 1 & -3 \end{pmatrix}$	$\begin{pmatrix} 2 \\ 1 \end{pmatrix}$
Σ_4	-4	2

- Given an output feedback $\boldsymbol{u} = \boldsymbol{K}\boldsymbol{y}$ and a positive value of α, the system Σ is α-stable if there exists a Lyapunov function $V = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}$ such that $\dot{V} \leq -\alpha V$ along its trajectories, i.e.

$$\boldsymbol{Q}\,(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}\boldsymbol{C}) + (\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}\boldsymbol{C})^{\mathrm{T}}\,\boldsymbol{Q} + \alpha\boldsymbol{Q} \leq 0.$$

  This is a non-structural property that depends on the control law (via the output feedback matrix $\boldsymbol{K}$), and again it is not equivalent to the α-stability of each subsystem $\Sigma_k, k =$

1, ..., 4 because the subsystems are coupled. As a matter of fact, it is well known that the interconnection of several stable subsystems might well result in an unstable system, as illustrated by the very simple example

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -1 & \theta_{12} \\ \theta_{21} & -2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

which is unstable for any values of the interconnection parameters such that $\theta_{12}\theta_{21} > 2$ although the self-dynamics of the two subsystems, respectively, $\dot{x}_1 = -x_1$ and $\dot{x}_2 = -2x_2$ are stable.

- Examples of degraded specifications would be to accept stabilisability instead of controllability, or a smaller decay rate in the α-stability specification. □

Recoverable faults. Let us first consider actuator faults under the reconfiguration strategy (sensor or system component faults are treated the same way). Let $I_N \subseteq I$ be an actuator configuration, i.e. the actuators that are available to achieve specification $\mathcal{P}$, if possible, when actuators in $I_F = I \backslash I_N$ are faulty and have been switched-off. The set of all possible configurations (including the nominal one) is 2^I the power set of I, i.e. the set of all its subsets. Remember that 2^I is a lattice, a mathematical structure whose properties have already been used in previous chapters to address implementation issues and evaluation measures. We will now use the lattice tool for distributed control systems, by considering the set of all possible information patterns, and analysing specific monotonicity properties of interest for the reconfiguration problem.

Remark that although configuration I_N is decomposed into $\{I_{N,k},\ k = 1, \ldots, s\}$ where $I_{N,k} \subseteq I_k$ is the subset of actuators available in subsystem Σ_k, recoverability must be analysed with respect to the global system, because non-decomposable specifications are considered.

Let $(\boldsymbol{u},\ \mathcal{Z})$ be a pair where $\mathcal{Z}$ is an information pattern and $\boldsymbol{u}$ is a control law under $\mathcal{Z}$. The notation $\mathcal{P}\,(I_N,\ \boldsymbol{u},\ \mathcal{Z})$ means that the pair $(\boldsymbol{u},\ \mathcal{Z})$ achieves the specification $\mathcal{P}$ when applied to the subset of actuators $I_N \subseteq 2^I$.

Definition 10.4 (*Admissibility, admissibility span*) A pair $(\boldsymbol{u},\ \mathcal{Z})$ is admissible for configuration I_N, if it satisfies the specification $\mathcal{P}$, i.e. if $\mathcal{P}\,(I_N,\ \boldsymbol{u},\ \mathcal{Z})$. The admissibility span of a pair $(\boldsymbol{u},\ \mathcal{Z})$ is the set $\mathcal{R}\,(\boldsymbol{u},\ \mathcal{Z})$ of all configurations I_N for which the control law $\boldsymbol{u}$ is admissible:

$$\mathcal{R}\,(\boldsymbol{u},\ \mathcal{Z}) = \left\{ I_N \in 2^I :\ \mathcal{P}\,(I_N,\ \boldsymbol{u},\ \mathcal{Z}) \right\}.$$

Definition 10.5 (*Recoverability, recoverability span*) The fault I_F - equivalently the configuration I_N - is recoverable under the information pattern $\mathcal{Z}$ if there exists a control law $\boldsymbol{u}$ such that the pair $(\boldsymbol{u},\ \mathcal{Z})$ is admissible for configuration I_N. The recoverability span of the information pattern $\mathcal{Z}$ is the set $\mathcal{R}\,(\mathcal{Z})$ of all configurations I_N that are recoverable under $\mathcal{Z}$:

$$\mathcal{R}\,(\mathcal{Z}) = \left\{ I_N \in 2^I :\ \exists\,(\boldsymbol{u},\ \mathcal{Z}) :\ \mathcal{P}\,(I_N,\ \boldsymbol{u},\ \mathcal{Z}) \right\}.$$

Note that recoverability is a structural property, since it depends only on the pair $(I_N,\ \mathcal{Z})$.

Example 10.6 Recoverability span
In Example 10.1, assume that it is desired to α-stabilise the system by decentralised control via output feedback. Under the information pattern $\mathcal{Z}_{\min} = \{y_1,\ y_2,\ y_3,\ y_4\}$, the design problem is to find the parameters k_{ij} such that the control laws

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} k_{11} \\ k_{21} \end{pmatrix} y_1$$
$$u_3 = k_{32} y_2$$
$$u_4 = k_{43} y_3$$
$$u_5 = k_{54} y_4$$

α-stabilise the system. Remember that a system is α-stable if there exists a Lyapunov function $V = \boldsymbol{x}^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{x}$ such that $\dot{V} \leq -\alpha V$ along its trajectories, i.e.

$$\boldsymbol{Q}\,(\boldsymbol{A} + \boldsymbol{BKC}) + (\boldsymbol{A} + \boldsymbol{BKC})^{\mathrm{T}}\,\boldsymbol{Q} + \alpha \boldsymbol{Q} \leq 0.$$

It is easy to verify that under the nominal actuator configuration $I = \{1,\ 2,\ 3,\ 4,\ 5\}$, the specification associated with $\alpha = 1$ is achieved using the control laws:

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 1.2991 \\ 5.6882 \end{pmatrix} y_1$$
$$u_3 = 4.6939 y_2 \qquad (10.19)$$
$$u_4 = 0.1190 y_3$$
$$u_5 = 3.3712 y_4.$$

Using the short notations 123 for configuration $\{1,\ 2,\ 3\}$, 2345 for configuration $\{2,\ 3,\ 4,\ 5\}$, etc., it can be easily checked that in the presence of actuator faults, configurations 2345, 1235, 1234, 1245, 245, 235, 234, 125 and 123 can still be α-stabilised using the local information pattern, but this is true neither for configurations 1345, 345, 145, 135, 134 and 124 nor for their subsets. The white nodes in Fig. 10.7 show the recoverability span associated with the local information pattern $\mathcal{Z}_{\min} = \{y_1,\ y_2,\ y_3,\ y_4\}$. For example, it can be checked that the α-stabilisation problem has a solution for configuration 245 which is

$$u_2 = 15.6231 y_1$$
$$u_4 = 16.0533 y_3$$
$$u_5 = 5.4455 y_4,$$

while it has no solution for the grey configurations.

Note that the algorithmic complexity of the determination of the set of recoverable configurations is limited by the fact that it is enough to find the minimal ones. Indeed, if a configuration I_N is recoverable under an information pattern $\mathcal{Z}$, then any configuration that includes I_N is also recoverable under $\mathcal{Z}$. In this example, there are two minimal recoverable configurations, namely 23 and 25, that are shown with a bold contour on the figure. □

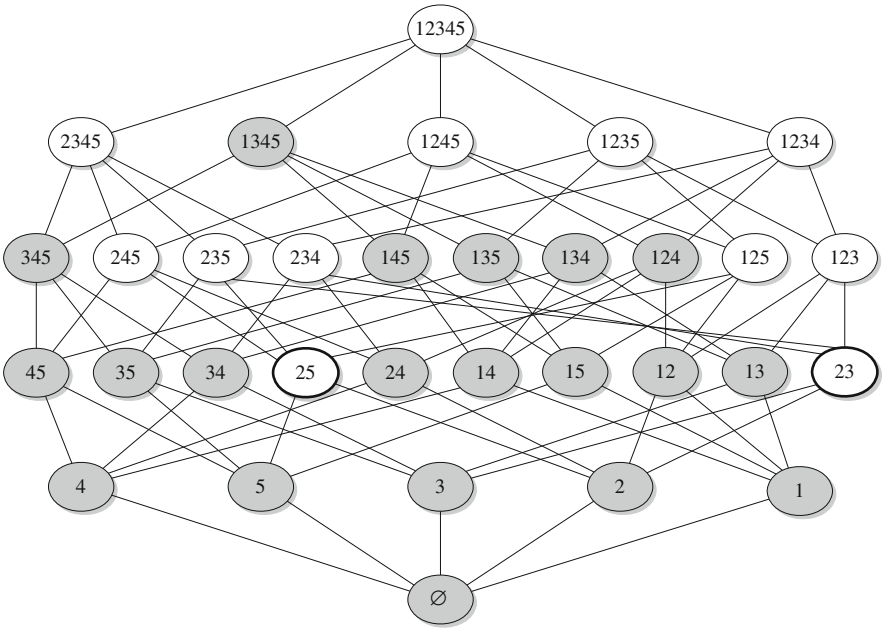


Fig. 10.7 Recoverability span under $\mathcal{Z}_{\min}$

10.5.2 Information Pattern Reconfiguration

As the recoverability of a configuration depends on the information pattern that is used, adapting the information pattern to the system situation is a means to achieve fault tolerance. Let I_C be the current system configuration (either nominal or the result of previous faults), and $\mathcal{Z}_C$ be the current information pattern, such that $I_C \in \mathcal{R}(\mathcal{Z}_C)$, and assume a fault occurs, leading to configuration $I_N \subset I_C$. Then, either $I_N \in \mathcal{R}(\mathcal{Z}_C)$ or $I_N \notin \mathcal{R}(\mathcal{Z}_C)$.

In the first case (Problem 1), one has to find a control law under $\mathcal{Z}_C$ that is admissible for I_N. Such a control law indeed exists, since $I_N \in \mathcal{R}(\mathcal{Z}_C)$. In the second case, no control law under $\mathcal{Z}_C$ can achieve the specification $\mathcal{P}$. A possibility is therefore to relax the information pattern constraint by finding, if possible, an information pattern $\mathcal{Z}_N$ such that $I_N \in \mathcal{R}(\mathcal{Z}_N)$ (Problem 2). Once $\mathcal{Z}_N$ is determined, Problem 1 is solved to find a control law that achieves the specification $\mathcal{P}$ under $\mathcal{Z}_N$. If an information pattern $\mathcal{Z}_N$ such that $I_N \in \mathcal{R}(\mathcal{Z}_N)$ does not exist, the fault is not recoverable at all and the objective reconfiguration level must be triggered.

In the sequel, we focus on the information pattern reconfiguration problem (Problem 2), since Problem 1 is nothing but the classical fault-tolerance problem.

Ordering the set of information patterns. Let $\boldsymbol{Z}$ be the set of information patterns that are considered. Note that while for distributed diagnosis, local diagnosers were provided with local and possibly remote control and measurement signals, and in

distributed control, each local controller has to be provided only with local and possibly remote measurement signals. The information patterns considered here are therefore simpler than in the distributed diagnosis case, being only associated with covers of J. It follows that $\boldsymbol{Z}$ is associated with the set of covers of J, and that the partial order relation on $\boldsymbol{Z}$ is also simplified as follows:

Definition 10.6 (*Order on the set of information patterns*) Let $\mathcal{Z}^+ = \{z_k^+,\ k = 1, \ldots, s\}$ and $\mathcal{Z}^- = \left\{z_k^-,\ k = 1, \ldots, s\right\}$ be two information patterns in $\boldsymbol{Z}$. $\mathcal{Z}^+$ is *wider* than $\mathcal{Z}^-$ ($\mathcal{Z}^+ \succeq \mathcal{Z}^-$) if

$$\forall k \in \{1, \ldots, s\},\ z_k^- \subseteq z_k^+.$$

Remark 10.7 Similar to the distributed diagnosis case, the full information pattern $\mathcal{Z}_{\max}$ is wider than any other, making $\mathcal{Z}_{\max}$ the maximal element of $\boldsymbol{Z}$. Also, there exists no information pattern in $\boldsymbol{Z}$ that is *narrower* than $\mathcal{Z}_{\min}$ (meaning that $\mathcal{Z}_{\min}$ would be wider than it); therefore, $\mathcal{Z}_{\min}$ is the minimal element of $\boldsymbol{Z}$ (to see this, consider any $\mathcal{Z}^* \preceq Z_{\min}$ and conclude that $\mathcal{Z}^* = \mathcal{Z}_{\min}$ if condition $\cup_{k=1,\ldots,s} Z_k = J$ is to be satisfied). □

A monotonicity property. The main result here is that the information pattern reconfiguration problem can be solved only for those configurations that are recoverable under the full information pattern.

Theorem 10.3 *Let* $(I_{\mathrm{C}},\ \mathcal{Z}_{\mathrm{C}})$ *be the current system situation and assume that a fault occurs such that* $I_{\mathrm{N}} \subset I_{\mathrm{C}}$. *A necessary and sufficient condition for the existence of an information pattern* $\mathcal{Z}_{\mathrm{N}}$ *such that* $I_{\mathrm{N}} \in \mathcal{R}\left(\mathcal{Z}_{\mathrm{N}}\right)$ *is that* $I_{\mathrm{N}} \in \mathcal{R}\left(\mathcal{Z}_{\max}\right)$.

This result is easy to understand from the fact that recoverability spans are monotonous with respect to the order $\succeq$ on $\mathcal{Z}$, i.e. one has

$$\mathcal{Z}^+ \succeq \mathcal{Z}^- \Rightarrow \mathcal{R}\left(\mathcal{Z}^-\right) \subseteq \mathcal{R}\left(\mathcal{Z}^+\right). \tag{10.20}$$

Indeed, under the information pattern $\mathcal{Z}^+$, each local controller can use a super-set of the measurement signals available under the pattern $\mathcal{Z}^-$. Therefore, if there exists an admissible control under $\mathcal{Z}^-$, there is one under $\mathcal{Z}^+$. Now, assume $I_{\mathrm{N}} \notin \mathcal{R}\left(\mathcal{Z}_{\max}\right)$, and then from Eq. (10.20), there is no $\mathcal{Z}_{\mathrm{N}} \preceq \mathcal{Z}_{\max}$ such that $I_{\mathrm{N}} \in \mathcal{R}\left(\mathcal{Z}_{\mathrm{N}}\right)$, and the fault is therefore non-recoverable. Assume now $I_{\mathrm{N}} \in \mathcal{R}\left(\mathcal{Z}_{\max}\right)$, and then $\mathcal{Z}_{\mathrm{N}} = \mathcal{Z}_{\max}$ solves the problem.

Note that the result in Theorem 10.3 is true whatever the status $I_{\mathrm{N}} \in \mathcal{R}\left(\mathcal{Z}_{\mathrm{C}}\right)$ or $I_{\mathrm{N}} \notin \mathcal{R}\left(\mathcal{Z}_{\mathrm{C}}\right)$. If $I_{\mathrm{N}} \in \mathcal{R}\left(\mathcal{Z}_{\mathrm{C}}\right)$, one indeed has $I_{\mathrm{N}} \in \mathcal{R}\left(\mathcal{Z}_{\max}\right)$ but there is no need to reconfigure the information pattern $\mathcal{Z}_{\mathrm{C}}$ (note that this does not mean that the control laws should not be reconfigured, but only that the data they use do not need to be changed!). If $I_{\mathrm{N}} \notin \mathcal{R}\left(\mathcal{Z}_{\mathrm{C}}\right)$ and $I_{\mathrm{N}} \notin \mathcal{R}\left(\mathcal{Z}_{\max}\right)$, the fault is not recoverable, and objective reconfiguration has to take place. We now consider the case $I_{\mathrm{N}} \notin \mathcal{R}\left(\mathcal{Z}_{\mathrm{C}}\right)$ (the fault is not recoverable under the current information pattern) but $I_{\mathrm{N}} \in \mathcal{R}\left(\mathcal{Z}_{\max}\right)$ (the fault is recoverable under the full information pattern).

Application to fault-tolerant distributed control. Let $(I_C,\ \mathcal{Z}_C)$ be the current system situation and consider a fault that is not recoverable under the current information pattern, but is recoverable under the full information pattern. Solving the information pattern reconfiguration problem is equivalent to determining the set:

$$\mathcal{Z}(I_N) = \{\mathcal{Z} \in \boldsymbol{Z} :\ I_N \in \mathcal{R}(\mathcal{Z})\}.$$

From an algorithmic point of view, testing every $\mathcal{Z} \in \boldsymbol{Z}$ for the possibility to find a control $\boldsymbol{u}$ such that $(\boldsymbol{u},\ \mathcal{Z})$ is admissible for I_N is a huge problem. Indeed, from Remark 10.7, one has

$$\mathcal{Z} \in \boldsymbol{Z} \Rightarrow \mathcal{Z}_{\min} \preceq \mathcal{Z} \preceq \mathcal{Z}_{\max}$$

and since $\mathcal{Z}_{\min} = \{\boldsymbol{y}_k,\ k = 1, \ldots, s\}$ and $\mathcal{Z}_{\max} = \{\boldsymbol{y},\ k = 1, \ldots, s\}$, it follows that for any information pattern $\mathcal{Z} = \{z_k,\ k = 1, \ldots, s\}$ one has $z_k = \boldsymbol{y}_k \cup \gamma_k$, where $\gamma_k \subseteq \boldsymbol{y} \backslash \boldsymbol{y}_k$ is the subset of measurements that are "added" to the ones already available to subsystem Σ_k in the local information pattern $\mathcal{Z}_{\min}$. It follows that

$$\boldsymbol{Z} = \prod_{k=1,\ldots,s} 2^{\boldsymbol{y} \backslash \boldsymbol{y}_k}.$$

Example 10.7 The number of candidate information patterns
Assume that four subsystems have the local measurement vectors $y_1 \in |\mathcal{R},\ \boldsymbol{y}_2 \in |\mathcal{R}^2$, $y_3 \in |\mathcal{R}$ and $\boldsymbol{y}_4 \in |\mathcal{R}^2$. Then, in addition to y_1, Σ_1 could receive any subset of the other five measurements, Σ_2 could receive any subset of the other four in addition to $\boldsymbol{y}_2$, etc. This gives a total of $2^5 \times 2^4 \times 2^5 \times 2^4$ information patterns that are wider than $\mathcal{Z}_{\min}$. □

Reducing the set of candidate information patterns. In this section, we consider simple arguments that allow to reduce the number of information patterns to be explored in order to determine $\mathcal{Z}(I_N)$.

Theorem 10.4 *Let $(I_C,\ \mathcal{Z}_C)$ be the current system situation and assume that a fault occurs such that $I_N \notin \mathcal{R}(\mathcal{Z}_C)$ but $I_N \in \mathcal{R}(\mathcal{Z}_{\max})$. Then, it holds that*

$$\mathcal{Z}(I_N) \subseteq \mathcal{Z} \backslash \mathcal{N}(\mathcal{Z}_C),$$

where $\mathcal{N}(\mathcal{Z}_C) = \{\mathcal{Z}_C^- : \mathcal{Z}_C^- \preceq \mathcal{Z}_C\}$ is the set of information patterns that are narrower than $\mathcal{Z}_C$ with respect to the order relation $\preceq$. □

Indeed, noting that $I_N \notin \mathcal{R}(\mathcal{Z}_C) \Rightarrow \mathcal{Z}_C \notin \mathcal{Z}(I_N)$, the result follows from Eq. (10.20) which implies

$$\mathcal{Z}_C^- \preceq \mathcal{Z}_C \iff \mathcal{Z}_C^- \notin \mathcal{Z}(I_N).$$

Furthermore, in order to determine all the elements of $\mathcal{Z}(I_N)$, it is enough to find its minimal ones. Remember that a minimal information pattern $\mathcal{Z}_{\min}$ in $\mathcal{Z}(I_N)$ is such that

$$\mathcal{Z}_{\min} \in \mathcal{Z}(I_{\mathrm{N}})$$
$$\mathcal{Z} \preceq \mathcal{Z}_{\min} \Rightarrow \mathcal{Z} \notin \mathcal{Z}(I_{\mathrm{N}}).$$

Theorem 10.5 *Let $\mathcal{Z}_{\mathrm{M}}(I_{\mathrm{N}})$ be the set of minimal elements of $\mathcal{Z}(I_{\mathrm{N}})$. Then one has*

$$\mathcal{Z}(I_{\mathrm{N}}) = \bigcup_{\mathcal{Z} \in \mathcal{Z}_{\mathrm{M}}(I_{\mathrm{N}})} \mathcal{W}(\mathcal{Z}),$$

where $\mathcal{W}(\mathcal{Z}) = \left\{\mathcal{Z}^{+} : \mathcal{Z} \preceq \mathcal{Z}^{+}\right\}$ is the set of information patterns that are wider than $\mathcal{Z}$ with respect to the order relation $\preceq$.

Indeed, it is clear that $\mathcal{Z}_{\min} \in \mathcal{Z}_{\mathrm{M}}(I_{\mathrm{N}}) \Rightarrow \mathcal{Z}_{\min} \in \mathcal{Z}(I_{\mathrm{N}})$. Then, the implication

$$\mathcal{Z} \in \mathcal{Z}(I_{\mathrm{N}}) \setminus \mathcal{Z}_{\mathrm{M}}(I_{\mathrm{N}}) \Rightarrow \exists \mathcal{Z}_{\min} \in \mathcal{Z}_{\mathrm{M}}(I_{\mathrm{N}}) : \mathcal{Z} \in \mathcal{W}(\mathcal{Z}_{\min})$$

is true, because $\mathcal{Z}$ being not minimal, and there exists an information pattern $\mathcal{Z}_1$ such that $\mathcal{Z}_1 \in \mathcal{Z}(I_{\mathrm{N}})$ and $\mathcal{Z} \in \mathcal{W}(\mathcal{Z}_1)$. If $\mathcal{Z}_1 \in \mathcal{Z}_{\mathrm{M}}(I_{\mathrm{N}})$, the conclusion of the theorem is obtained. If not, which means $\mathcal{Z}_1$ is not minimal, there exists an information pattern $\mathcal{Z}_2$ such that $\mathcal{Z}_2 \in \mathcal{Z}(I_{\mathrm{N}})$ and $\mathcal{Z}_1 \in \mathcal{W}(\mathcal{Z}_2)$, which implies $\mathcal{Z} \in \mathcal{W}(\mathcal{Z}_2)$ by transitivity. If $\mathcal{Z}_2 \in \mathcal{Z}_{\mathrm{M}}(I_{\mathrm{N}})$, the conclusion is obtained; otherwise, the process is repeated until the conclusion holds, which must eventually occur because $\mathcal{Z}$ contains a finite number of information patterns. Finally, the monotonicity property (10.20) implies that

$$\mathcal{Z} \in \bigcup_{\mathcal{Z} \in \mathcal{Z}_{\mathrm{M}}(I_{\mathrm{N}})} \mathcal{W}(\mathcal{Z}) \Rightarrow \mathcal{Z} \in \mathcal{Z}(I_{\mathrm{N}}).$$

Remark 10.8 A subset of $\mathcal{Z}(I_{\mathrm{N}})$ is found by exploring only a subset of $\mathcal{Z} \setminus \mathcal{N}(\mathcal{Z}_{\mathrm{C}})$, provided it contains at least one admissible information pattern. This obviously happens with $\mathcal{W}(\mathcal{Z}_{\mathrm{C}})$ since one has $I_{\mathrm{N}} \in \mathcal{R}(\mathcal{Z}_{\max})$. In the sequel, we look for solutions within $\mathcal{W}(\mathcal{Z}_{\mathrm{C}})$ because information patterns that are wider than $\mathcal{Z}_{\mathrm{C}}$ are easy to construct, especially if technological constraints associated with the communication system are taken into account. The next sections, respectively, consider the publisher/subscriber and the bilateral agreements schemes. □

10.5.3 Publisher/Subscriber Scheme

Optimal subscriptions. Before we address the construction of wider information patterns in the publisher/subscriber scheme, let us first remark that since the communication cost is associated with the published variables, the best use of the published variables is achieved when the local controllers subscribe to all of them. Indeed, let $\mathcal{Z}_{\mathrm{C}} = \left\{z_{\mathrm{C,k}},\ k = 1, \ldots, s\right\}$ be the current information pattern, and let γ_{C} and $\gamma_{\mathrm{C,k}}$ be, respectively, the current set of published variables, and the current set of

variables subscribed by subsystem Σ_k (meaning that $z_{C,k} = y_k \cup \gamma_{C,k}$). One has $\gamma_{C,k} \subseteq \gamma_C$, $k = 1, \ldots, s$ but from Eq. (10.20) it follows that for the cost of publishing the variables γ_C, the largest sets of recoverable configurations are obtained with the subscriptions $\gamma_{C,k} = \gamma_C, k = 1, \ldots, s$. However, it is worth to remark that in order to recover a given configuration, there is no obligation for all subsystems to subscribe to all the published variables γ_C.

The set of wider information patterns. As opposed to distributed diagnosis, only subsets of measurements are published for distributed fault-tolerant control. Therefore, in the publisher/subscriber scheme, any information pattern wider than $\mathcal{Z}_C$ is associated with the publication of a subset of variables in $y \backslash \gamma_C$. It follows that $\mathcal{W}(\mathcal{Z}_C)$ is the lattice $2^{y \backslash \gamma_C}$. Note that the minimal elements in $\mathcal{W}(\mathcal{Z}_C)$ are the first ones found when exploring $2^{y \backslash \gamma_C}$ by increasing levels (indeed they are associated with the minimal sets of measurements to be published in addition to those already present in $\mathcal{Z}_C$).

Example 10.8 Information pattern reconfiguration in the publisher/subscriber scheme

Let y_1, y_2, y_3, y_4 be the local measurements associated with a system composed of four subsystems. In the information pattern $\mathcal{Z}_C = \{(y_1, y_2), y_2, (y_1, y_3), (y_2, y_4)\}$, the published variables are $\{y_1, y_2\}$. However, $\mathcal{Z} = \{(y_1, y_2), (y_1, y_2), (y_1, y_2, y_3), (y_1, y_2, y_4)\}$ is wider and has the same communication cost. Assume $\mathcal{Z}_C$ is the current information pattern and a fault that is not recoverable under $\mathcal{Z}_C$, but is recoverable under $\mathcal{Z}_{max}$ occurs. Then, a subset of solutions to the information pattern reconfiguration problem is generated by considering

$$\mathcal{W}(\mathcal{Z}_C) = \left\{\mathcal{Z}(\gamma) : k = 1, \ldots, 4,\ z_k = z_{C,k} \cup \gamma, \gamma \subseteq y_3 \cup y_4\right\}.$$

Note that any information pattern in $\mathcal{W}(\mathcal{Z}_C)$ is obtained by publishing one subset of $\{y_3, y_4\}$ in addition to the variables already published in $\mathcal{Z}_C$. Note also that $\mathcal{W}(\mathcal{Z}_C)$ is a subset of $\mathcal{W}(\mathcal{Z}_{min})$ because any $\mathcal{Z}_C \in \mathcal{Z}$ is wider than $\mathcal{Z}_{min}$. Finally, note that $\mathcal{W}(\mathcal{Z}_{min})$ can be determined off-line, since the set of publishable data is nothing but the lattice of the sensors subsets. This lattice is displayed in Fig. 10.8, where $\emptyset$ corresponds to $\mathcal{Z}_{min}$ (no data are published), while 1234, which stands for $\{y_1, y_2, y_3, y_4\}$, is associated with $\mathcal{Z}_{max}$ (all sensor outputs are published). Since $\{y_1, y_2\}$ are currently published in $\mathcal{Z}_C$ (so $\mathcal{Z}_C$ is represented by node 12), $\mathcal{W}(\mathcal{Z}_C)$ is the sub-lattice with grey nodes. □

10.5.4 Bilateral Communication Scheme

The set of wider information patterns. Let $\mathcal{A}_C$ characterise the current set of agreements, leading to the equivalence classes $\mathcal{E}(\mathcal{A}_C) = \left\{E_{C,l},\ l = 1, \ldots, \sigma\right\}$, and the current information pattern $\mathcal{Z}_C = \left\{Z_{C,k},\ k = 1, \ldots, s\right\}$. A wider information pattern can only be obtained by establishing a set of agreements whose graph $\mathcal{A}_W$ is such that $\mathcal{A}_C \subset \mathcal{A}_W$. This results in the equivalence classes $\mathcal{E}(\mathcal{A}_W) = \left\{E_{W,k},\ k = 1, \ldots, \rho\right\}$ where $\rho < \sigma$ such that each class of $\mathcal{E}(\mathcal{A}_W)$ is equal to one class of $\mathcal{E}(\mathcal{A}_C)$, or is the

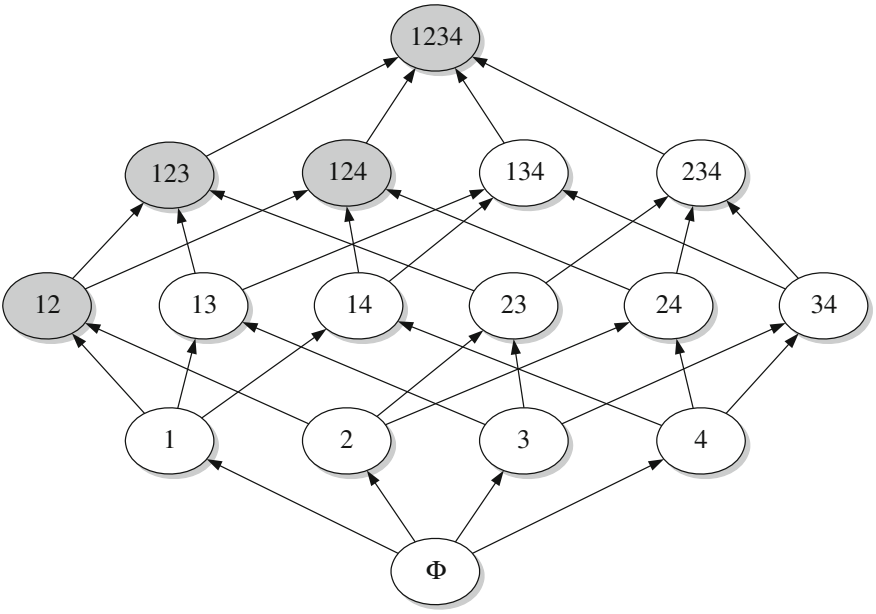


Fig. 10.8 Publishable sets of data

union of several classes of $\mathcal{E}(\mathcal{A}_{\mathrm{C}})$. The set $\mathcal{W}(\mathcal{Z}_{\mathrm{C}})$ of information patterns wider than $\mathcal{Z}_{\mathrm{C}}$ is then

$$\mathcal{W}(\mathcal{Z}_{\mathrm{C}}) = \left\{\mathcal{Z}_{\mathrm{W}} = \left\{z_{\mathrm{W,i}},\ i = 1, \ldots, s :\ \Sigma_i \in E_{\mathrm{W,k}} \longrightarrow z_{\mathrm{W,i}} = z\left(E_{\mathrm{W,k}}\right), \forall \mathcal{A}_{\mathrm{W}} \supset \mathcal{A}_{\mathrm{C}}\right\}\right\}.$$

Hierarchical decomposition. In order to construct $\mathcal{W}(\mathcal{Z}_{\mathrm{C}})$, remark that sets of agreements and system decompositions are related. Indeed, given a set of agreements $\mathcal{A}$ and the equivalence classes $\mathcal{E}(\mathcal{A}) = \{E_{\mathrm{l}},\ l = 1, \ldots, \sigma\}$, all the subsystems in class E_{l} share the same information, and therefore they constitute one (high-level) subsystem $\Sigma(E_{\mathrm{l}}), l = 1, \ldots, \sigma$ whose state $\boldsymbol{x}(E_{\mathrm{l}})$ and control $\boldsymbol{u}(E_{\mathrm{l}})$ are the concatenation of the local states $\{\boldsymbol{x}_i,\ \Sigma_i \in E_{\mathrm{l}}\}$ and controls $\{\boldsymbol{u}_i,\ \Sigma_i \in E_{\mathrm{l}}\}$. Since $\mathcal{E}(\mathcal{A})$ is a partition of Σ into $\sigma \leq s$ classes, the decomposition of Σ defined by $\{\Sigma(E_{\mathrm{l}}), l = 1, \ldots, \sigma\}$ is *coarser* than $\{\Sigma_i,\ i = 1, \ldots, s\}$, meaning that every subsystem Σ_i is included in one and only one $\Sigma(E_{\mathrm{l}})$.

It follows that there is a one to one correspondence between the set of all bilateral agreements and a hierarchy $\mathcal{H}$ of decompositions of Σ.

Definition 10.7 (*Hierarchy of decompositions*) A hierarchy $\mathcal{H}$ is a set of decompositions of Σ organised into levels $\mathcal{H}_\sigma$ that contain decompositions into σ subsystems. Level $\mathcal{H}_1$ is the overall system, while level $\mathcal{H}_s$ is $\{\Sigma_i,\ i = 1, \ldots, s\}$, called the *atomic* decomposition. Two decompositions $\mathcal{E}_0$ and $\mathcal{E}_1$ that belong to two adjacent levels $\mathcal{H}_\sigma$

and $\mathcal{H}_{\sigma-1}$ contain the same subsystems, except for one subsystem in $\mathcal{E}_1$ that is the union of two subsystems in $\mathcal{E}_0$.

Figure 10.9 illustrates two possible hierarchies for the decomposition of a system with four subsystems. Each subsystem is represented by its index, for example, 1 stands for Σ_1 while 34 represents the union of the two subsystems $\Sigma_3 \cup \Sigma_4$ and 1234 stands for the overall system $\cup_{i=1,\ldots,4} \Sigma_i$.

Based on the correspondence between bilateral agreements and decomposition hierarchies, it follows that the minimal elements of $\mathcal{W}(\mathcal{Z}_C)$ are associated with the first decompositions found when exploring the hierarchy by decreasing levels (indeed, they concern the minimum sets of variables shared between subsystems in addition to the variables already shared in the previous step).

Example 10.9 Information pattern reconfiguration in the bilateral agreements scheme

Figure 10.10 displays the hierarchy associated with all possible information patterns under bilateral agreements, for the four subsystems' example. The atomic decomposition $\{\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4\}$ is abbreviated as 1, 2, 3, 4, while 1234 stands for the overall system $\cup_{i=1,\ldots,4} \Sigma_i$, and 14, 23 represent the decomposition into two subsystems $\Sigma_1 \cup \Sigma_4$ and $\Sigma_2 \cup \Sigma_3$. Assuming the current information pattern is $\mathcal{Z}_C = \{(y_1, y_3), y_2, (y_1, y_3), y_4\}$ (represented by the nodes 13, 2, 4 with a bold contour in Fig. 10.10), the white sub-lattice shows the set $\mathcal{W}(\mathcal{Z}_C)$ under the bilateral agreements communication scheme. The minimal elements are (123, 4), (134, 2) and (13, 24). The figure also shows the three hierarchies associated with the three paths between the node (13, 2, 4) associated with the current information pattern and the node (1234) associated with the maximal information pattern. □

Example 10.10 Information pattern reconfiguration

Figure 10.7 displays the non-recoverable configurations under the local information pattern associated with the decentralised control of the system in Example 10.1. Applying Theorem 10.3, it can be checked that some configurations that are non-recoverable under the local information pattern become recoverable by an information pattern extension.

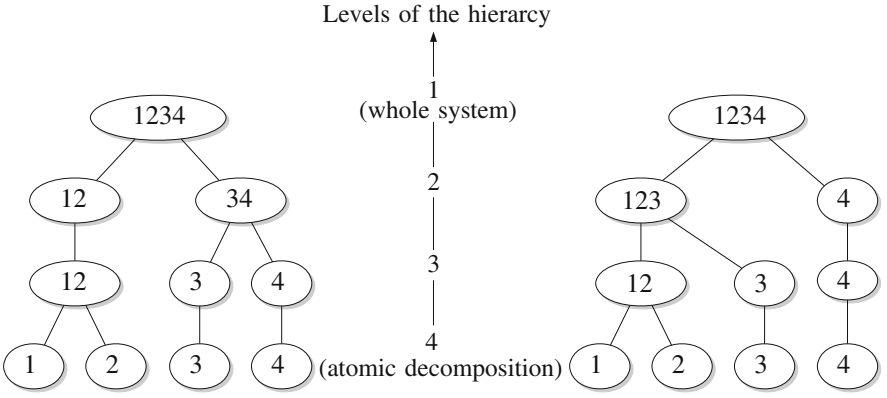


Fig. 10.9 Two possible decomposition hierarchies

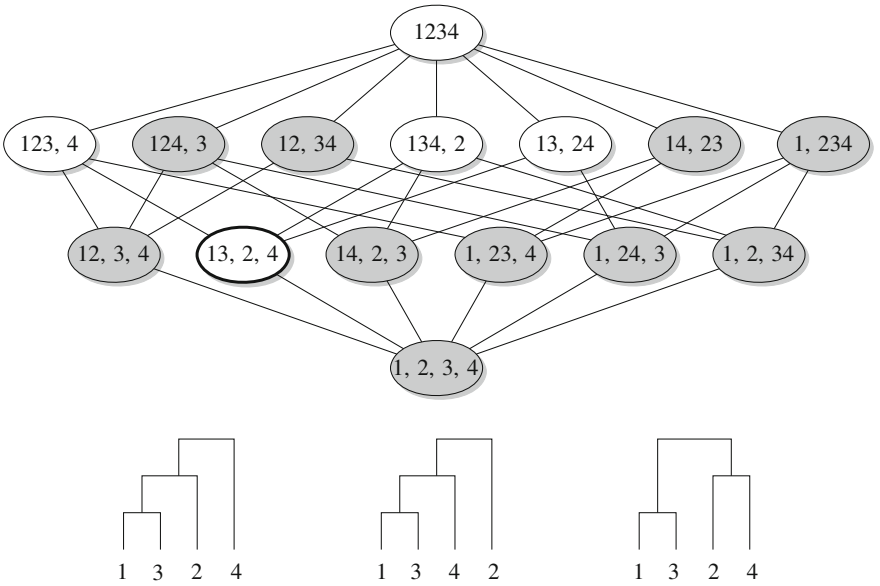


Fig. 10.10 Information patterns under bilateral agreements for four subsystems

Let us focus, for example, on configuration 1345 which is recoverable under a reconfigured information pattern. The publication of y_2 leads to the reconfigured information pattern $\mathcal{Z}_{1345} = \{(y_1, y_2), y_2, (y_2, y_3), (y_2, y_4)\}$ under which there exists a reconfigured distributed control of the form:

$$\begin{aligned} u_1 &= k_{11}y_1 + k_{12}y_2 \\ u_3 &= k_{32}y_2 \\ u_4 &= k_{42}y_2 + k_{43}y_3 \\ u_5 &= k_{52}y_2 + k_{54}y_4. \end{aligned} \tag{10.21}$$

It can indeed be checked that the reconfigured control laws,

$$\begin{aligned} u_1 &= 9.6660y_1 + 8.3264y_2 \\ u_3 &= 0.6995y_2 \\ u_4 &= 2.9872y_2 + 13.8406y_3 \\ u_5 &= 1.7713y_2 + 2.0056y_4, \end{aligned}$$

allows the system to be recovered after the fault. However, it is interesting to remark that the solution of Eq. (10.21) is not unique. Indeed, Eq. (10.22) exhibits another solution, such that k_{42} and k_{52} are both equal to zero, meaning that Σ_2, Σ_3 and Σ_4 still work in a decentralised way. This illustrates the fact that while $\mathcal{Z}_{1345}$ is the most efficient information pattern associated with the publication of y_2 as noted in the presentation of the publisher/subscriber scheme (Optimal subscriptions), solutions based on narrower information patterns might also exist. Clearly, the publication of y_2 is necessary for configuration 1345 to become recoverable, but that does not mean that all subsystems have to subscribe to the newly published variable y_2:

$$\begin{aligned} u_1 &= 9.8167y_1 + 8.800y_2 \\ u_3 &= 0.7470y_2 \\ u_4 &= 9.2974y_3 \\ u_5 &= 9.1418y_4. \end{aligned} \tag{10.22}$$

On another hand, should the bilateral communication scheme be of interest, the hierarchy of Fig. 10.10 suggests the reconfigured information pattern $\mathcal{Z}_{1345} = \{(y_1, y_2), (y_1, y_2), y_3, y_4\}$ which results in an admissible controller:

$$\begin{aligned} u_1 &= 9.5717y_1 + 8.0028y_2 \\ u_3 &= 4.8538y_1 + 1.4189y_2 \\ u_4 &= 8.4352y_3 \\ u_5 &= 18.3147y_4. \ \square \end{aligned} \tag{10.23}$$

10.5.5 Extensions

Several extensions of the information pattern reconfiguration frame can be considered. The simplest one addresses sensor (or general system component) faults, since only actuator faults (hence actuator configurations) have been considered up to now. Other extensions address optimality issues in the information pattern reconfiguration process, namely the selection of an optimal information pattern reconfiguration from the point of view of the communication cost and the more complex issue of minimising the reconfiguration effort.

Sensor faults. Only actuator faults have been considered up to now, for the sake of simplicity. It is easy to see that sensor faults (more generally system components faults) can easily be dealt with in the reconfiguration strategy. Indeed, assume there is a set J_F of faulty sensors, then the available sensors are $J_N = J \backslash J_F$ (note that faults in the communication system that prevent the measurements of some sensors to be transmitted to the controllers that need them are also represented by this model). The pre-fault information pattern $\mathcal{Z} = \{z_k,\ k = 1, \ldots, s\}$ becomes the post-fault information pattern $\mathcal{Z}_N = \{z_k \cap \boldsymbol{y}_N,\ k = 1, \ldots, s\}$, and $\mathcal{Z}_{\max} = \{\boldsymbol{y},\ k = 1, \ldots, s\}$ becomes $\mathcal{Z}_{N\max} = \{\boldsymbol{y}_N,\ k = 1, \ldots, s\}$. The fault is recoverable if and only if the current actuator configuration I_N and the current sensor configuration J_N are such that $I_N \in \mathcal{R}(\mathcal{Z}_{N\max})$.

Minimal communication cost. Since each $\mathcal{Z} \in \mathcal{Z}(I_N)$ is associated with the communication cost $com(\mathcal{Z}, \mathcal{J})$, selecting the information pattern $\mathcal{Z}^*$ such that

$$\mathcal{Z}^* = \arg \min_{\mathcal{Z} \in \mathcal{Z}(I_N)} com(\mathcal{Z}, \mathcal{J}) \tag{10.24}$$

provides an optimally reconfigured information pattern with respect to the communication cost. It is easily proved, from the monotonicity of the cost function $com(\mathcal{Z}, \mathcal{J})$, that the solutions of Eq. (10.24) belong to the set $\mathcal{Z}_M(I_N)$ of the minimal elements of $\mathcal{Z}(I_N)$. Following Remark 10.8, sub-optimal solutions are easily obtained from

$$\mathcal{Z}^*_{\text{sub}} = \arg \min_{\mathcal{Z} \in \mathcal{W}(\mathcal{Z}_C)} com\,(\mathcal{Z}, \mathcal{J}) \tag{10.25}$$

once the set $\mathcal{W}(\mathcal{Z}_C)$ has been determined.

10.5.6 Minimal Reconfiguration Effort

Let us consider again the case where the system is operating with the current subset of actuators $I_C \subseteq I$ and the current information pattern $\mathcal{Z}_C$, and a fault occurs such that the post-fault configuration $I_N \subset I_C$ no longer belongs to $\mathcal{R}(\mathcal{Z}_C)$. Theorem 10.3 gives a necessary and sufficient condition for the existence of a solution to the information pattern reconfiguration problem, namely "Is there an information pattern $\mathcal{Z}_N$ such that $I_N \in \mathcal{R}(\mathcal{Z}_N)$?" When solutions exist, Theorem 10.5 and Remark 10.8 provide some practical tools to find such information patterns, whose communication cost can be minimised by solving the problem (10.24) (or (10.25)). However, these results do not provide any characterisation of the *number of subsystems* whose data or control laws have to be reconfigured, a number that clearly characterises the reconfiguration effort. In order to address this point, we will now consider constraints on the possible reconfigured information patterns or the possible reconfigured control laws.

Σ_K-recoverability. We first start with the notion of Σ_K-recoverability, which addresses the number of subsystems whose available data have to be reconfigured after the occurrence of a fault.

Definition 10.8 (Σ_K-*recoverability*) Let $\Sigma_K = \{\Sigma_k, k \in K \subseteq \{1, \ldots, s\}\}$ be a subset of subsystems. A configuration I_N is Σ_K-recoverable if it is recoverable by reconfiguring only the data available to the subsystems in Σ_K.

Theorem 10.6 *Let* $(I_C, \mathcal{Z}_C)$ *be the current system configuration, and assume a fault occurs such that the resulting configuration is* $I_N \subset I_C$. *A necessary and sufficient condition for configuration* I_N *to be* Σ_K*-recoverable is that* $I_N \in \mathcal{R}\left(\mathcal{Z}_{K,\max}\right)$ *where* $\mathcal{Z}_{K,\max} = \mathcal{Z}_C$ *except for* $z_k = y$, $k \in K$.

The idea of this theorem is quite similar to the idea of Theorem 10.3.

Comments.

1. Taking $K = \{1, \ldots, s\}$, i.e. accepting the possibility for the data of all the subsystems to be reconfigured, is just the problem addressed by Theorem 10.3.
2. The set of all possible subsets Σ_K is the lattice $2^{\{1,\ldots,s\}}$, which implies the monotonicity property that if I_N is Σ_L-recoverable, and $L \subseteq K$, then I_N is Σ_K-recoverable. It follows that the minimal subsets Σ_L such that I_N is Σ_L-recoverable, defined by

$$I_{\mathrm{N}} \in \mathcal{R}\left(\mathcal{Z}_{\mathrm{L,max}}\right)$$
$$\forall K \subset L, I_{\mathrm{N}} \notin \mathcal{R}\left(\mathcal{Z}_{\mathrm{K,max}}\right)$$

can be found using a classical bottom-up algorithm on the lattice $2^{\{1,\ldots,s\}}$.

Example 10.11 Σ_{K}**-recoverability**
It has been seen that configuration 1345 is not recoverable under the decentralised information pattern $\mathcal{Z}_{\min} = \{y_1, y_2, y_3, y_4\}$ but becomes recoverable when it is reconfigured as $\mathcal{Z}_{1345} = \{(y_1, y_2), y_2, y_3, y_4\}$.

Figure 10.11 shows all the subsets Σ_{K} such that configuration 1345 is Σ_{K} recoverable.

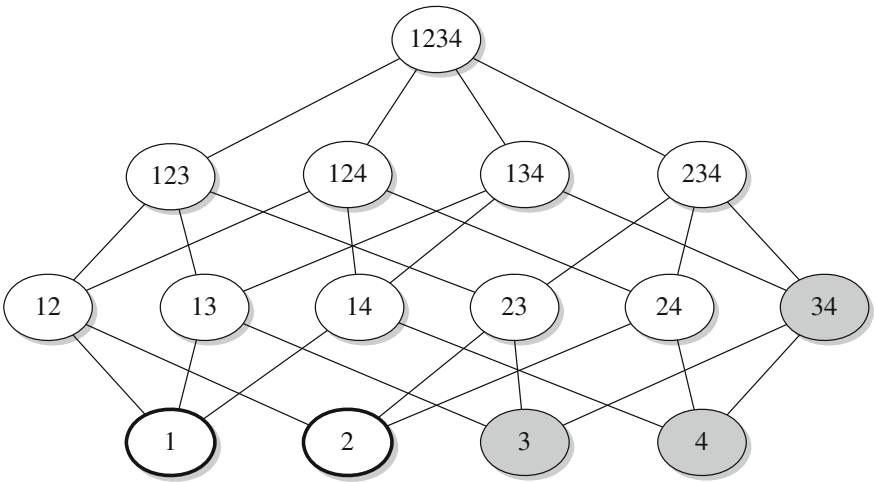


Fig. 10.11 Σ_{K}-recoverability of configuration 1345

Remark 10.9 The Σ_{K}-recoverability of a configuration I_{N} means that I_{N} becomes recoverable if the system information pattern is reconfigured so that only the data available to the subsystems in Σ_{K} are reconfigured (and so are the corresponding control laws). Although their available data remain unchanged, note that the control parameters of the subsystems that do not belong to Σ_{K} are allowed to change. A stronger version of the minimal reconfiguration effort problem is set by constraining the controls of those subsystems that do not belong to Σ_{K} to remain unchanged. In order to solve this problem, strong Σ_{K}-recoverability is now defined. □

Strong Σ_{K}-recoverability. The following notion is defined for a more detailed analysis:

Definition 10.9 (*Strong Σ_{K}-recoverability*) Let $\Sigma_{\mathrm{K}} = \{\Sigma_k, k \in K \subseteq \{1, \ldots, s\}\}$ be a subset of subsystems and let $\boldsymbol{u}$ be decomposed into $(\boldsymbol{u}_{\mathrm{K}}, \overline{\boldsymbol{u}}_{\mathrm{K}})$ where $\boldsymbol{u}_{\mathrm{K}}$ gathers the controls of those subsystems that belong to Σ_{K} while $\overline{\boldsymbol{u}}_{\mathrm{K}}$ gathers the controls of

the other ones. A configuration I_N is strongly Σ_K-recoverable if it is recoverable by reconfiguring only the data available to the subsystems in Σ_K and the control laws in $\boldsymbol{u}_K$.

Theorem 10.7 *Let $\Sigma_K = \{\Sigma_k, k \in K \subseteq \{1, \ldots, s\}\}$ be a subset of subsystems and let $\boldsymbol{u}$ be decomposed into $(\boldsymbol{u}_K, \overline{\boldsymbol{u}}_K)$. Let $(I_C, \mathcal{Z}_C)$ be the current system situation, and assume a fault occurs which results in configuration $I_N \subset I_C$. A necessary and sufficient condition for configuration I_N to be strongly Σ_K-recoverable is that there exists a control law $\boldsymbol{v}_K$ such that $(\boldsymbol{v}_K, \overline{\boldsymbol{u}}_K)$ is admissible under $\mathcal{Z}_{K,\max}$ where $\mathcal{Z}_{K,\max} = \mathcal{Z}_C$ except for $Z_k = J, \forall k \in K$.*

Example 10.12 Strong Σ_K-recoverability
Let us consider again configuration 1345, which is not recoverable under the decentralised information pattern $\mathcal{Z}_{\min} = \{y_1, y_2, y_3, y_4\}$ but becomes recoverable when it is reconfigured as $\mathcal{Z}_{1345} = \{(y_1, y_2), y_2, y_3, y_4\}$. By comparing Eqs. (10.19) and (10.22), it is seen that although the data they use were unchanged, subsystems Σ_2, Σ_3 and Σ_4 did reconfigure the parameters of their control laws from $u_3 = 4.6939y_2, u_4 = 0.1190y_3, u_5 = 3.3712y_4$ to $u_3 = 0.7470y_2, u_4 = 9.2974y_3, u_5 = 9.1418y_4$. Unfortunately, there exists no solution to the strong Σ_1-recoverability problem: it is indeed impossible to recover configuration 1345 by changing only the data and the control law of subsystem Σ_1. Figure 10.12 displays the result obtained when applying Theorem 10.7: the white nodes are those subsets of subsystems with respect to which actuator configuration 1345 is strongly recoverable. In other words, it is possible to recover configuration 1345 by reconfiguring only the data sets and control laws of those subsystems. Note that if a configuration is strongly Σ_K-recoverable, then it is also strongly Σ_L-recoverable for any subset of subsystems Σ_L that includes Σ_K. In the figure, the minimal subsets of subsystems such that 1345 is strongly recoverable are shown with a bold contour. These subsystems are associated with the minimal reconfiguration effort to recover the configuration of interest, namely the minimal number of data and control laws to be reconfigured for its recovery to be possible. In this example, it is seen that configuration 1345 is strongly Σ_2-recoverable. It can indeed be checked that the control laws,

$$\begin{aligned} u_1 &= 1.2991y_1 \\ u_3 &= 9.3399y_1 + 6.7874y_2 + 7.5774y_3 + 8.4913y_4 \\ u_4 &= 0.1190y_3 \\ u_5 &= 3.3712y_4 \end{aligned}$$

where u_1, u_4 and u_5 are unchanged from the nominal decentralised case, satisfy the α-stability specification.

10.6 Exercises

Exercise 10.1 Diagnosis of the two-tank system
In this exercise, we develop the complete diagnosis scheme of the two-tank system in Chap. 2, where two level sensors h_{1m} and h_{2m} were implemented in addition to the flow sensor q_m. The set of constraints and unknown variables are the following:

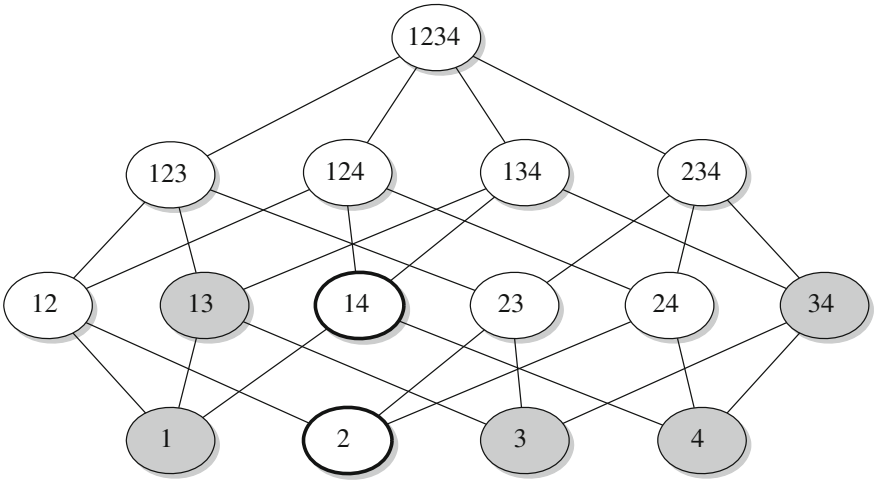


Fig. 10.12 Strong Σ_{K}-recoverability of configuration 1345

$$\begin{aligned} \boldsymbol{f} \cup \boldsymbol{g} &= \{c_1, c_2, c_3, d_4, c_5, c_6, d_7, c_8, c_{\mathrm{m}}, c_{h1}, c_{h2}\} \\ \mathcal{X} &= \left\{q_L, q_P, h_1, \dot{h}_1, h_2, \dot{h}_2, q_2, q_{12}\right\}. \end{aligned}$$

The correspondence with the model in Chap. 2 is as follows: c_1 is Eq. (2.7), c_2 is Eq. (2.6), c_3 is Eq. (2.1), c_5 is Eq. (2.4), c_6 is Eq. (2.2) and c_8 is Eq. (2.5). The measurement equations are c_{m} which is Eq. (2.3) and c_{h1}, c_{h2} which are, respectively, the added measurements of the two levels h_1 and h_2. The constraints d_4 and d_7, respectively, express that $\dot{h}_1$ and $\dot{h}_2$ are the time derivatives of h_1 and h_2. The incidence matrix with respect to $\mathcal{X}$ is:

Σ_1	q_L	q_P	$\dot{h}_1$	h_1	q_{12}	h_2	$\dot{h}_2$	q_2
c_1	①			1				
c_2		①		1				
c_3	1	1		1	①			
d_4			①	1				
c_5				1	1	1		
c_{h1}				①				
c_6					1		1	1
d_7						1	①	
c_8						1		1
c_{m}								①
c_{h2}						①		

Based on the complete matching shown by the entries ①, the over-constrained subsystem produces three residuals whose structures are

$$\begin{aligned}\mathcal{C}(\rho_1) &= \{c_1, c_2, c_3, c_5, c_{h1}, c_{h2}\}\\ \mathcal{C}(\rho_2) &= \{c_1, c_2, c_3, c_6, d_7, c_{\mathrm{m}}, c_{h1}, c_{h2}\}\\ \mathcal{C}(\rho_3) &= \{c_8, c_{\mathrm{m}}, c_{h2}\}.\end{aligned}$$

1. What is the residuals' signature table.
2. The mathematical constraints d_4 and d_7 specify that $\dot{h}_1$ and $\dot{h}_2$ are the derivatives of h_1 and h_2. Discarding them (since they cannot be faulty), determine the system's distinguishability classes and draw the distinguishability table.
3. For each of the eight possible residual configurations, find the minimal hitting sets and draw the diagnosis table. □

Exercise 10.2 Two-tank system decomposition
This exercise illustrates Remark 10.5 still with the two-tank system. Assume each tank is a subsystem with the structures:

$$\begin{aligned}\boldsymbol{f}_1 \cup \boldsymbol{g}_1 &= \{c_1, c_2, c_3, d_4, c_5, c_{h1}\}\\ \boldsymbol{x}_1 &= \{q_L, q_P, h_1, \dot{h}_1, q_{12}\}\\ \overline{\boldsymbol{x}}_1 &= \{h_2\}\\ \boldsymbol{f}_2 \cup \boldsymbol{g}_2 &= \{c_6, d_7, c_8, c_{\mathrm{m}}, c_{h2}\}\\ \boldsymbol{x}_2 &= \{h_2, \dot{h}_2, q_2\}\\ \overline{\boldsymbol{x}}_2 &= \{q_{12}\}.\end{aligned}$$

The global incidence matrix is decomposed as follows,

Σ_1	q_L	q_P	$\dot{h}_1$	h_1	q_{12}	h_2
c_1	①			1		
c_2		①		1		
c_3	1	1		1	①	
d_4			①	1		
c_5				1	1	①
c_{h1}				①		

Σ_2	$\dot{h}_2$	h_2	q_2	q_{12}
c_6	1		1	①
d_7	①	1		
c_8		1	1	
c_{m}			①	
c_{h2}		①		

and two complete matchings with respect to the unknown variables are shown by ①:

1. How many local residuals are, respectively, provided by Σ_1 and Σ_2 and what are their structures?
2. Can you explain why there are less local residuals than when considering the global structure? □

Exercise 10.3 Coordination of local diagnosis
Consider a system in which there are three different estimation versions of an unknown variable x from the known variables $u' \cup y'$ (remember that the notation u', y' means u, y and a number of their time derivatives):

$$\begin{aligned} x &= f_1\left(u', y'\right) \quad \text{using the subset of constraints } C_1 = \{a, b, c, d\} \\ x &= f_2\left(u', y'\right) \quad \text{using the subset of constraints } C_2 = \{e, f\} \\ x &= f_3\left(u', y'\right) \quad \text{using the subset of constraints } C_3 = \{b, f, g, h\}. \end{aligned}$$

Three residuals are obtained:

$$\begin{aligned} \rho_1 &= f_1\left(u', y'\right) - f_2\left(u', y'\right) \\ \rho_2 &= f_1\left(u', y'\right) - f_3\left(u', y'\right) \\ \rho_3 &= f_2\left(u', y'\right) - f_3\left(u', y'\right). \end{aligned}$$

1. What are the structures of the residuals?
2. What is the distinguishability table?
3. Assuming there are three subsystems that run one residual each, what are the local diagnosis tables?
4. What is the coordinated diagnosis table? □

10.7 Bibliographical Notes

Fault-tolerant computing. Due to the increasing complexity of software applications and the increasing size of data bases, distributed computing and related reliability issues have been an important research area in the Computer Science community. A conceptual taxonomy of the basic concepts in the dependability of computer systems (reliability, availability, safety, confidentiality, integrity, maintainability, etc.) is presented in [10]. Many solutions have been proposed, ranging from node-level to system-level approaches: redundant execution of critical programmes on several nodes, providing each node with a fail-aware ability, with the capacity of testing its neighbours, of estimating the state of all nodes, of entering a fail-silent state [2, 101].

New problems in networked and multi-agent systems. A huge research activity has also been triggered in the control community on large-scale control systems distributed over networks and multi-agent systems. New theoretical problems range from the role of the information pattern in the problem solvability [137] to the controllability and observability analysis of networked dynamical systems [413]. Technological problems are not only connected with the controlled process (sensors, actuators, process components faults) but also with channel limitations (packet rates,

sampling, delays) or channel failures (packet dropouts). The impact of such faults on the system stability and performance is analysed in [144, 369], and a survey of recent results on estimation, analysis and controller synthesis can be found in [60, 146, 307]. Estimation by means of geographically distributed sensors has been thoroughly studied using linear estimation [53, 81, 315] and Kalman filtering [1].

Analytical redundancy-based diagnosers. Although it involves signal derivatives, the applicability of the Analytical Redundancy-based approach is well established by observers or specific integration schemes, examples of which can be found in [350]. The logical theory of model-based FDI was developed in the artificial intelligence community [77], and its connections with the structural analysis approach were further analysed in [70].

Distributed diagnosis schemes. When the implementation of a global diagnoser is not technically possible, distributed diagnosis schemes rest on assigning a part of the global task to each subsystem/agent [74, 229]. Under specific assumptions about the locally available models and data, the investigated problems range from distributed estimation [53, 81, 83] and the design of a coordination process [312, 315] to robustness with respect to network uncertainties [144, 262, 360, 378], or model uncertainties and non-linearities. Global system models are often assumed to be available [100, 411], or local models are used along with a real-time estimator of the interconnections [314], based on global or only locally sensed information [295].

Distributed control and fault-tolerant control. The main features underlying the control or fault-tolerant control of distributed dynamical systems or networks of dynamical agents are the (often unknown) interactions between subsystems/agents and the limited amount of information available to make their local decisions. Networks of dynamical agents are a wide application area: [137] addresses the synthesis of control laws via a sub-optimal algorithm, the agents coordination problem is studied in [404] and the problem of achieving a consensus under partial information is the subject of [312]. For control systems distributed over a network, a generic fault-tolerance strategy is proposed in [267]. Reference [260] analyses the fault accommodation problem under partially available information, while the reconfiguration of the information pattern was recently shown to allow fault tolerance under some conditions [338].

Operations research and mathematical tools. The basic tools of operations research (task allocation problem) and lattices that are used in this chapter can be found in [73, 382].

Part III
Discrete-Event Systems

Chapter 11
Fault Diagnosis of Discrete-Event Systems

Abstract This chapter presents diagnostic methods for discrete-event systems that are described by deterministic, nondeterministic or stochastic automata. Based on the solution to the state observation problem for discrete systems, the fault diagnostic problem is solved for all model classes by observing the unknown state of the model of the faulty systems and, hence, by deciding which model is currently consistent with the system behaviour.

11.1 Overview of Part III

This and the next chapters are devoted to discrete-event dynamical systems whose behaviour is described by sequences of discrete inputs and outputs. In contrast to the preceding chapters where the continuous changes of the signals occurring in the system have been investigated, the class of discrete-event systems is characterised by sequences of abrupt signal changes, because the signals have a finite value set.

Discrete-event systems occur naturally in the engineering practice. If the actuators like switches or valves can only jump between discrete positions, the input signal is binary and the signal values 0 and 1 are associated with the closed and the open position. Sensors may indicate that a physical quantity like the liquid level in a tank or a voltage exceeds a prescribed bound. Alarm sensors are typical examples for such sensors. Also the internal state of the system is often a discrete variable. For example, a robot gripper is empty or has grasped some part, a production step prescribed by a recipe has been carried out or not.

Whether or not a given dynamical system is considered as a discrete-event system depends upon the purpose of the investigations. It is typical for process supervision problems that a rather broad view on the system behaviour can be adopted which is based on a qualitative assessment of the signal values. If the supervisor should make a robot carry out a given sequence of movements or apply a certain recipe, then its decisions depend on discrete signal values like those mentioned in the examples and, hence, a discrete-event view-point is adequate. However, if signals have to remain in a narrow tolerance band, the continuous changes of these signals have to be considered and, thus, the continuous-systems point of view used in the preceding chapters

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8_11

is appropriate. The alternative considerations of the tank system as a continuous-system for level control or as a discrete system for carrying out a batch process demonstrates the dependence of the representation level of a dynamical system upon the task to be performed. Note that the terms "continuous" and "discrete" refer to the signal spaces and, hence, to the different description levels. If a distinction has to be made concerning the temporal behaviour, the notions of continuous-time systems and discrete-time systems will be used.

As the dynamical behaviour of discrete systems is described by the changes of the discrete signal values that are called *events*, such systems are said to be discrete-event systems. However, the behaviour of such systems can be equivalently represented by the sequence of discrete values that the input, the state and the output assume in the considered time interval. This notation will be used here, where the kth value of the input is denoted by $v(k)$ and the input sequence in the time horizon $k = 0 \ldots k_e$ by

$$V(0 \ldots k_e) = (v(0), v(1), \ldots, v(k_e)). \tag{11.1}$$

Similarly, the kth value of the output is denoted by $w(k)$ and the output sequence by

$$W(0 \ldots k_e) = (w(0), w(1), \ldots, w(k_e)). \tag{11.2}$$

Based on this information, the diagnostic system should identify which fault $f \in \mathcal{F}$ has occurred (Fig. 11.1). The result is, in general, a set $\mathcal{F}(k_e) \subseteq \mathcal{F}$ of faults rather than a unique element of the fault set $\mathcal{F}$. Note that for discrete systems also the faults are usually considered to be discrete phenomena with the symbol f denoting a single fault that may or may not occur in the system.

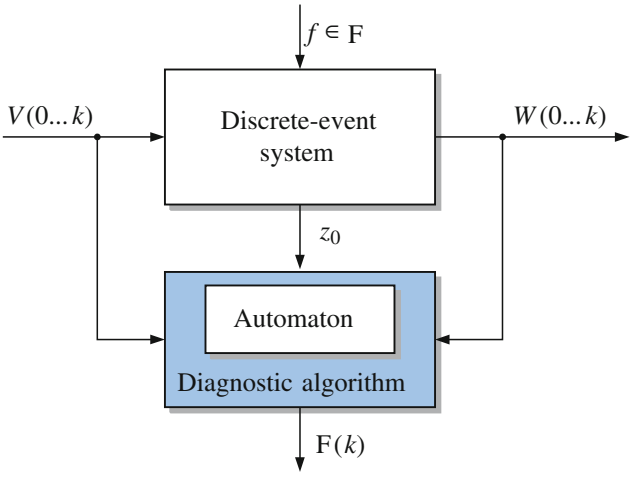


Fig. 11.1 Diagnostic problem

Consistency-based diagnosis of discrete-event systems. The main idea for solving the diagnostic problem is to test the consistency of the measured I/O pair with a set $\{\mathcal{A}_f \mid f \in \mathcal{F}\}$ of models that describe the system for all fault situations $f \in \mathcal{F}$ considered. This model will have the form of a deterministic, nondeterministic or stochastic automaton in this chapter. As in case of continuous-variable systems, the consistency of a pair $(V(0 \ldots k_e), W(0 \ldots k_e))$ of input and output sequences (11.1), (11.2) with finite time horizon k_e (in short: an I/O pair) is defined with respect to the behaviour $\mathcal{B}_f$ that the system subject to fault f possesses (cf. Sect. 1.3). To show the conceptual similarity of discrete-event and continuous-variable system diagnosis, assume that the model $\mathcal{A}_f$ of a discrete-event system is represented by a map ϕ_f that associates with each initial state z_0 and with every input sequence $V(0 \ldots k_e)$ the output sequence $W(0 \ldots k_e)$ of the system:

$$W(0 \ldots k_e) = \phi_f(z_0, V(0 \ldots k_e)). \tag{11.3}$$

The *behaviour* is the set of all I/O pairs (V, W) of arbitrary length $k_e \geq 0$ that are consistent with the model $\mathcal{A}_f$:

$$\mathcal{B}_f = \{(V(0 \ldots k_e), W(0 \ldots k_e)) \mid W(0 \ldots k_e) = \phi_f(z_0, V(0 \ldots k_e)); k_e \geq 0\}.$$

Diagnosis is based on the investigation to which behaviour $\mathcal{B}_f, (f \in \mathcal{F})$ the measured I/O pair belongs. An I/O pair is said to be *consistent with the model* $\mathcal{A}_f$ if it belongs to the behaviour $\mathcal{B}_f$:

$$(V(0 \ldots k_e), W(0 \ldots k_e)) \in \mathcal{B}_f. \tag{11.4}$$

In this relation, the left-hand side represents the measurement data and the right-hand side the behaviour of the system subject to fault f as it is represented by the model ϕ_f. If this relation holds, the fault f is called a *fault candidate*. The set of all fault candidates, which is denoted by $\mathcal{F}^*$, is the best possible diagnostic result:

$$\mathcal{F}^*(V(0 \ldots k_e), W(0 \ldots k_e)) = \{f \in \mathcal{F} \mid (V(0 \ldots k_e), W(0 \ldots k_e)) \in \mathcal{B}_f\}. \tag{11.5}$$

It depends upon the time horizon k_e and usually shrinks with increasing horizon because the more information about the system behaviour is obtained by measurements, the more faults can be excluded from the set of fault candidates.

The main problem of fault diagnosis of discrete-event systems is, hence, the elaboration of methods to test the consistency of I/O pairs with a model. This chapter develops consistency tests for systems that are described by deterministic, nondeterministic and stochastic automata.

Chapter overview. The theory of discrete-event systems has been developed rather separately with respect to the theory of continuous-variable systems. Therefore, the state of the art is different from what is known about continuous systems. The main theoretical results concern the modelling, analysis and supervisory control of discrete systems, but only a few results are available for diagnosis and fault-tolerant control.

This is the reason why Part III of this book concerns mainly the fault diagnostic problem and presents merely preliminary results on fault control reconfiguration.

In Sect. 11.2 models of discrete-event systems are introduced. Section 11.3 gives a survey of the solutions to the diagnostic problems, which will be dealt with in more detail in Sects. 11.4–11.7 for deterministic, nondeterministic and stochastic automata. The common aspect is given by the fact that all diagnostic methods are based on state observation methods for the corresponding automata, which will be explained first and later extended to fault detection and to fault identification.

11.2 Models of Discrete-Event Systems

11.2.1 Deterministic and Nondeterministic Systems

This section explains the basic dynamical properties of discrete-event systems and shows how such systems can be described. It extends the brief introduction to discrete-event modelling given in Sect. 3.6.

Discrete-valued signals. The discrete input, state and output of the system are denoted by the symbols v, z and w (Fig. 11.2) and the elements of their discrete value sets are enumerated as follows:

$$\begin{aligned} v &\in \mathcal{V} = \{1, 2, \ldots, M\} \\ z &\in \mathcal{Z} = \{1, 2, \ldots, N\} \\ w &\in \mathcal{W} = \{1, 2, \ldots, R\}. \end{aligned}$$

It is assumed that the numbers M, N or R are finite.

In order to use scalar signals, in many applications the physical variables have to be encoded. For example, if a tank system with 4 on/off valves is considered, the input can be represented by a 4-vector $\boldsymbol{v} = (v_1\ v_2\ v_3\ v_4)^{\mathrm{T}}$ whose ith component describes the position of the ith valve. As each component can assume either the value 0 or 1 in correspondence with the closed or the open position, $\boldsymbol{v}$ has one of 16 different values. These 16 values are represented in the following by a scalar input symbol v with the value set $\mathcal{V} = \{1, 2, \ldots, 16\}$. To do so, a mapping from the set of the 16 values of $\boldsymbol{v}$ onto the set $\mathcal{V}$ of the scalar input v has to be defined.

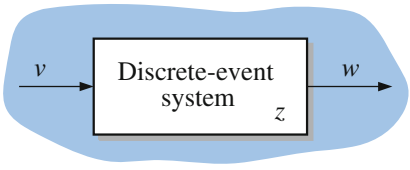


Fig. 11.2 Discrete-event system

Every change of the symbolic value of v, z or w is called an event. For example, if the state z jumps from the value j to the value i, a state event denoted by e_{ij} occurs (Fig. 11.3). However, the models introduced in this section use the sequences (11.1) and (11.2) of symbolic values rather then the sequences of events to characterise the behaviour of the discrete-event system under consideration.

Logical behaviour. The events occur at the time instants t_k which are enumerated as $k = 0, 1, 2, \ldots$ (Fig. 11.3). In the following, only the number k of the event is considered but not the actual time t_k. Therefore, the models used are called untimed or logical. They describe in which order the events occur but they say nothing about the temporal distance of these events. Besides the sequences (11.1) and (11.2) of inputs and outputs, the sequence of states

$$Z(0 \ldots k_e) = (z(0), z(1), z(2), \ldots, z(k_e))$$

is used to represent the logical behaviour of the system.

The motivation for using untimed models is twofold. First, the basic ideas of diagnosis and fault-tolerant control of discrete-event systems can be explained using untimed models, which are much simpler than timed models. Extensions of the methods to timed models are mentioned in the bibliographical notes. Second, in many practical circumstances, the untimed model yields the wanted results. For example, for many discrete-event systems faults can be detected due to the change of the order in which the events occur and no temporal information is necessary.

A further simplification is made in this chapter like in literature with respect to the synchronisation of the input, the state and the output. It is assumed that these events occur synchronously. That is, the state can only change if the input changes, which, at the same time, results in a change of the output. Repetitions of the symbols indicate that no real event occurs but a signal remains constant (Fig. 11.4). To shift the input, state and output events to the same time point is an abstraction that is reasonable for many applications.

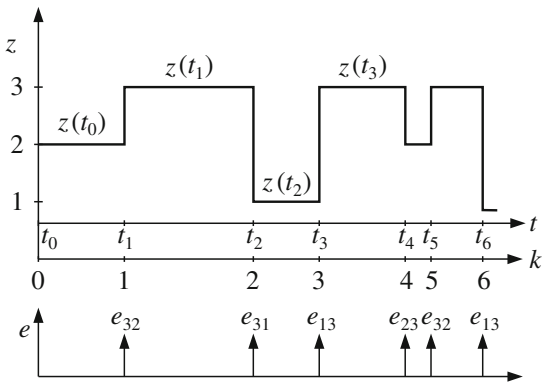


Fig. 11.3 Sequences of symbolic states and event sequences

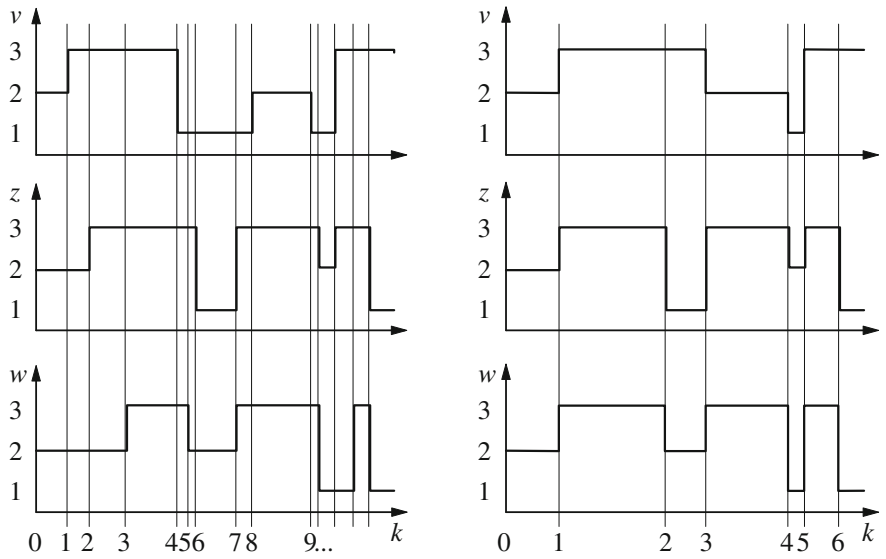


Fig. 11.4 Asynchronous (*left*) and synchronous (*right*) input, state and output sequences

Deterministic and nondeterministic systems. The assumed synchronous occurrence of the input, state and output events lead to the following "working principle" of discrete-event systems: At time $k = 0$, the system is in state $z(0) = z_0$ and obtains the input $v(0)$ (Fig. 11.2). The system generates the output $w(0)$ and its state jumps to the next value $z(1)$. Under the next input $v(1)$ the system changes its state from $z(1)$ to $z(2)$ and so on. In this way, for given initial state z_0 and input sequence $V(0 \ldots k_e)$ the discrete system follows a state sequence $Z(0 \ldots k_e)$ and generates an output sequence $W(0 \ldots k_e)$.

For deterministic systems, the generated state and output sequences Z and W are uniquely defined by z_0 and V. A standard form for describing deterministic systems is the automaton, which will be introduced in this section. It is a formalisation of the function ϕ, which has been used in Eq. (11.3) to represent a model of a deterministic discrete-event system.

For nondeterministic systems the state and output sequences are not unique but the system may generate any sequences of the sets $\mathcal{Z}(z_0, V)$ and $\mathcal{W}(z_0, V)$. This nondeterminism has to be understood in the following way. The technological system under consideration has a unique performance, because it cannot assume different states at the same time. Hence, for a fixed initial state and a fixed input sequence, an unambiguous state sequence Z and an unambiguous output sequence W occur. Nondeterministic behaviour occurs if the information about the system is not sufficient to predict these sequences unambiguously. If the system is brought into a particular initial state z_0 for several times and gets the same input sequence V, then the generated state and output sequences may differ. Hence, the model used to diagnose or control

this system has to describe some sets $\mathcal{Z}(z_0, V)$ and $\mathcal{W}(z_0, V)$ of possible sequences Z and W from which the real system "selects" one sequence. Such models have the form of nondeterministic automata, stochastic automata or Petri nets.

11.2.2 Deterministic Automata

This and the next section introduce models which can be used to describe the relation between the initial state $z(0)$ and the input sequence $V(0 \ldots k_e)$ on the one hand and the state sequence $Z(0 \ldots k_e)$ and output sequence $W(0 \ldots k_e)$ that the discrete system generates on the other hand.

The deterministic input–output automaton (I/O automaton) is defined by a 6-tuple

$$\mathcal{A} = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, G, H, z_0)$$

with

- $\mathcal{Z}$ - set of states,
- $\mathcal{V}$ - set of input values (also called the input alphabet),
- $\mathcal{W}$ - set of output values (also called the output alphabet),
- $G : \mathcal{Z} \times \mathcal{V} \to \mathcal{Z}$ - state transition function,
- $H : \mathcal{Z} \times \mathcal{V} \to \mathcal{W}$ - output function,
- z_0 - initial state.

The dynamics of the automaton are described by the functions G and H in the following recursive way:

$$z(k+1) = G(z(k), v(k)), \quad z(0) = z_0 \tag{11.6}$$

$$w(k) = H(z(k), v(k)). \tag{11.7}$$

For the initial state z_0 and the input sequence $V(0 \ldots k_e)$, these functions yield the state sequence $Z(0 \ldots k_e + 1)$ and output sequence $W(0 \ldots k_e)$. If in the later investigations the functions G and H should be analysed, the time counter k does not matter and the equations above are written shorter as

$$z' = G(z, v) \tag{11.8}$$

$$w = H(z, v), \tag{11.9}$$

where z' denotes the "next state" following the state z.

The diagnostic problem may be considered for situations where the initial state z_0 is known or unknown. The initial state is known, for example, if the system to be diagnosed performs a cyclic function and the diagnostic system can be invoked whenever the system moves through its intial state. Also in the start-up phase of a system, the initial state is usually known and one says that the automaton is *initialised*.

Knowing the initial state z_0 considerably simplifies the diagnosis. If z_0 is unknown, a state observation problem is included in the diagnostic problem, which makes the overall problem much more involved. Then it is assumed that a set $\mathcal{Z}_0$ of possible initial states is known (with $\mathcal{Z}_0 = \mathcal{Z}$ as the trivial assumption). Both situations will be considered in this chapter.

Automaton graph. A nice graphical interpretation of an automaton is the *automaton graph*, whose vertices depict the states $z \in \mathcal{Z}$ and whose edges show how the state of the automaton can change (Fig. 11.5). Every directed edge represents a state transition described by Eq. (11.8) together with the output that is generated according to Eq. (11.9). In part (b) of the figure, the state transitions belonging to the input $v = 2$ are drawn by dashed arrows to illustrate the influence of the input upon the state transitions. The initial state is marked by an arrow not emerging from any other vertex. For a given initial state z_0 and input sequence V, the dynamical behaviour of the automaton is represented by the path through the automaton graph that starts in the vertex z_0 and whose edges are associated with the input prescribed by V.

The automaton is deterministic because the state and the input unambiguously determine the edge to follow in the automaton graph and, hence, the successor state and output. For example, if the automaton in Fig. 11.5 is in state 3, depending on the input $v \in \{1, 2\}$ the automaton goes either towards state 4 or towards 5. In this example, in both cases it generates the output 2.

Automaton map and behaviour. The *automaton map*

$$\phi : \mathcal{Z} \times \mathcal{V}^* \to \mathcal{W}^*$$

associates with each initial state $z_0 \in \mathcal{Z}$ and input sequence $V(0 \ldots k_e) \in \mathcal{V}^*$ the output sequence $W(0 \ldots k_e) \in \mathcal{W}^*$ of the automaton, which is obtained by applying Eqs. (11.46), (11.47) for $z(0) = z_0$:

$$W(0 \ldots k_e) = \phi(z_0, V(0 \ldots k_e)). \tag{11.10}$$

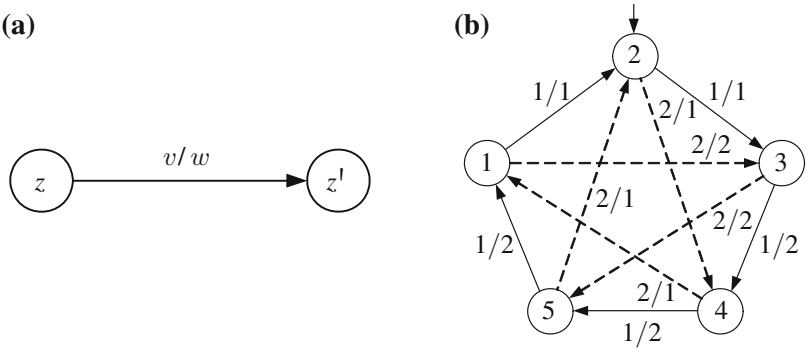


Fig. 11.5 Automaton graph of a deterministic automaton

$\mathcal{V}^*$ and $\mathcal{W}^*$ denote the sets of arbitrary sequences $V(0 \ldots k_e)$ or $W(0 \ldots k_e)$ of any length k_e that can be built using the input or output symbols of $\mathcal{V}$ or $\mathcal{W}$, respectively. The automaton map shows that an I/O pair (V, W) is consistent with an automaton if and only if the relation (11.10) holds.

The *behaviour* of an initialised automaton $(\mathcal{A}, z_0)$ is the set of all I/O pairs (V, W) that can be generated by the automaton map ϕ:

$$\mathcal{B} = \{(V(0 \ldots k_e), W(0 \ldots k_e)) \mid W(0 \ldots k_e) = \phi(z_0, V(0 \ldots k_e)); k_e \geq 0\}.$$

In applications, usually not the automaton map ϕ, but the functions G and H are used to carry out the consistency test included in the diagnostic problem. Therefore, it is important to represent the behaviour $\mathcal{B}$ in terms of G and H. Obviously, the relation (11.10) holds if there exists a state sequence

$$Z(0 \ldots k_e) = (z(0), z(1), \ldots, z(k_e))$$

that satisfies for the given input sequence $V(0 \ldots k_e)$ the relation (11.6) and for which the outputs generated by Eq. (11.7) coincides with the output sequence $W(0 \ldots k_e)$:

$$\mathcal{B} = \{(V(0 \ldots k_e), W(0 \ldots k_e)) \mid \exists Z(0 \ldots k_e) : \quad z(k+1) = G(z(k), v(k)), \\ w(k) = H(z(k), v(k))\}. \qquad (11.11)$$

11.2.3 Nondeterministic Automata

In the nondeterministic automaton

$$\mathcal{N} = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, L_n, \mathcal{Z}_0)$$

the functions G and H occurring in the deterministic automaton are replaced by the function

$$L_n : \mathcal{Z} \times \mathcal{W} \times \mathcal{Z} \times \mathcal{V} \rightarrow \{0, 1\}$$

that describes for every state z and input v which successor state z' can be assumed while generating the output w. This function represents the behaviour of the automaton in terms of all 4-tuples (z', w, z, v) that are consistent with the automaton and form the set

$$\mathcal{R}(L_n) = \{(z', w, z, v) : L_n(z', w, z, v) = 1\} \subseteq \mathcal{Z} \times \mathcal{W} \times \mathcal{Z} \times \mathcal{V}. \qquad (11.12)$$

As the set (11.12) is, mathematically, a *relation*, the function L_n is called *behavioural relation* of the nondeterministic automaton.

For nondeterministic automata, the initial state is usually assumed to belong to a set $\mathcal{Z}_0$ of possible initial states. However, in some particular situations, the initial state z_0 may be unambiguously known.

The description of the dynamical behaviour of a nondeterministic automaton differs from that of a deterministic automaton given by Eqs. (11.6) and (11.7). Instead of a unique successor state z' and output w, the behavioural relation L_n yields, for the current state $z(k)$ and input $v(k)$, the following sets of possible successor states and output values:

$$\mathcal{Z}'(z(k), v(k)) = \{z' : \exists w \in \mathcal{W} \text{ such that } L_\mathrm{n}(z', w, z(k), v(k)) = 1\} \quad (11.13)$$

$$\mathcal{W}(z(k), v(k)) = \{w : \exists z' \in \mathcal{Z} \text{ such that } L_\mathrm{n}(z', w, z(k), v(k)) = 1\}. \quad (11.14)$$

Figure 11.6 depicts a part of the automaton graph of a nondeterministic automaton. It shows that for the input $v = 1$, the state may change from 1 towards 2 or towards 10, whereas for $v = 2$ only the state transition $1 \to 2$ can occur. The state change from 1 to 10 may either cause the output $w = 1$ or the output $w = 2$. Examples for the sets defined in Eqs. (11.13) and (11.14) are the following:

$$\mathcal{Z}'(1, 1) = \{2, 10\}$$
$$\mathcal{W}(1, 2) = \{2\}.$$

If the behavioural relation L_n associates with each pair z, v a unique successor state z' and output w it can be represented by a state transition function G and an output function H and the nondeterministic automaton becomes deterministic.

The behaviour $\mathcal{B}$ of the nondeterministic automaton can be represented in terms of the behavioural relation l_n as follows. An I/O pair belongs to $\mathcal{B}$ if there exists a state sequence

$$Z(0 \ldots k_\mathrm{e} + 1) = (z(0), z(1), \ldots, z(k_\mathrm{e} + 1))$$

that satisfies for the input sequence $V(0 \ldots k_\mathrm{e})$ and the output sequence $W(0 \ldots k_\mathrm{e})$ the function L_n:

$$L_\mathrm{n}(z(k + 1), w(k), z(k), v(k)) = 1, \quad k = 0, 1, \ldots, k_\mathrm{e}.$$

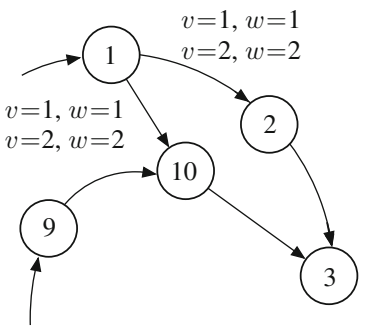


Fig. 11.6 Part of the automaton graph of a nondeterministic automaton

Hence, $\mathcal{B}$ can be represented as

$$\mathcal{B} = \{(V(0\ldots k_e), W(0\ldots k_e)) \mid \exists Z(0\ldots k_e+1) : L_n(z(k+1), w(k), z(k), v(k)) = 1,\ k = 0, \ldots, k_e\}. \tag{11.15}$$

Markov property. Both the deterministic and the nondeterministic automaton possess an important property, which is referred to as the *Markov property* of these models. This property means that the successor state $z(k+1)$ depends *only* upon the current state $z(k)$ and the current input $v(k)$ but it does not depend on the whole state sequence $Z(0\ldots k)$ or the whole input sequence $V(0\ldots k)$ that the automaton has generated or obtained until time k. The Markov property makes it possible to find the recursive relations (11.6) and (11.13) both of which are a relation between the next state $z(k+1)$ and the current state $z(k)$ and input $v(k)$ only. In case of the nondeterministic automaton, this relation does not determine the future state unambiguously, but it fixes the set of future states. Note that the set $\mathcal{Z}'(z(k), v(k))$ given in Eq. (11.13) depends merely upon $z(k)$ and $v(k)$.

In applications, the question whether or not the system under investigation possesses the Markov property depends upon the definition of the state z. Roughly speaking, if the state z includes all the information about the signals up to time k which is necessary to determine the future behaviour of the system, then the future state can be represented only in terms of the current state $z(k)$ and there is no need to refer to earlier states or input values occurring in the sequences $Z(0\ldots k-1)$ or $V(0\ldots k-1)$. To define the state appropriately is an important modelling step.

11.2.4 Stochastic Automata

The stochastic automata introduced in this section extend the description of nondeterministic discrete-event systems in such a way that the frequency of the occurrence of the different events can be assessed. They provide very useful additional information, because nondeterministic systems often produce a large set of different state and output sequences, but in practice these sets do by no means occur with the same frequency. In particular, fault diagnosis has to deal with nominal state sequences that occur (hopefully) with a very large frequency but the models have to include faulty sequences that a system follows seldom. The model should provide information about the frequency of occurrence.

The following explains how nondeterministic automata can be associated with the probabilities with which the different sequences occur. First, the notion of a stochastic process has to be introduced.

Stochastic processes. A stochastic process is a nondeterministic system for which the state and output sequences are generated with a certain probability. Its nondeterminism is not only considered in terms of the sets $\mathcal{Z}'(z_0, V)$ and $\mathcal{W}(z_0, V)$ defined

in Eqs. (11.13) and (11.14) but also in terms of the frequency with which the different elements of these sets are generated. Some of them may occur very often whereas others occur rarely.

Capital letters V, Z and W are used to denote the random variables of the input, state and output of the stochastic process. They are stochastic variables, that is, variables whose values are determined by chance. In every experiment, these variables assume values from the sets $\mathcal{V}$, $\mathcal{Z}$ or $\mathcal{W}$, respectively (Fig. 11.7). As the ranges of these variables and the time k are discrete, the process is called more precisely a *discrete stochastic process*.

The stochastic automaton should describe the probability with which the system assumes at time k the state $z \in \mathcal{Z}$ and generates the output $w \in \mathcal{W}$

$$\text{Prob}(Z(k) = z), \quad \text{Prob}(W(k) = w)$$

or with which the system follows a state trajectory $Z(0 \ldots k_e)$. To do so, it has to represent the generation law underlying the stochastic process, which is given by the transition probability

$$\text{Prob}(Z(k+1) = z(k+1), W(k) = w(k) \mid Z(k) = z(k), V(k) = v(k)).$$

Representation of stochastic processes by stochastic automata. Stochastic processes with finite sets of input values, output values and states are represented by finite-state stochastic automata

$$\mathcal{S} = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, L, p_0) \tag{11.16}$$

with $\mathcal{Z}$, $\mathcal{V}$ and $\mathcal{W}$ defined as before and

- $L : \mathcal{Z} \times \mathcal{W} \times \mathcal{Z} \times \mathcal{V} \longrightarrow [0, 1]$ - state transition probability (behavioural relation)
- p_0 - initial state probability distribution.

p_0 is the set of the N probability values $\text{Prob}(Z(0) = z_0)$ for the N possible initial states $z_0 \in \mathcal{Z}$. Hence, the automaton may start its behaviour in any state of the set

$$\mathcal{Z}_0 = \{z_0 \in \mathcal{Z} : p_0(z_0) > 0\}.$$

For each state $z \in \mathcal{Z}_0$ it is known with which probability $\text{Prob}(Z(0) = z) = p_0$ this state occurs as initial state of the system under investigation.

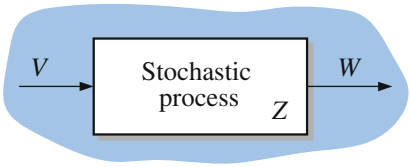


Fig. 11.7 Stochastic process

The function L represents the transition probability

$$L(z', w, z, v) = \text{Prob}(Z(1)=z',\ W(0)=w \mid Z(0)=z,\ V(0)=v). \tag{11.17}$$

In extension of the terminology used for the nondeterministic automaton, the function L is called *behavioural relation* of the stochastic automaton. In order to indicate that the behavioural relation is a conditional probability distribution, the symbol $L(z', w \mid z, v)$ is used instead of $L(z', w, z, v)$.

The probability distribution has the properties

$$\begin{aligned} 0 \le L(z', w \mid z, v) &\le 1, \quad \forall z', z \in \mathcal{Z},\ v \in \mathcal{V},\ w \in \mathcal{W} \\ \sum_{z' \in \mathcal{Z}} \sum_{w \in \mathcal{W}} L(z', w \mid z, v) &= 1, \quad \forall z \in \mathcal{Z},\ v \in \mathcal{V} \end{aligned} \tag{11.18}$$

and it leads to the two boundary distributions

$$G(z' \mid z, v) = \sum_{w \in \mathcal{W}} L(z', w \mid z, v) \tag{11.19}$$

$$H(w \mid z, v) = \sum_{z' \in \mathcal{Z}} L(z', w \mid z, v). \tag{11.20}$$

$G(z'|z, v)$ is called the *state transition relation* and $H(w|z, v)$ the *output relation* of the stochastic automaton. Due to Eq. (11.17) these relations represent the conditional probability distributions

$$G(z' \mid z, v) = \text{Prob}(Z(1)=z' \mid Z(0)=z, V(0)=v) \tag{11.21}$$

$$H(w \mid z, v) = \text{Prob}(W(0)=w \mid Z(0)=z, V(0)=v) \tag{11.22}$$

and possess the properties

$$\sum_{z' \in \mathcal{Z}} G(z' \mid z, v) = 1 \tag{11.23}$$

$$\sum_{w \in \mathcal{W}} H(w \mid z, v) = 1 \tag{11.24}$$

for all $z \in \mathcal{Z}$ and all $v \in \mathcal{V}$. Note that the functions G and H defined in Eqs. (11.21) and (11.22) include less information than the behavioural relation L because L also reflects the stochastic dependence between z' and w. Therefore, the following investigations refer to L rather than G and H. G is useful for problems in which only the state sequence is considered and the output sequence ignored.

Stochastic automata for which the behavioural relation L can be reconstructed from G and H are called stochastic *Mealy automata*. For them the relation

$$L(z', w \mid z, v) = G(z' \mid z, v) \cdot H(w \mid z, v)$$

holds for all $z, z' \in \mathcal{Z}$, $v \in \mathcal{V}$ and $w \in \mathcal{W}$. For these automata, the conditional probability distributions G and H defined in Eqs. (11.21) and (11.22) replace the state transition function G and the output function H of the deterministic automaton used in Eqs. (11.8) and (11.9).

The behaviour $\mathcal{B}$ of the stochastic automaton is the set of all I/O pairs for which a state sequence

$$Z(0 \ldots k_e + 1) = (z(0), z(1), \ldots, z(k_e + 1))$$

exists with positive probability such that the behavioural relation L is satisfied for all k in the time horizon considered:

$$L(z(k+1), w(k) \,|\, z(k), v(k)) > 0, \quad k = 0, 1, \ldots, k_e.$$

Hence, $\mathcal{B}$ can be represented as follows:

$$\begin{aligned} \mathcal{B} = \{&(V(0 \ldots k_e), W(0 \ldots k_e)) \,| \\ &\exists Z(0 \ldots k_e + 1) : L(z(k+1), w(k), z(k), v(k)) > 0,\ k = 0, \ldots, k_e\}. \end{aligned} \tag{11.25}$$

Autonomous stochastic automaton. If the automaton has no input, it is called an autonomous automaton. If, furthermore, the output coincides with the state, this automaton is given by the triple

$$\mathcal{S}_a = (\mathcal{Z}, G, p_0),$$

where G denotes the state transition relation given by Eq. (11.21) after neglecting the input v:

$$G(z' \mid z) = \text{Prob}(Z(1) = z' \mid Z(0) = z).$$

Graph of stochastic automata. The automaton graph is a directed graph, whose vertices denote the states and whose edges denote the possible state transitions. Figure 11.8 gives an example. The edges are associated with the state transition probability given by the value of the behavioural relation L for the pair of states connected by the edges and for the input v obtained and the output w generated for this state transition. The edge from state 4 towards state 5 shows the abbreviated labels, where 1/2/1 means that the state transition occurs for the input $v = 1$ while generating the output $w = 2$ with probability 1.

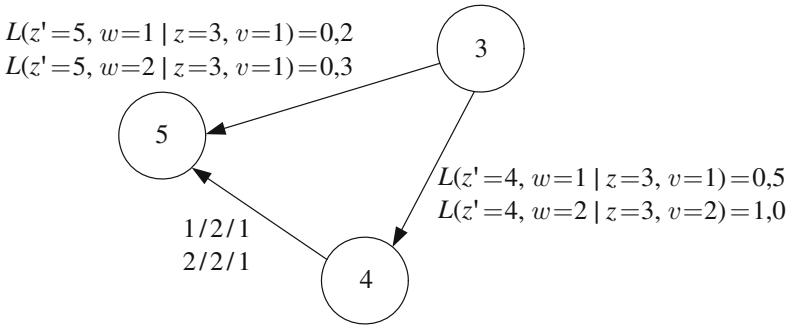


Fig. 11.8 Autonomous stochastic automaton

Beginning in any state z_0 with $\text{Prob}(Z(0) = z_0) > 0$, the automaton moves along the directed edges according to the corresponding probabilities. If more than one edge starts in a state, then all these edges can be followed which results in alternative state and output sequences. The frequencies with which the automaton follows these edges are described by the transition probabilities.

Example 11.1 Properties of stochastic automata
In the example shown in Fig. 11.9, the automaton may step from state 1 towards state 2 or state 10 if it obtains input $v = 1$, but it is known to step towards state 2 if the input $v = 2$ is applied. Moreover, the automaton may produce either output $w = 1$ or $w = 2$. The behavioural relation says that the probability to step from state 1 towards state 10 when getting the input $v = 1$ is 0.2 if this step is associated with the output $w = 1$ and 0.3 if $w = 2$ occurs. The sum of 0.5 of both values is the probability that the automaton steps from 1 to 10 under the input $v = 1$ while producing *some* output. Hence, $G(10 \mid 1, 1) = 0.5$ holds. Alternatively, the automaton may step from 1 towards 2. It definitely does this step if it obtains input $v = 2$ and it is known to produce output $w = 2$ during this step. If the automaton gets input $v = 1$, then the probability to move to state 2 while generating output $w = 1$ is 0.5.

The property (11.18) of the behavioural relation is satisfied, because for $z = 1$ and $v = 1$ the example yields

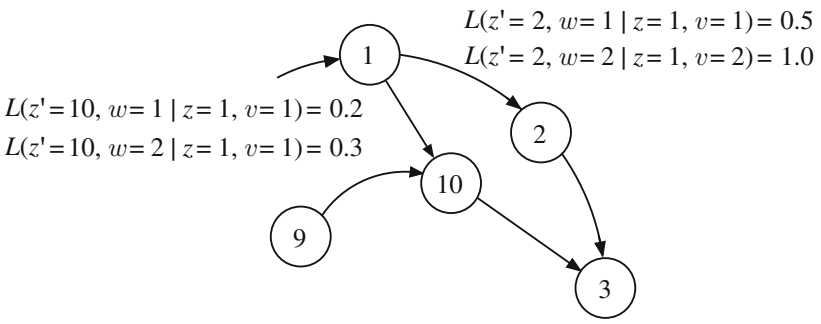


Fig. 11.9 Part of the automaton graph of a stochastic automaton with input and output

$$\sum_{z'}\sum_{w} L(z', w \mid z = 1, v = 1) =$$
$$= L(z'=10, w=1 \mid z=1, v=1) + L(z'=10, w=2 \mid z=1, v=1) +$$
$$+ L(z'=2, w=1 \mid z=1, v=1)$$
$$= 0.2 + 0.3 + 0.5 = 1$$

and for $z = 1$ and $v = 2$

$$\sum_{z' \in \mathcal{Z}} \sum_{w \in \mathcal{W}} L(z', w \mid z = 1, v = 2) \; = \; L(z'=2, w=2 \mid z = 1, v = 2) \; = \; 1.$$

The state transition relation G defined in Eq. (11.19) ignores the output and considers merely the transition between the states in dependence upon the input. For the example, the following relations hold

$$\begin{aligned} G(z'=10 \mid z=1, v=1) &= 0.2 + 0.3 = 0.5 \\ G(z'=2 \mid z=1, v=1) &= 0.5 \\ G(z'=2 \mid z=1, v=2) &= 1. \end{aligned}$$

They say that for the input $v=1$ the automaton goes from state 1 to state 2 or to state 10 with probability 0.5, but if it obtains input $v=2$ the automaton is known to go towards state 2.

The output relation H defined in Eq. (11.20) says which output is produced independently of the next state that is assumed by the automaton. For the example, the output relation has the values

$$\begin{aligned} H(w=1 \mid z=1, v=1) &= 0.7 \\ H(w=2 \mid z=1, v=1) &= 0.3 \\ H(w=2 \mid z=1, v=2) &= 1, \end{aligned}$$

which means that the automaton is known to produce the output $w = 2$ if it obtains the input $v = 2$ in state 1, but for the input $v = 1$ it may generate the output $w = 1$ with probability 0.7 and $w = 2$ with probability 0.3. Note that there is in general no way to reconstruct L from given G and H as mentioned above. □

Prediction. Stochastic automata can be used to predict the behaviour of a discrete-event system when starting from some state

$$z(0) = \mathcal{Z}_0 = \{z \in \mathcal{Z} \; : \; p_0(z) > 0\}$$

and getting the input sequence

$$V(0 \ldots k_e) = (v_0, v_1, \ldots, v_{k_e}).$$

For the initial state, the probability distribution is given by $p_0(z)$:

$$\text{Prob}(Z(0) = z) = p_0(z).$$

If the first input symbol v_0 has been obtained, the stochastic automaton carries out a state transition $z_0 \xrightarrow{v_0} z_1$ with $z_0 \in \mathcal{Z}_0$ according to the state transition probability $G(z_1 \mid z_0)$:

$$\text{Prob}(Z(1) = z_1 \mid v(0)) = \sum_{z_0 \in \mathcal{Z}_0} G(z_1 \mid z_0) \cdot \text{Prob}(Z(0) = z_0).$$

More generally, after the input symbols up to time k_e have been received, the automaton is in the state $z_{k_\text{e}+1}$ with the probability

$$\begin{aligned} &\text{Prob}(Z(k_\text{e}+1) = z_{k_\text{e}+1} \mid V(0 \ldots k_\text{e})) \\ &\quad = \sum_{z_{k_\text{e}} \in \mathcal{Z}} G(z_{k_\text{e}+1} \mid z_{k_\text{e}}) \cdot \text{Prob}(Z(k_\text{e}) = z_{k_\text{e}} \mid V(0 \ldots k_\text{e}-1)). \end{aligned} \quad (11.26)$$

Markov property. Stochastic automata describe discrete-event systems only if several assumptions are satisfied, which are summarised now.

A discrete stochastic process is defined by the probability with which a certain state change appears and an output symbol occurs at time k for a given sequence of input symbols. In general, for the state $z(k+1)$ and the output $w(k)$ this probability depends on the sequence of states and the sequence of input symbols up to time k and is thus described by the conditional probability

$$\text{Prob}\left(\begin{array}{c|l} Z(k+1)=z(k+1), & Z(0)=z(0), \ldots, Z(k)=z(k), \\ W(k)=w(k) & V(0)=v(0), \ldots, V(k)=v(k) \end{array} \right).$$

In the following, only those stochastic processes are considered that possess the *Markov property*. For such processes, the relation

$$\begin{aligned} &\text{Prob}(Z(k+1)=z(k+1), W(k)=w(k) \mid Z(k)=z(k), V(k)=v(k)) \\ &= \text{Prob}\left(\begin{array}{c|l} Z(k+1)=z(k+1), & Z(0)=z(0), \ldots, Z(k)=z(k), \\ W(k)=w(k) & V(0)=v(0), \ldots, V(k)=v(k) \end{array} \right) \end{aligned} \quad (11.27)$$

holds for all k, $z(k+1), z(k), \ldots, z(0)$, $w(k), w(k-1), \ldots, w(0)$ and $v(k), v(k-1), \ldots, v(0)$. It is common to say that $z(k+1)$ and $w(k)$ are *conditionally independent of the past variables* for given $z(k)$ and $v(k)$. The consequence of this assumption is the fact that the model of the system has only to include information of all single-state transitions, which in the stochastic automaton is represented by the behavioural relation L.

If the Markov property were not valid for a system, the model has to represent relations over a longer time interval; for example, the information about the next state if the system came into the current state 1 from the precessor state 5 or from predecessor state 4.

Furthermore, it is assumed that the process is homogeneous which means that the transition probability does not explicitly depend on the time variable k. Whenever the 4-tuple of successor state z', output w, current state z and input v is considered, the transition probability is the same. Hence, the relation

$$\begin{aligned}&\text{Prob}(Z(k+1)=z',\, W(k)=w \mid Z(k)=z,\, V(k)=v)\\ &\quad=\text{Prob}(Z(1)=z',\, W(0)=w \mid Z(0)=z,\, V(0)=v)\end{aligned} \tag{11.28}$$

holds for all k. A discrete stochastic process whose generation law is described by the state transition probability (11.28) is called *homogenous Markov process* with input and output.

For describing the stochastic process by a stochastic automaton, it is assumed furthermore that the appearance of a certain input symbol at time k is independent of the states and of the input values that have occurred up to that time, and independent of the time k. Therefore, the relation

$$\begin{aligned}&\text{Prob}\left(V(k)=v \;\middle|\; \begin{array}{l} Z(0)=z(0),\ldots,Z(k)=z(k),\\ V(0)=v(0),\ldots,V(k-1)=v(k-1)\end{array}\right)\\ &\quad=\text{Prob}(V=v)\end{aligned} \tag{11.29}$$

holds, where $\text{Prob}(V=v)$ describes the probability with which the input symbol v occurs. This assumption is not satisfied, for example, if the input $v(k)$ is prescribed by a supervisor that acts in dependence upon the measured output $w(k)$. Then this "feedback" needs to be included into the stochastic description of the process in order to satisfy the assumption (11.29) which will be used in the following.

11.2.5 Model of the Faulty System

This section explains how the system representation by automata can be extended to include information about the occurrence of a fault and the effect that the fault has on the future behaviour of the system. In literature, two main ideas have been followed in the past, both of which will be compared now.

Both ideas start from the interpretation of a fault as an *unobservable event* f. The attribute "unobservable" means that the fault event does not directly generate a measurement event that indicates this fault. The diagnostic method to be elaborated should find the fault from the changes that this fault causes in the future state or output sequences.

The first idea, which will be used later in this book, emphasises the fact that in many technological applications a fault changes the dynamics of the system under consideration. That means that the behaviour before the occurrence and after the occurrence of a fault f distinguishes from one another and, hence, has to be described by

different models. For example, if in a batch process a valve is blocked, the behaviour of the process changes qualitatively because the effect of the fault may influence the behaviour of the process at any future time.

This situation is depicted in Fig. 11.10. Part (a) of the figure shows two models, which describe the system for the fault cases f_0 and f_1, where as usual f_0 indicates the faultfree situation. For shortage of notation, the I/O pairs that are represented by the labels of the edges of the automaton graph are abbreviated here by the symbols a, b, c and d. As long as the system is faultless, it generates I/O pairs that are repetitions of (a, b, c, d). Hence, the left model, which is denoted by $\mathcal{A}_0$, is valid. After the fault has occurred, the system behaviour is decribed by the right model, which is denoted by $\mathcal{A}_f$ and shows that the future I/O pairs are represented by repetitions of (b, a).

For this kind of faults, the diagnostic problem concerns the question which of the two models represents the current behaviour of the system. Stated again in the sense of consistency-based diagnosis, the diagnostic problem leads to the consistency check for the measured I/O pair with respect to a set of models that is denoted by $\{\mathcal{A}_f, f \in \mathcal{F}\}$. The appearance of a fault is represented by the unobservable event, which changes the valid model from $\mathcal{A}_0$ to $\mathcal{A}_f$.

The alternative approach is depicted in Fig. 11.11. The fault is again considered as an unobservable event, but now such events occur as additional state transitions in the model of the faultless system. Consequently, after the occurrence of the fault the system behaves as before the fault. This situation is typical, for example, for computer or control systems, where a fault event represents an erroneous modification of a data set. The application of algorithms denoted by the labels a, b, c and d in the automaton graph does not change, but their sequence is modified by the fault. Before and after the fault f, the same algorithms are applied, possibly in a temporarily changed order.

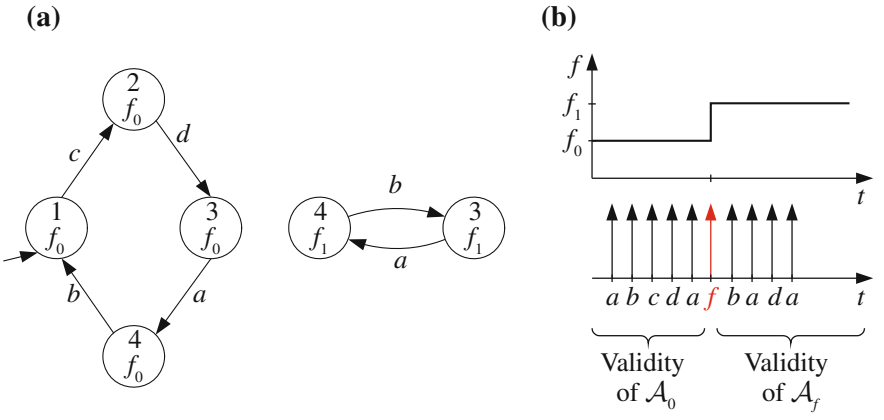


Fig. 11.10 Faults change the system properties

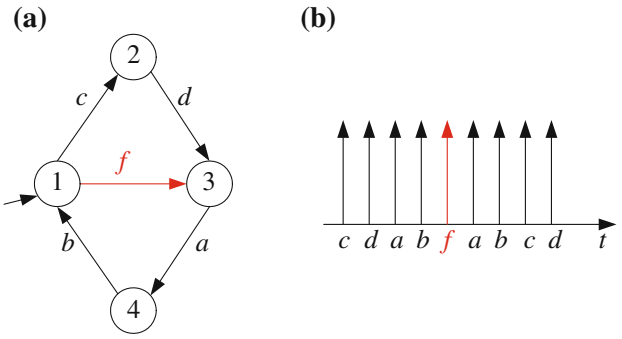


Fig. 11.11 Fault interpreted as an unobservable event

In this situation, the fault has again to be found by comparing the measured I/O pair with the model of the system, but the methods to do so differ from the methods to be developed here because of the different assumptions on the faulty system behaviour.

Both approaches to include faulty events into the model of a system are similar, but have been followed by different groups in literature. The similarity becomes obvious if the fault event shown by the arrow in Fig. 11.11a between the state 1 and state 3 is introduced as a state transition between a state of the model $\mathcal{A}_0$ in Fig. 11.10a and a state of the model $\mathcal{A}_f$. The difference of both approaches lies in the fact that in the first approach the fault brings the state into another "region" of the state set of the automaton that usually has a completely different structure than the state set of the faultless behaviour.

Diagnosis as a model-selection problem. This and the next chapters follow the first approach of fault modelling, in which the fault changes the dynamics and, hence, the structure of the automaton graph of a system. Accordingly, the system is described by a set of models, which is denoted by $\{\mathcal{A}_f,\ f \in \mathcal{F}\}$, where $\mathcal{F}$ is the set of faults considered. Every model has its own behavioural relation L_f, where the index f indicates the fault case. For this modelling approach, the diagnostic problem can be stated as the problem to select the model of the current hehavior out of the model set, (Fig. 11.12). If this model is valid for the fault f, then f is considered as a solution to the diagnostic problem.

As the fault f changes, the dynamics of the system, the state transition function or the behavioural relation of the model used depends upon f. If deterministic automata are used to represent the system, the model set $\{\mathcal{A}_f,\ f \in \mathcal{F}\}$ consists of deterministic automata

$$\mathcal{A}_f = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, G_f, H_f, z_0),$$

where the state transition function G_f and the output function H_f depend upon the fault f. The other components $\mathcal{Z}$, $\mathcal{V}$, $\mathcal{W}$ and z_0 may also depend upon f, but to simplify notation, it is assumed that these elements are the same for all models.

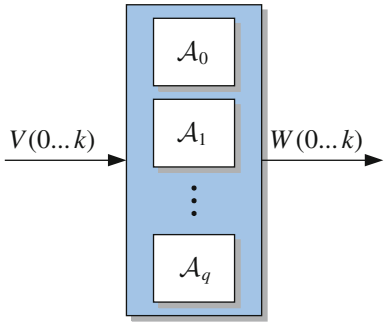


Fig. 11.12 Fault identification as model identification problem

If nondeterministic automata

$$\mathcal{N}_f = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, L_{\mathrm{n}f}, z_0)$$

are used to represent the system, the behavioural relation L_{n} depends upon f and these relations of different models are distinguished by the additional index f. For stochastic automata

$$\mathcal{S}_f = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, L_f, p_0(z))$$

the state transition relation L changes with the occurrence of the fault f. The fault appears as an additional element in the conditional probability distribution:

$$L_f = \mathrm{Prob}(Z(1) = z', W(0) = w \mid Z(0) = z, V(0) = v, F = f),$$

where F denotes the stochastic variable for the fault. In all three cases, the indices f indicate that the corresponding functions depend upon the fault f considered.

All the given models can be used if the fault f does not change over time. This assumption will be adopted in most parts of this chapter to elaborate the main ideas. It does not mean that the system must be faulty from the beginning, but that the models are set-up for the situation that the fault does not change during the online application of the diagnostic algorithm. Of course, the algorithms developed in this chapter can be used for changing faults, but then not all properties that are proved for them are still valid.

Changing faults. If the diagnostic problem should be considered for changing faults, the fault has to be interpreted as an external signal $f(k)$ that is described by the sequence

$$F(0 \ldots k_{\mathrm{e}}) = (f(0), f(1), \ldots, f(k_{\mathrm{e}})). \tag{11.30}$$

The models have to be extended to cope with the additional input (Fig. 11.13). In the deterministic case, the functions G and H have now two input arguments such that Eqs. (11.6) and (11.7) have to be replaced by

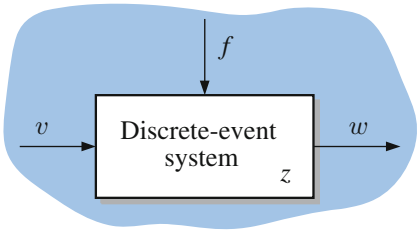


Fig. 11.13 Fault interpreted as an additional input

$$z(k+1) = G(z(k), v(k), f(k)), \quad z(0) = z_0 \tag{11.31}$$

$$w(k) = H(z(k), v(k), f(k)). \tag{11.32}$$

For the nondeterministic automaton, the behavioural relation now describes the behaviour of the system with the additional argument $f(k)$ as follows:

$$L_{\mathrm{n}}(z(k+1), w(k), z(k), v(k), f(k)) = 1.$$

For stochastic automata, in the state transition probability distribution

$$L = \mathrm{Prob}(Z(1) = z', W(0) = w \mid Z(0) = z, V(0) = v, F(k) = f)$$

the fault is now a time-dependent variable.

Fault model. It is usually too general to assume that the fault may follow an arbitrary sequence (11.30), because this would mean that the fault may change with every event occurring. If the fault is considered as an external signal, then the kind of signals to be considered has to be restricted by a *fault model*. Its usefulness should be explained here for the stochastic automaton, but the same idea is applicable for other forms of discrete-event models too.

The fault model

$$\mathcal{S}_{\mathrm{f}} = (\mathcal{F}, G_{\mathrm{f}}, p_{f0}) \tag{11.33}$$

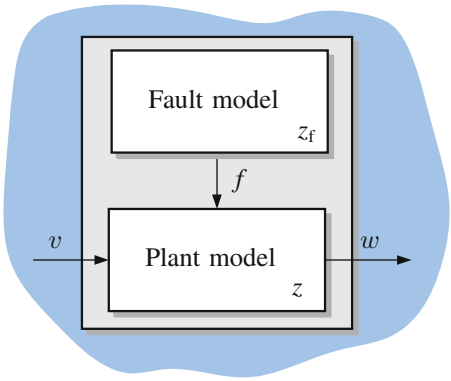


Fig. 11.14 Representation of a faulty system including a fault model

says how the fault $f(k)$ can behave. Hence, it describes a generator of the additional input to the plant model as depicted in Fig. 11.14. Its state set corresponds to the fault set $\mathcal{F}$ and the function $G_\mathrm{f} : \mathcal{F} \times \mathcal{F} \longrightarrow [0, 1]$ describes the conditional probability that the fault changes from f towards f' within one time step:

$$G_\mathrm{f}(f' \mid f) = \mathrm{Prob}(F(1) = f' \mid F(0) = f). \tag{11.34}$$

If the time steps are equidistant (like in discrete-time systems), these probabilities are closely related to the well-known measure of mean time before failure. p_{f0} denotes the probability distribution over the initial fault set.

The combination of the plant model with the fault model depicted in Fig. 11.14 results in a stochastic automaton

$$\tilde{\mathcal{S}} = (\tilde{\mathcal{Z}}, \mathcal{V}, \mathcal{W}, \tilde{L}, \tilde{p}_0) \tag{11.35}$$

whose state set is given by

$$\tilde{\mathcal{Z}} = \mathcal{Z} \times \mathcal{F} \tag{11.36}$$

and whose behavioural relation $\tilde{L}$ is obtained from L and G_f according to

$$\tilde{L}(z', f', w \mid z, f, v) = L(z', w \mid z, f, v) \cdot G_\mathrm{f}(f' \mid f) \tag{11.37}$$

with $z, z' \in \mathcal{Z}$, $v \in \mathcal{V}$, $w \in \mathcal{W}$ and $f, f' \in \mathcal{F}$. If the elements $\tilde{z} \in \tilde{\mathcal{Z}}$ are written as the vector

$$\tilde{z} = \begin{pmatrix} z \\ f \end{pmatrix}, \tag{11.38}$$

the behavioural relation $\tilde{L}$ in Eq. (11.37) can be rewritten as

$$\begin{aligned} &\tilde{L}(\tilde{z}', w \mid \tilde{z}, v) = \mathrm{Prob}(\tilde{Z}(1) = \tilde{z}', W(0) = w \mid \tilde{Z}(0) = \tilde{z}, V(0) = v) \\ &= \mathrm{Prob}\left(\begin{pmatrix} Z(1) \\ F(1) \end{pmatrix} = \begin{pmatrix} z' \\ f' \end{pmatrix}, W(0) = w \;\middle|\; \begin{pmatrix} Z(0) \\ F(0) \end{pmatrix} = \begin{pmatrix} z \\ f \end{pmatrix}, V(0) = v \right) \end{aligned}$$

which gives $\tilde{L}$ the standard form of the behavioural relation of stochastic automaton.

11.3 Diagnostic Problems and Ways of Solution

This sections gives a survey of the diagnostic problems for discrete-event systems and ways of solutions that will be explained in more detail in the remaining part of this chapter.

The diagnostic problem can be stated in a general form as follows (Fig. 11.1):

Diagnostic problem for discrete-event systems

Given:	Model set $\mathcal{A}_f$, $(f \in \mathcal{F})$
	Input sequence $V(0 \ldots k_e)$
	Output sequence $W(0 \ldots k_e)$
Find:	Set of fault candidates.

As the set of fault candidates has been defined with respect to the behaviour $\mathcal{B}_f$ of the model set $\mathcal{A}_f$, $(f \in \mathcal{F})$ and for the three models introduced in Sect. 11.2, the behaviour has been defined in a uniform way by Eqs. (11.11), (11.15) or (11.25), respectively, the solution of the diagnostic problem can be found by checking the consistency of the measured I/O pair with the model set used.

As the diagnostic problem should be solved usually online with increasing time horizon $k_e = 0, 1, \ldots$, an important aspect of all methods developed in this and the following chapter lies in the fact that these methods are formulated in a recursive way in which they use the diagnostic result up to the time horizon k_e to find the result for the extended time horizon $k_e + 1$. As the intermediate result, the last state $z(k_e)$ or a set $\mathcal{Z}(k_e)$ of possible states has to be stored. These intermediate results include all information about the input sequence $V(0 \ldots k_e)$ and the output sequence $W(0 \ldots k_e)$ processed so far that is relevant for the future diagnostic result. Upon arrival of the next measured values $(v(k_e + 1), w(k_e + 1))$ the diagnostic unit checks which models of the set $\{\mathcal{A}_f,\ f \in \mathcal{F}\}$ are still consistent with the extended sequences $V(0 \ldots k_e + 1)$ and $W(0 \ldots k_e + 1)$ and updates the set of fault candidates $\mathcal{F}^*(k_e + 1)$ accordingly.

The main ideas of the diagnostic methods will be explained for constant faults f, but the bibliographical notes include references, in which the extension for time-varying faults are given.

The difficulty of solving the diagnostic problem depends upon the kind of models used and the online information included in the I/O pair. The following classification starts with the simplest problem and shows the increase in complexity of the diagnostic problem if the model becomes more involved.

Diagnosis of deterministic automata with state measurements. In the simplest case, the models $\mathcal{A}_f$, $(f \in \mathcal{F})$ are deterministic automata and the current state $z(k)$ is measurable, which means that the output $w(k)$ coincides with the model-state $z(k)$. Under this strong assumptions, the consistency test of the I/O pair with a model $\mathcal{A}_f$ reduces to check whether the last measured state transition $z(k_e) \rightarrow z(k_e + 1)$ can occur in the model $\mathcal{A}_f$:

$$z(k_e + 1) = G_f(z(k_e), v(k_e)). \tag{11.39}$$

In this relation, the three signal values $z(k_e)$, $v(k_e)$ and $z(k_e + 1)$ are known and it is checked whether these symbols satisfy the state transition function G_f of the automaton $\mathcal{A}_f$.

To formulate the diagnostic method for all classes of systems considered in this chapter in a uniform way, the result of the consistency test is described by the binary variable $p_f(k_e)$, where $p_f(k_e) = 1$ indicates that the I/O pair $(V(0 \dots k_e), W(0 \dots k_e))$ is consistent with the model $\mathcal{A}_f$ describing the system subject to the fault f, whereas $p_f(k_e) = 0$ shows that the I/O pair is not consistent with this model. Consequently, the set of fault candidates is given by

$$\mathcal{F}^*(k_e) = \{f \in \mathcal{F} : p_f(k_e) = 1\}. \tag{11.40}$$

Diagnosis of deterministic automata with output measurements. The diagnostic problem becomes more involved if the initial state z_0 of the system is unknown and if instead of the state z some output w is measured. Then the diagnostic problem includes a state observation problem.

The diagnostic method starts with the assumption that the initial state z_0 belongs to a given set $\mathcal{Z}_0$. After the first I/O pair $(v(0), w(0))$ is known, it is tested for which initial states $z_0 \in \mathcal{Z}_0$ there exists a state transition $z_0 \xrightarrow{v(0)/w(0)} z(1)$ to some state $z(1)$ that is consistent with the state transition function G_f and the output function H_f of the automaton $\mathcal{A}_f$. What "consistent" means can be seen in the automaton graph, in which an edge from z_0 towards $z(1)$ with the label $v(0)/w(0)$ has to exist. Formulated as equations, "consistency" means that the relations

$$\begin{aligned} z(1) &= G_f(z_0, v(0)) \\ w(0) &= H_f(z_0, v(0)) \end{aligned}$$

are valid.

To represent this method in a recursive way, two sets of states are introduced:

$$\begin{aligned} \mathcal{Z}_f(0 \mid -1) &= \mathcal{Z}_0 \\ \mathcal{Z}_f(0 \mid 0) &= \{z_0 \in \mathcal{Z}_0 : w(0) = H_f(z_0, v(0))\}. \end{aligned}$$

The set $\mathcal{Z}_f(0 \mid -1)$ represents the a-priori information about the initial state, which has to be given as input $\mathcal{Z}_0$ to the diagnostic algorithm. After the I/O pair $v(0), w(0))$ is known, the set $\mathcal{Z}_f(0 \mid 0)$ is determined, which includes all elements of $\mathcal{Z}_0$ for which the output $H_f(z_0, v(0))$ generated by the automaton coincides with the measured output $w(0)$. Furthermore the set

$$\mathcal{Z}_f(1 \mid 0) = \{z(1) = G_f(z_0, v(0)) : z_0 \in \mathcal{Z}_f(0 \mid 0)\}$$

can be determined, which includes all states to which the model $\mathcal{A}_f$ can move under the input $v(0)$.

The set $\mathcal{Z}_f(1 \mid 0)$ is used as the starting point of the second recursion step. After the I/O pair $(v(1), w(1))$ has been measured, the two sets

$$\mathcal{Z}_f(1\,|\,1) = \{z(1) \in \mathcal{Z}_f(1\,|\,0) : w(1) = H_f(z(1), v(1))\}$$
$$\mathcal{Z}_f(1\,|\,0) = \{z(2) = G_f(z(1), v(1)) : z(1) \in \mathcal{Z}_f(1\,|\,1)\}$$

are determined in a similar way as the sets $\mathcal{Z}_f(0\,|\,0)$ and $\mathcal{Z}_f(1\,|\,0)$. In general, for each time horizon k_e the following two sets are generated alternately after the I/O pair $(v(k_\mathrm{e}), w(k_\mathrm{e}))$ has been measured:

$$\mathcal{Z}_f(k_\mathrm{e}\,|\,k_\mathrm{e}) = \{z(k_\mathrm{e}) \in \mathcal{Z}_f(k_\mathrm{e}\,|\,k_\mathrm{e}-1) : w(k_\mathrm{e}) = H_f(z(k_\mathrm{e}), v(k_\mathrm{e}))\} \tag{11.41}$$
$$\mathcal{Z}_f(k_\mathrm{e}+1\,|\,k_\mathrm{e}) = \{z(k_\mathrm{e}+1) = G_f(z(k_\mathrm{e}), v(k_\mathrm{e})) : z(k_\mathrm{e}) \in \mathcal{Z}_f(k_\mathrm{e}\,|\,k_\mathrm{e})\}. \tag{11.42}$$

As long as both sets are not empty, there exists a state sequence for the automaton $\mathcal{A}_f$ such that the automaton generates for the measured input sequence the measured output sequence and, consequently, the I/O pair is consistent with the model $\mathcal{A}_f$. Hence, the indicator $p_f(k_\mathrm{e})$ has now to be determined as

$$p_f(k_\mathrm{e}) = \begin{cases} 1 \text{ if } \mathcal{Z}_f(k_\mathrm{e}+1\,|\,k_\mathrm{e}) \neq \emptyset \\ 0 \text{ else.} \end{cases} \tag{11.43}$$

After this indicator is known for the current time horizon k_e for all models of the set $\{\mathcal{A}_f,\ f \in \mathcal{F}\}$, the set of fault candidates is obtained by Eq. (11.40).

Diagnosis of nondeterministic automata. For systems described by nondeterministic automata $\mathcal{N}_f$, $(f \in \mathcal{F})$ the diagnostic problem includes always a state observation problem, because even if the initial state z_0 is unambiguously known, the nondeterminism of the automaton can produce ambiguity with respect to the current state in each state transition. The complexity of the diagnostic problem increases due to the nondeterminism that allows the model to associate with each state z and input v a set of successor states z' rather than a unique state.

For the known I/O pair $(v(k_\mathrm{e}), w(k_\mathrm{e}))$, the sets $\mathcal{Z}_f(k_\mathrm{e}\,|\,k_\mathrm{e})$ and $\mathcal{Z}_f(k_\mathrm{e}+1\,|\,k_\mathrm{e})$ have to be determined using the behavioural relation $L_{\mathrm{n}f}$ of the nondeterministic automaton $\mathcal{N}_f$ according to the relations

$$\mathcal{Z}_f(k_\mathrm{e}\,|\,k_\mathrm{e}) = \{z(k_\mathrm{e}) \in \mathcal{Z}_f(k_\mathrm{e}\,|\,k_\mathrm{e}-1) : \exists z(k_\mathrm{e}+1) : L_{\mathrm{n}f}(z(k_\mathrm{e}+1), w(k_\mathrm{e}), z(k_\mathrm{e}), v(k_\mathrm{e})) = 1\}$$
$$\mathcal{Z}_f(k_\mathrm{e}+1\,|\,k_\mathrm{e}) = \{z(k_\mathrm{e}+1) : \exists z(k_\mathrm{e}) \in \mathcal{Z}_f(k_\mathrm{e}\,|\,k_\mathrm{e}) : L_{\mathrm{n}f}(z(k_\mathrm{e}+1), w(k_\mathrm{e}), z(k_\mathrm{e}), v(k_\mathrm{e})) = 1\}.$$

Then the indicator $p_f(k_e)$ can be found by Eq. (11.43) and the set of fault candidates by Eq. (11.40).

Diagnosis of stochastic automata. If the system is described by stochastic automata $\mathcal{S}_f$, $(f \in \mathcal{F})$, the ambiguities of the diagnostic result can be reduced by associating with each element of the set of fault candidates the probability

$$p_f(k_e) = \text{Prob}(f \mid V(0 \ldots k_e), W(0 \ldots k_e)),$$

which replaces the binary indicator $p_f(k_e)$ of consistency used for the deterministic or nondeterministic automaton. For the set of fault candidates the relation (11.40) is extended to become

$$\mathcal{F}^*(k_e) = \{f \in \mathcal{F} : p_f(k_e) > 0\}, \tag{11.44}$$

but besides this set the value of $p_f(k_e)$ remains important for the interpretation of the diagnostic result. This indicator shows with which certainty each fault $f \in \mathcal{F}^*(k_e)$ is present.

The basis for determining the probability $p_f(k_e)$ is again the solution of the state observation problem, which now means to calculate the probability

$$\text{Prob}(Z(k_e) = z \mid V(0 \ldots k_e), W(0 \ldots k_e), f)$$

that the system subject to fault f can be in state z after it has obtained the input sequence $V(0 \ldots k_e)$ and generated the output sequence $W(0 \ldots k_e)$. This probability is abbreviated as $p_f(z, k_e)$:

$$p_f(z, k_e) = \text{Prob}(Z(k_e) = z \mid V(0 \ldots k_e), W(0 \ldots k_e), f).$$

Similarly, the probability to be in the state z' at time $k_e + 1$ after having received the input sequence $V(0 \ldots k_e)$ and generated the output sequence $W(0 \ldots k_e)$ is denoted by

$$p'_f(z', k_e) = \text{Prob}(Z(k_e + 1) = z' \mid V(0 \ldots k_e), W(0 \ldots k_e), f).$$

In Sect. 11.6 a recursive representation of these two probabilities will be given to complete the consistency test and to make it possible to determine the set of fault candidates together with the mentioned probabilities.

Outline of the diagnostic methods. The diagnostic problem will be solved for the three classes of automata in the next sections in a uniform way. This survey has shown that the diagnostic problem includes a state observation problem unless the state is measurable. Therefore, the state observation problem is solved first and later extended to fault detection and fault identification.

11.4 Diagnosis of Deterministic Automata

11.4.1 Diagnostic Algorithm

As the diagnostic problem is very simple if the state z is measurable (cf. Eq. (11.39)), this section is devoted to deterministic automata with outputs w. A pair $(V(0\ldots k_e), W(0\ldots k_e))$ of input and output sequences with finite time horizon k_e is called *consistent* with the deterministic automaton $\mathcal{A}_f$ if there exists a state sequence

$$Z(0\ldots k_e+1) = (z(0), z(1), \ldots, z(k_e+1)) \tag{11.45}$$

for which the relations

$$z(k+1) = G(z(k), v(k)), \quad z(0) = z_0 \tag{11.46}$$

$$w(k) = H(z(k), v(k)) \tag{11.47}$$

are satisfied for $k = 0, 1, \ldots, k_e$. If the initial state z_0 is known, the consistency can be tested by simply applying Eq. (11.46) for $k = 0, 1, \ldots, k_e$ to generate for the measured input sequence (11.1) the state sequence (11.45). Then Eq. (11.47) is used to test whether the outputs $w(k)$ obtained by the model are identical to the elements of the measured output sequence (11.2). If the initial state is only known to belong to a set $\mathcal{Z}_0$, then the sequences of state sets described in Eqs. (11.42) and (11.43) have to be generated and consistency means that none of them is empty.

The following algorithm summarises the diagnostic steps to be performed for deterministic automata. The algorithm starts for $k_e = 0$ and considers an increasing time horizon. The newest measured I/O pair $(v(k_e), w(k_e))$ is denoted by $(\bar{v}, \bar{w})$ in the algorithm and the state sets $\mathcal{Z}_f(k_e \,|\, k_e)$ and $\mathcal{Z}_f(k_e+1 \,|\, k_e)$ by $\mathcal{Z}_f$ or $\mathcal{Z}'_f$, respectively, for $f \in \mathcal{F}$.

Algorithm 11.1 *Diagnosis of deterministic automata*

Given: Deterministic automata $\mathcal{A}_f$, $(f \in \mathcal{F})$
Set of initial states $\mathcal{Z}_0$

Initialisation: $\mathcal{Z}'_f = \mathcal{Z}_0$ for all $f \in \mathcal{F}$
$k_e = 0$

Loop: 1. Measure the next I/O pair $(\bar{v}, \bar{w})$.

2. Determine $\mathcal{Z}_f = \{z \in \mathcal{Z}'_f \,:\, \bar{w} = H_f(z, \bar{v})\}$ for all $f \in \mathcal{F}$.

3. Determine $\mathcal{Z}'_f = \{z' = G_f(z, \bar{v}) \,:\, z \in \mathcal{Z}_f\}$ for all $f \in \mathcal{F}$.

4. Set $p_f(k_e) = 1$ if $\mathcal{Z}'_f \neq \emptyset$ and $p_f(k_e) = 0$ otherwise for all $f \in \mathcal{F}$.

5. Determine $\mathcal{F}^*(k_e) = \{f \in \mathcal{F} : p_f(k_e) = 1\}$.
6. $k_e := k_e + 1$
   Continue with Step 1.

Result: Set of fault candidates $\mathcal{F}^*(k_e)$ for increasing time horizon k_e

This algorithm can be applied to a large number of different models $\mathcal{A}_f$ and to I/O sequences of arbitrary length k_e because its complexity is only linear with respect to the number of models and the length of the sequences. The algorithm gives the best possible result, because it determines the set of fault candidates. For every element of the set $\mathcal{F}^*(k_e)$, the I/O pair is consistent with the corresponding model $\mathcal{A}_f$ up to time k_e and there is no fault in the remaining set $\mathcal{F}\backslash\mathcal{F}^*(k_e)$ for which the I/O pair is consistent with the model $\mathcal{A}_f$.

The diagnostic result has the following consequences. If f_0 does not belong to $\mathcal{F}^*(k_e)$, a fault is detected with certainty. For every time horizon k_e, the system may be subject to any fault $f \in \mathcal{F}^*(k_e)$.

Diagnosability of deterministic automata. An important question asks under what conditions the diagnostic method developed so far is able to detect a fault (fault detectability) or to unambiguously identify the fault (fault identifiability). These two properties will be investigated in the following paragraphs. As a basis for these investigations, the next section reviews known results on the equivalence of states of deterministic I/O automata from [211].

11.4.2 Results on Deterministic Automata with Equivalent States

Equivalence of states. This section considers automata with arbitrary initial states z_0. It compares the behaviour of a single automaton if this automaton starts its movement in two different initial states or the behaviour of two automata with different initial states.

Definition 11.1 (*Equivalent states*) Two states z and $\tilde{z}$ of the automaton $\mathcal{A} = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, G, H)$ are said to be equivalent if the relation

$$\phi(z, V) = \phi(\tilde{z}, V) \tag{11.48}$$

holds for all $V \in \mathcal{V}^*$. Otherwise they are called *distinguishable*.

Equivalence of the states z and $\tilde{z}$ is indicated by $z \sim \tilde{z}$. An automaton that has no equivalent states is called *minimal*.

State equivalence of two automata. Definition 11.1 can be applied to states of two distinct automata

$$\mathcal{A}_i = (\mathcal{Z}_i, \mathcal{V}, \mathcal{W}, G_i, H_i) \quad \text{and} \quad \mathcal{A}_j = (\mathcal{Z}_j, \mathcal{V}, \mathcal{W}, G_j, H_j).$$

Analogously to Eq. (11.48), the states $z \in \mathcal{Z}_i$ and $\tilde{z} \in \mathcal{Z}_j$ are said to be equivalent if the relation

$$\phi_i(z, V) = \phi_j(\tilde{z}, V) \tag{11.49}$$

is satisfied for all $V \in \mathcal{V}^*$. Equation (11.49) can be written in the form (11.48), in which the same automaton map ϕ occurs on both sides, if the automata $\mathcal{A}_i$ and $\mathcal{A}_j$ are lumped together to get the automaton

$$\bar{\mathcal{A}} = (\bar{\mathcal{Z}}, \mathcal{V}, \mathcal{W}, \bar{G}, \bar{H})$$

with

$$\bar{\mathcal{Z}} = \mathcal{Z}_i \cup \mathcal{Z}_j$$

$$\bar{G}(z, v) = \begin{cases} G_i(z, v) & \text{if } z \in \mathcal{Z}_i \\ G_j(z, v) & \text{if } z \in \mathcal{Z}_j \end{cases} \qquad \bar{H}(z, v) = \begin{cases} H_i(z, v) & \text{if } z \in \mathcal{Z}_i \\ H_j(z, v) & \text{if } z \in \mathcal{Z}_j. \end{cases}$$

Denote the automaton map of $\bar{\mathcal{A}}$ by $\bar{\phi}$. Then the states $z \in \mathcal{Z}_i$ and $\tilde{z} \in \mathcal{Z}_j$ of the two automata $\mathcal{A}_i$ and $\mathcal{A}_j$ are equivalent if the relation

$$\bar{\phi}(z, V) = \bar{\phi}(\tilde{z}, V) \tag{11.50}$$

holds. Consequently, the equivalence test described in the next paragraph for a single automaton can also be applied for testing the equivalence of states of two automata.

Equivalence test. The equivalence test uses the following recursive equivalence definition:

- Two states z and $\tilde{z}$ are said to be 0-equivalent ($z \overset{0}{\sim} \tilde{z}$), if Eq. (11.48) holds for a single input symbol $V = v$. This is true if and only if the relation

  $$H(z, v) = H(\tilde{z}, v) \quad \text{for all} \quad v \in \mathcal{V}$$

  is satisfied. Accordingly, the state set $\mathcal{Z}$ is partitioned into sets $\mathcal{Z}_i^0$, ($i = 1, 2, \ldots, q_0$) such that a state pair $(z, \tilde{z})$ belongs to the same set $\mathcal{Z}_i^0$ if and only if the automaton produces for every input symbol the same output for both states:

$$z, \tilde{z} \in \mathcal{Z}_i^0 \Longleftrightarrow H(z, v) = H(\tilde{z}, v) \quad \text{for all} \quad v \in \mathcal{V}. \tag{11.51}$$

The function

$$H^* : \ \mathcal{Z}^0 \times \mathcal{V} \to \mathcal{W}$$

with

$$\mathcal{Z}^0 = \{\mathcal{Z}_i^0 \mid i = 1, 2, \ldots, q_0\}$$

is introduced to associate with each state set $\mathcal{Z}_i^0$ and input $v \in \mathcal{V}$ the output $w = H(z, v)$ such that Eq. (11.51) holds:

$$H^*(\mathcal{Z}_i^0, v) = H(z, v) \quad \text{for all} \quad z \in \mathcal{Z}_i^0, \ v \in \mathcal{V}. \tag{11.52}$$

States belonging to two different sets $\mathcal{Z}_i^0$ and $\mathcal{Z}_j^0$, $(i \neq j)$ are 0-distinguishable.

- For $k \geq 0$ two states z and $\tilde{z}$ are said to be $(k+1)$-equivalent ($z \overset{k+1}{\sim} \tilde{z}$), if Eq. (11.48) holds for all input sequences $V(0 \ldots l)$ with time horizon $l \leq k+1$. This is true if and only if the successor states $z' = G(z, v)$ and $\tilde{z}' = G(\tilde{z}, v)$ of z or $\tilde{z}$, respectively, are k-equivalent:

$$G(z, v) \overset{k}{\sim} G(\tilde{z}, v) \quad \text{for all} \quad v \in \mathcal{V}.$$

Hence, the kth partition of $\mathcal{Z}$ into the sets $\mathcal{Z}_i^k$, $i = 1, 2, \ldots, q_k$ is refined to get the $(k+1)$st partition of $\mathcal{Z}$ into the sets $\mathcal{Z}_i^{k+1}$, $(i = 1, 2, \ldots, q_{k+1})$ such that the relation

$$z, \tilde{z} \in \mathcal{Z}_i^{k+1} \Longleftrightarrow \forall v \in \mathcal{V} \ \exists j : G(z, v), G(\tilde{z}, v) \in \mathcal{Z}_j^k \tag{11.53}$$

holds. The function

$$G_k^* : \ \mathcal{Z}^{k+1} \times \mathcal{V} \to \mathcal{Z}^k$$

with

$$\begin{aligned} \mathcal{Z}^k &= \{\mathcal{Z}_i^k \mid i = 1, 2, \ldots, q_k\} \\ \mathcal{Z}^{k+1} &= \{\mathcal{Z}_i^{k+1} \mid i = 1, 2, \ldots, q_{k+1}\} \end{aligned}$$

associates with each state set $\mathcal{Z}_i^{k+1}$ and each input $v \in \mathcal{V}$ the set $\mathcal{Z}_j^k$ such that Eq. (11.53) holds. Hence, the relation

$$G(z, v) \in G_k^*(\mathcal{Z}_i^{k+1}, v) \quad \text{for all} \quad z \in \mathcal{Z}_i^{k+1}, \ v \in \mathcal{V} \tag{11.54}$$

is valid.

If states are $(N-1)$-equivalent with $N = |\mathcal{Z}|$, they are equivalent according to Definition 11.1. Then the final result of the recursive state partitioning is obtained

and denoted by symbols $\mathcal{Z}_i$ without superscript:

$$\mathcal{Z} = \cup_{i=1}^{q} \mathcal{Z}_i . \tag{11.55}$$

The mapping G^*_{N-2} is also denoted by G^*. Usually, the refinement of the state partitioning finishes before the $(N-1)$st refinement step.

If the states z and $\tilde{z}$ are not k-equivalent, they are called k-distinguishable.

Lemma 11.1 (Uniqueness of the state set partition) [211] *The state set partition (11.55) is unique. Two states $z, \tilde{z}$ belong to the same set $\mathcal{Z}_i$ if and only if they are equivalent.*

The lemma implies that the states $z, \tilde{z}$ are k-equivalent if and only if they belong to the same set $\mathcal{Z}_i^k$ of the kth state partition. Otherwise, they are k-distinguishable.

This test is summarised in the following algorithm:

Algorithm 11.2 *Partitioning of the state set into sets of equivalent states*

Given: Deterministic automaton $\mathcal{A}$

1. Determine the partition $\mathcal{Z}^0$ such that Eq. (11.51) is satisfied.
2. For $k = 0, 1, \ldots, N-2$, determine the partition $\mathcal{Z}^{k+1}$ such that Eq. (11.53) holds
   If $\mathcal{Z}^{k+1} = \mathcal{Z}^k$ holds, finish this step.
3. Denote the final partition obtained by $\mathcal{Z}_i$, $(i = 1, 2, \ldots, q)$.

Result: Partition (11.55) of the state set into sets $\mathcal{Z}_i$ of equivalent states.

This algorithm has the complexity $O(N^2)$, where N is the cardinality of $\mathcal{Z}$.

Properties of automata with equivalent states. Consider the state sequences that an automaton can generate starting in a pair $(z, \tilde{z})$ of equivalent states. An important fact

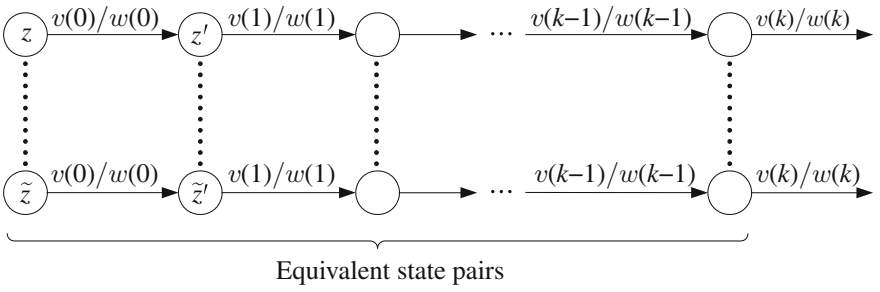


Fig. 11.15 State trajectories over equivalent state pairs

is that for all input sequences $V(0 \ldots k_e)$ with arbitrary time horizon k_e these two state sequences only go over equivalent state pairs $(z, \tilde{z})$, $(z', \tilde{z}')$ etc. (Fig. 11.15, cf. [128], Theorem 3.3). All state transitions involved have the same I/O pair $(v(k), w(k))$, $(k = 0, 1, \ldots, k_e)$.

For k-distinguishable state pairs $(z, \tilde{z})$, there exists an input sequence $V(0 \ldots k)$ for which the output sequences that are generated by the automaton starting in the initial state z or $\tilde{z}$, respectively, are not equal:

$$\phi(z, V(0 \ldots k)) \neq \phi(\tilde{z}, V(0 \ldots k)). \tag{11.56}$$

The input sequence $V(0 \ldots k)$ for which the relation (11.56) holds is called a *distinguishing input sequence* of the state pair $(z, \tilde{z})$.

If the states z and $\tilde{z}$ are distinguishable, distinguishing input sequences have at most $N - 1$ symbols. Hence, for distinguishable states the output sequences are identical for at most $N-2$ symbols and the fact that the states are distinguishable can be identified, for a reasonably chosen input sequence, in finite time. Consequently, all further investigations can be restricted to finite input sequences $V(0 \ldots k_e)$ with $k_e < N - 1$.

If z and $\tilde{z}$ are $(k-1)$-equivalent but k-distinguishable, then the output sequences W are identical up to the element $w(k-1)$ and there exists a distinguishing input sequence $\bar{V}(0 \ldots k)$ such that the output sequences are not completely identical:

$$\begin{aligned} &\phi(z, V) = \phi(\tilde{z}, V) \quad \text{for all} \quad V \in \mathcal{V}^l, \; l = 0, 1, \ldots, k \\ &\exists\, \bar{V} \in \mathcal{V}^{k+1} : \; \phi(z, \bar{V}) \neq \phi(\tilde{z}, \bar{V}). \end{aligned} \tag{11.57}$$

An important fact is that the output sequences

$$\begin{aligned} W(0 \ldots k) &= \phi(z, \bar{V}(0 \ldots k)) = (w(0), w(1), \ldots, w(k)) \\ \tilde{W}(0 \ldots k) &= \phi(\tilde{z}, \bar{V}(0 \ldots k)) = (\tilde{w}(0), \tilde{w}(1), \ldots, \tilde{w}(k)) \end{aligned}$$

differ only in in the last element $w(k)$:

$$\begin{aligned} w(0) &= \tilde{w}(0) \\ w(1) &= \tilde{w}(1) \\ &\vdots \\ w(k-1) &= \tilde{w}(k-1) \\ w(k) &\neq \tilde{w}(k). \end{aligned}$$

Therefore, the state trajectories starting in the states z and $\tilde{z}$ go over state pairs with decreasing equivalence properties (Fig. 11.16).

This fact has a direct interpretation. When starting in the two initial states z or $\tilde{z}$ the first k elements $v(0), \ldots, v(k-1)$ of the input sequence $\bar{V}(0 \ldots k)$ are used to

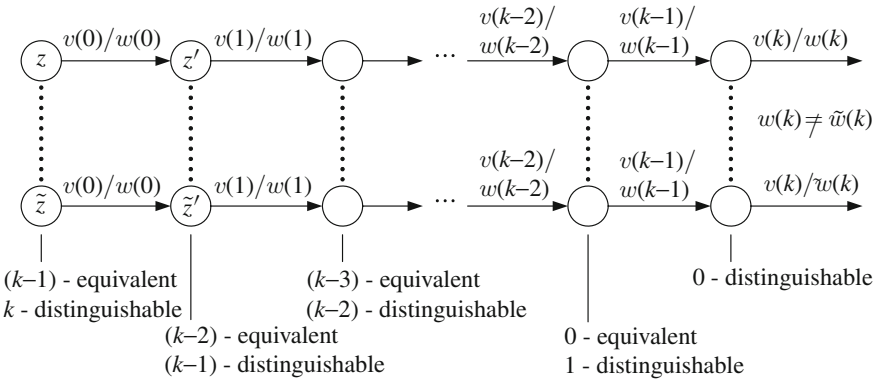


Fig. 11.16 State trajectories generated by a distinguishing input sequence $\bar{V}(0 \ldots k)$ that start in a k-distinguishable, $(k-1)$-equivalent state pair

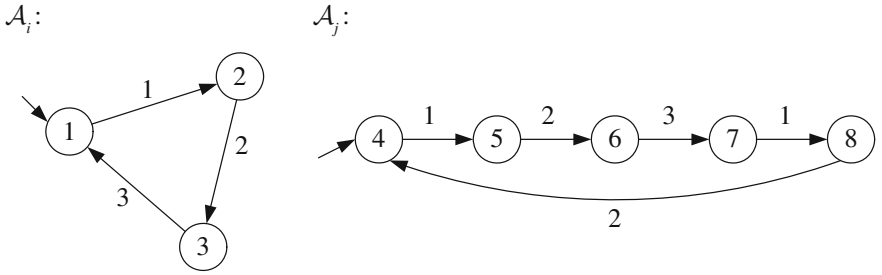


Fig. 11.17 Two automata

bring the system into two states $z(k)$ or $\tilde{z}(k)$, respectively, that are 0-distinguishable. Hence, for these states an input symbol $v(k)$ exists for which the outputs $w(k)$ and $\tilde{w}(k)$ generated in both states are different:

$$w(k) = H(z(k), v(k)) \neq H(\tilde{z}(k), V(k)) = \tilde{w}(k).$$

If the states to be tested belong to different automata with N_i of N_j states, respectively, the maximum length necessary to distinguish these states depends on the cardinality of both state sets. At most $\max_{i,j}(N_i, N_j) + 1$ input symbols are necessary. Figure 11.17 illustrates this fact for two automata with 3 or 5 states. To distinguish the states 1 and 4, an input sequence of length 6 is necessary.

11.4.3 Fault Detectability

Fault detection concerns the question whether or not a fault has occurred in the system. This section deals with the question under what conditions a fault f can be detected.

The notion of fault detectability describes the property of a system to change its behaviour in case of a fault f in such a way that a diagnostic system, knowing the I/O pair and the model $\mathcal{A}_0$ of the faultless system, can detect the fault f. The fault is detectable only if the I/O pair (V, W) generated by the faulty system is not consistent with the behaviour $\mathcal{B}_0$ of the faultless system. Since for the system subject to fault f, the output sequence is given by $W = \phi_f(z_{f0}, V)$, this condition can be represented as

$$(V, \phi_f(z_{f0}, V)) \notin \mathcal{B}_0. \tag{11.58}$$

Note that whether or not the relation (11.58) is satisfied depends upon the input sequence V. There may exist input sequences V such that the I/O pair generated by the faulty system coincides with some I/O pair of the faultless system and, hence, do not give any indication for the diagnostic system to detect the fault, whereas for other input sequences Eq. (11.58) holds.

Hence, fault detectability has to be defined as the chance to find out the presence of the fault. This "chance" exists if there is some input sequence such that the relation (11.58) is satisfied. This fact is the motivation for the following definition, in which, as before, it is assumed that the faultless system is described by the initialised automaton $(\mathcal{A}_0, z_{00})$ with the automaton map ϕ_0 and the system subject to fault f by the initialised automaton $(\mathcal{A}_f, z_{f0})$ with the automaton map ϕ_f.

Definition 11.2 (*Fault detectability*) The fault f is said to be *detectable* if there exists a finite input sequence V such that the relation

$$(V, \phi_f(z_{f0}, V)) \notin \mathcal{B}_0 \tag{11.59}$$

holds, where $\mathcal{B}_0$ denotes the behaviour of the faultless system $\mathcal{A}_0$.

Note that the detectability of a fault f is a property of the system to change its dynamical behaviour with respect to the faultless system $\mathcal{A}_0$. For a system, there may exist detectable and undetectable faults.

Detectability test. A direct consequence of the detectability definition is described in the following lemma:

Lemma 11.2 *A fault f is detectable if and only if the behaviour $\mathcal{B}_f$ of the system subject to fault f is different from the behaviour $\mathcal{B}_0$ of the faultless system*

$$\mathcal{B}_f \neq \mathcal{B}_0. \tag{11.60}$$

Condition (11.60) claims that there exists at least one I/O pair occurring for the faulty system that does not occur for the faultless system. In Fig. 11.18 the I/O pair

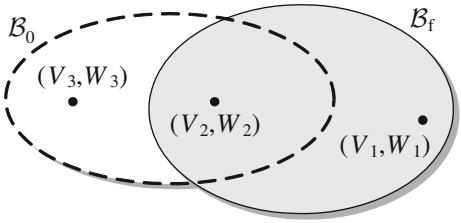


Fig. 11.18 Illustration of the detectability condition (11.60)

(V_1, W_1) satisfies the relation (11.59). It shows that if the input sequence V_1 is applied to the faulty system, the output sequence W_1 occurs and the fault can be detected because the I/O pair (V_1, W_1) is not consistent with the behaviour $\mathcal{B}_0$ of the faultless system. The figure also shows that the fault can only be detected for specific input sequences. If instead of V_1 the input sequence V_2 is applied, the faulty system results in the I/O pair (V_2, W_2) that belongs not only to $\mathcal{B}_f$ but also to $\mathcal{B}_0$ and the diagnostic system does not get any indication for the presence of the fault.

The property of fault detectability can be tested as follows:

Theorem 11.1 (Detectability criterion) *The fault f is detectable if and only if the initial states z_{00} and z_{f0} of the deterministic automata $\mathcal{A}_0$ or $\mathcal{A}_f$, respectively, are distinguishable.*

Proof (Sufficiency:) If the initial states are equivalent, then Eq. (11.49) holds, which for the automata considered here reads as

$$\phi_f(z_{f0}, V) = \phi_0(z_{00}, V) \qquad \text{for all } V \in \mathcal{V}^*.$$

Hence, the behaviours $\mathcal{B}_0$ and $\mathcal{B}_f$ of the automata $\mathcal{A}_0$ and $\mathcal{A}_f$ are identical, which is in contradiction to Eq. (11.60).

(Necessity:) If the initial states are distinguishable, then there exists an input sequence V such that the inequality

$$W_0 = \phi_0(z_{00}, V) \neq W_f = \phi_f(z_{f0}, V)$$

holds. Hence, the I/O pair (V, W_f) belongs to the behaviour $\mathcal{B}_f$ of the faulty system but not to the behaviour $\mathcal{B}_0$ of the faultless system and, due to Eq. (11.59), the fault is detectable. □

Fault detectability test. According to Theorem 11.1, the detectability of a fault f can be tested by applying the equivalence test described in Sect. 11.4.2 to the initial states z_{00} and z_{f0} of the automata $\mathcal{A}_0$ and $\mathcal{A}_f$. The complexity of this test is $O(N^2)$.

Remarks. The degree of distinguishability of the initial states z_0 and z_{f0} is a measure of the length of the input sequence V for which the fault f can be detected. If both states are k-distinguishable, a distinguishing input sequence $V(0 \ldots k)$ of length $k+1$ exists. A method for finding distinguishing input sequences with minimum length will be developed in Sect. 11.4.5.

To solve the fault detection task necessitates only the availability of the model $\mathcal{A}_0$ of the faultless system. However, the information included in this model is neither sufficient for the test of the detectability of the faults $f \in \mathcal{F}$ nor for the determination of a distinguishing input sequence V for which Eq. (11.59) holds. For both steps it has to be known how the behaviour of the faulty system distinguishes from the behaviour of the faultless system. This information is included in the model set $\{\mathcal{A}_f \mid f \in \mathcal{F}\}$.

11.4.4 Fault Identifiability

Fault identification is the task to find the fault $f \in \mathcal{F}$ that the system is subject to. This task requires to know the whole model set $\{\mathcal{A}_f \mid f \in \mathcal{F}\}$.

This section deals with the important question under what conditions it is possible to identify a fault f. Fault identifiability is a system property that depends upon the automaton maps ϕ_f or, equivalently, upon the state transition functions G_f and output functions H_f of the system for all $f \in \mathcal{F}$, but not on the diagnostic method.

The notion of fault identifiability should describe the situation that a diagnostic unit can be able to identify a fault after a finite number of input symbols. That is, for the I/O pairs generated for a specific input sequence V the relations

$$(V, W_{\tilde{f}}) \in \mathcal{B}_{\tilde{f}} \tag{11.61}$$

$$(V, W_f) \notin \mathcal{B}_f \quad \text{for all} \quad f \neq \tilde{f} \tag{11.62}$$

should hold for some fault $\tilde{f}$. Like fault detectability, the possibility to find the fault depends upon a reasonable choice of the input sequence V.

Figure 11.19 illustrates this situation for a system with the fault set $\mathcal{F} = \{0, 1, 2\}$. For every fault case $f \in \mathcal{F}$, there exists an input sequence denoted by V_f, $(f \in \mathcal{F})$ such that the output sequence

$$W_f = \phi(z_{f0}, V_f)$$

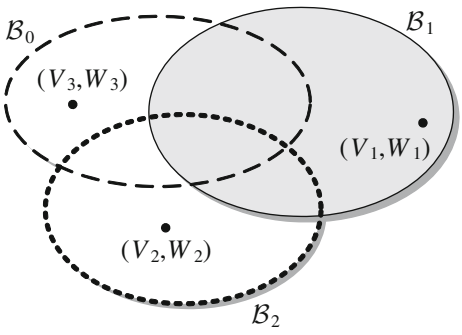


Fig. 11.19 Illustration of fault identification

generated by the system subject to fault f results in an I/O pair (V_f, W_f) that only belongs to the behaviour $\mathcal{B}_f$ relevant to this fault and not to the behaviour of the other fault cases. If the system gets this input sequence, the fault f can be unumbiguously identified.

Definition 11.3 (*Fault identifiability*) Consider a system that is described by a set $\{\mathcal{A}_f \mid f \in \mathcal{F}\}$ of deterministic automata. The fault $\tilde{f}$ is called *identifiable*, if there exists an input sequence V such that Eqs. (11.61) and (11.62) hold with $W_f = \phi_f(z_{f0}, V)$ and, hence, the set of fault candidates is a singleton: $\mathcal{F}^*(V, W) = \{\tilde{f}\}$.

Identifiability test. Before stating the identifiability criterion, the fault identification task is reformulated. Fault identification can be considered as the task to decide for a given I/O pair (V, W) which component $\mathcal{A}_f$ of the model set $\{\mathcal{A}_f \mid f \in \mathcal{F}\}$ generates for the input sequence V the output sequence W. The overall model

$$\bar{\mathcal{A}} = (\bar{\mathcal{Z}}, \mathcal{V}, \mathcal{W}, \bar{G}, \bar{H}),$$

which includes the behaviour of all models $\mathcal{A}_f$, $(f \in \mathcal{F})$ is obtained as follows:

$$\bar{\mathcal{Z}} = \cup_{f\in\mathcal{F}} \mathcal{Z}_f \tag{11.63}$$

$$\bar{G}(z, v) = G_f(z, v) \quad \text{if } z \in \mathcal{Z}_f \tag{11.64}$$

$$\bar{H}(z, v) = H_f(z, v) \quad \text{if } z \in \mathcal{Z}_f. \tag{11.65}$$

It is initialised with one of the initial states z_{f0}:

$$\bar{z} \in \bar{\mathcal{Z}}_0 = \{z_{f0} \mid f \in \mathcal{F}\}. \tag{11.66}$$

For such an initial state and an input sequence V, the automaton $\bar{\mathcal{A}}$ generates one of the output sequences that the models of the set $\{\mathcal{A}_f \mid f \in \mathcal{F}\}$ can generate. Hence, its behaviour $\bar{\mathcal{B}}$ includes the behaviour of all models $\mathcal{A}_f$:

$$\bar{\mathcal{B}} = \cup_{f\in\mathcal{F}} \mathcal{B}_f.$$

Theorem 11.2 (Identifiability criterion) [215] *Assume that all automata $\mathcal{A}_f$, $(f \in \mathcal{F})$ are minimal. Then all faults $f \in \mathcal{F}$ are identifiable if the automaton $\bar{\mathcal{A}}$ is minimal.*

Example 11.2 System with identifiable faults
As an example, consider the model $\mathcal{A}_f$, $(f = 0, 1, 2)$ shown in Fig. 11.20. The absence of equivalent states is proved by means of the state transition function $\bar{G}$ and the output function $\bar{H}$ of the automaton $\bar{\mathcal{A}}$ obtained by Eqs. (11.64), (11.65):

$\bar{G} =$

v \ z	1	2	3	4	5
1	2	1	4	3	5
2	2	1	4	3	5

,

$\bar{H} =$

v \ z	1	2	3	4	5
1	1	2	1	1	1
2	1	2	1	2	1

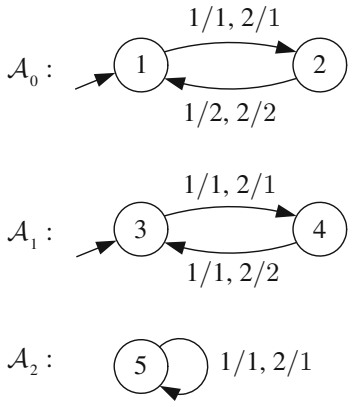


Fig. 11.20 Models of three fault cases

The analysis of the output function $\bar{H}$ results in the function H^*

$H^* =$

v \ z	$\mathcal{Z}_1^0$			$\mathcal{Z}_2^0$	$\mathcal{Z}_3^0$
	1	3	5	2	4
1	1	1	1	2	1
2	1	1	1	2	2

and the sets

$$\mathcal{Z}_1^0 = \{1, 3, 5\}, \quad \mathcal{Z}_2^0 = \{2\}, \quad \mathcal{Z}_3^0 = \{4\}.$$

Note that in the table representing the function H^*, the columns belonging to all states of the same set $\mathcal{Z}_i^0$ are identical. For a given input v, the entry in the corresponding row represents the value $H^*(\mathcal{Z}_i^0, v)$. Furthermore, the function G^* is obtained

$G^* =$

v \ z	$\mathcal{Z}_1^1$	$\mathcal{Z}_2^1$	$\mathcal{Z}_3^1$	$\mathcal{Z}_4^1$	$\mathcal{Z}_5^1$
	1	3	5	2	4
1	$\mathcal{Z}_2^0$	$\mathcal{Z}_3^0$	$\mathcal{Z}_1^0$	$\mathcal{Z}_1^0$	$\mathcal{Z}_1^0$
2	$\mathcal{Z}_2^0$	$\mathcal{Z}_3^0$	$\mathcal{Z}_1^0$	$\mathcal{Z}_1^0$	$\mathcal{Z}_1^0$

which proves that the automaton $\bar{\mathcal{A}}$ is minimal.

To select input sequences for which the faults are identifiable, note that the automata $\mathcal{A}_0$ and $\mathcal{A}_1$ are in the initial states $z_{00} = 1$ or $z_{01} = 3$, respectively, as shown in the figure. With the input sequence

$$V_{\text{det}} = (1, 1)$$

a fault can be detected, because the output sequences

$$W_{0\text{det}} = (1, 2), \quad W_{1\text{det}} = (1, 1), \quad W_{2\text{det}} = (1, 1)$$

are different for the faultless case ($W_{0\text{det}}$) and for the faulty cases ($W_{1\text{det}}$, $W_{2\text{det}}$). With the input sequence

$$V_{\text{id}} = (1, 2)$$

the output sequences of the models $\mathcal{A}_1$ and $\mathcal{A}_2$ are different

$$W_{1\text{id}} = (1, 2), \quad W_{2\text{id}} = (1, 1)$$

and the fault can be identified. In summary, the concatenated input sequence

$$V = V_{\text{det}} \cdot V_{\text{id}} = (1, 1, 1, 2)$$

yields the output sequences

$$\begin{aligned} W_0 &= W_{0\text{det}} \cdot W_{0\text{id}} = (1, 2, 1, 2) \\ W_1 &= W_{1\text{det}} \cdot W_{1\text{id}} = (1, 1, 1, 2) \\ W_2 &= W_{2\text{det}} \cdot W_{2\text{id}} = (1, 1, 1, 1), \end{aligned}$$

which unambiguously identify the fault. □

The example also shows that the length $4 = N_1 + N_2 + N_3 - 1$ of the distinguishing input sequence V depends on the cardinality $N_1 + N_2 + N_3$ of the overall model $\bar{\mathcal{A}}$ rather than on the separate cardinalities N_i of the state sets of the models $\mathcal{A}_i$, $(i = 0, 1, 2)$. However, the length 2 of the inputs V_{det} and V_{id} are due to the cardinality of pairs of submodels ($2 = \max_{i \neq j}(N_i + N_j) - 1$).

11.4.5 Method for Determining Distinguishing Input Sequences

The important result of the investigations of the preceding section is the fact that for identifiable faults there exist distinguishing input sequences V with finite length such that the pair (V, W) belongs to precisely one set $\mathcal{B}_f$, $(f \in \mathcal{F})$. The question considered in this section is how to find this input sequence:

Determination of distinguishing input sequences	
Given:	Deterministic automaton $\mathcal{A}$ with automaton map ϕ
	State pair $(z, \tilde{z})$
Find:	Input sequence V such that $\phi(z, V) \neq \phi(\tilde{z}, V)$

The main problem to be solved concerns the determination of the shortest input sequence with which identifiable faults can be unambiguously identified. It should

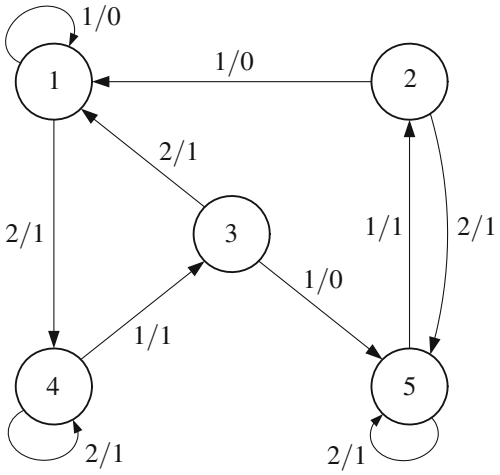


Fig. 11.21 Automaton

become clear after a *minimum* number of input symbols to which behaviour $\mathcal{B}_f$, ($f \in \mathcal{F}$) the I/O pair belongs.

The preceding section also has shown that the identification of a fault f can be reduced to the problem of identifying the state of the automaton $\bar{\mathcal{A}}$. If the result belongs to the state set $\mathcal{Z}_f$, then the fault f has been identified. As a consequence, this section deals with the problem of finding the shortest distinguishing input sequence for identifying the state of the automaton $\bar{\mathcal{A}}$.

Explanation of the method by an example. The solution to the problem stated above will be explained first by considering the example automaton shown in Fig. 11.21.

Example 11.3 Determination of distinguishing inputs
The automaton is described by the following state transition function G and output function H:

$G =$

v \ z	1	2	3	4	5
1	1	1	5	3	2
2	4	5	1	4	5

, $H =$

v \ z	1	2	3	4	5
1	0	0	0	1	1
2	1	1	1	1	1

To determine 0-equivalent states the output function is analysed with the following result:

$H^* =$

	$\mathcal{Z}_1^0$			$\mathcal{Z}_2^0$	
	1	2	3	4	5
$v = 1$	0	0	0	1	1
$v = 2$	1	1	1	1	1

The further decomposition of the state set $\mathcal{Z}$ is obtained by means of the state transition function G:

$G_0^* =$

	$\mathcal{Z}_1^1$		$\mathcal{Z}_2^1$	$\mathcal{Z}_3^1$	
	1	2	3	4	5
$v=1$	$\mathcal{Z}_1^0$	$\mathcal{Z}_1^0$	$\mathcal{Z}_2^0$	$\mathcal{Z}_1^0$	$\mathcal{Z}_1^0$
$v=2$	$\mathcal{Z}_2^0$	$\mathcal{Z}_2^0$	$\mathcal{Z}_1^0$	$\mathcal{Z}_2^0$	$\mathcal{Z}_2^0$

$\downarrow$

$G_1^* =$

	$\mathcal{Z}_1^2$		$\mathcal{Z}_2^2$	$\mathcal{Z}_3^2$	$\mathcal{Z}_4^2$
	1	2	3	4	5
$v=1$	$\mathcal{Z}_1^1$	$\mathcal{Z}_1^1$	$\mathcal{Z}_3^1$	$\mathcal{Z}_2^1$	$\mathcal{Z}_1^1$
$v=2$	$\mathcal{Z}_3^1$	$\mathcal{Z}_3^1$	$\mathcal{Z}_1^1$	$\mathcal{Z}_3^1$	$\mathcal{Z}_3^1$

$\downarrow$

$G_2^* =$

	$\mathcal{Z}_1^3$	$\mathcal{Z}_2^3$	$\mathcal{Z}_3^3$	$\mathcal{Z}_4^3$	$\mathcal{Z}_5^3$
	1	2	3	4	5
$v=1$	$\mathcal{Z}_1^2$	$\mathcal{Z}_1^2$	$\mathcal{Z}_4^2$	$\mathcal{Z}_2^2$	$\mathcal{Z}_1^2$
$v=2$	$\mathcal{Z}_3^2$	$\mathcal{Z}_4^2$	$\mathcal{Z}_1^2$	$\mathcal{Z}_3^2$	$\mathcal{Z}_4^2$

The result shows that the automaton does not possess equivalent states, because all state sets obtained are singletons.

In the following, it will be shown that the sequence of results obtained when testing the existence of equivalent states by means of Algorithm 11.2 can be used to determine distinguishing input sequences. First consider the function H^* defined in Eq. (11.52). The decomposition shows that for the input $v = 1$ the state pairs $(1, 4)$, $(1, 5)$, $(2, 4)$, $(2, 5)$, $(3, 4)$, $(3, 5)$ can be distinguished by the output that the system generates. For all six pairs, the automaton $\mathcal{A}$ gives the output $w = 0$ if it is in the first state and the output $w = 1$ for the second state. The listed state pairs are the elements of the set

$$\mathcal{Z}_1^0 \times \mathcal{Z}_2^0 = \{(1, 4), (1, 5), (2, 4), (2, 5), (3, 4), (3, 5)\}.$$

These state pairs are 0-distinguishable and the distinguishing input is $v = 1$.

The function G_0^* is obtained when partioning the state set $\mathcal{Z}_1^0$ into the sets $\mathcal{Z}_1^1$ and $\mathcal{Z}_2^1$:

$$\mathcal{Z}_1^0 = \mathcal{Z}_1^1 \cup \mathcal{Z}_2^1.$$

It shows how to distinguish between the state pairs

$$\mathcal{Z}_1^1 \times \mathcal{Z}_2^1 = \{(1, 3), (2, 3)\}.$$

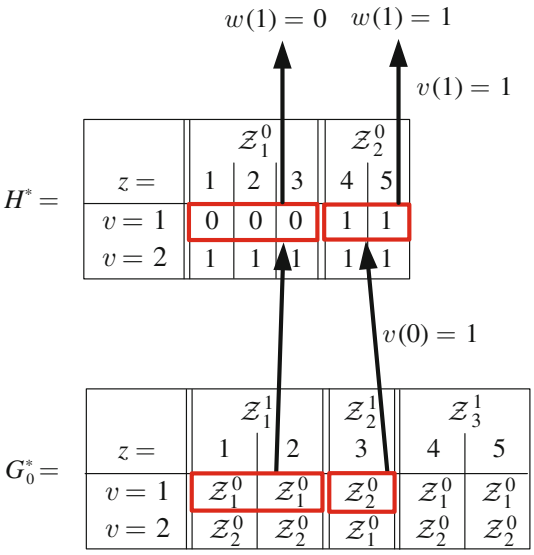


Fig. 11.22 Determination of the distinguishing input sequence of 1-distinguishing state pairs

These state pairs are 1-distinguishable and 0-equivalent, because they belong to the same set $\mathcal{Z}_1^0$ of the first decomposition and, thus, are not distinguishable without any state transition. However, if an input sequence of length 2 is used and, hence, one state transition is caused, these state pairs can be distinguished.

The distinguishing input sequence can be read off the decomposition by looking for a row in the G_0^* table with different entries for the state sets $\mathcal{Z}_1^1$ and $\mathcal{Z}_2^1$. The entries $\mathcal{Z}_1^0$ and $\mathcal{Z}_2^0$ in the first row say that using the input symbol $v(0) = 1$ state transitions occur where the states $z = 1$ and $z = 2$ are moved to one of the states $z' \in \mathcal{Z}_1^0$ whereas from the state $\tilde{z} = 3$ the automaton moves towards one of the states $\tilde{z}' \in \mathcal{Z}_2^0$ (Fig. 11.22). As the successor states belong to different sets of 0-distinguishable states, using the second input symbol $v(1) = 1$ the automaton gives the output $w(1) = 0$ if it has started in the initial state $z = 1$ or $z = 2$; whereas it generates the output $w(1) = 1$ if it has started in the state $\tilde{z} = 3$. Consequently, the state pairs

$$(z, \tilde{z}) \in \{(1, 3), (2, 3)\}$$

are 1-distinguishable and a distinguishing input sequence is

$$V(0 \ldots 1) = (1, 1).$$

As the final example, consider the last decomposition step of the state transition function G, where the set $\mathcal{Z}_1^2$ is partitioned

$$\mathcal{Z}_1^2 = \mathcal{Z}_1^3 \cup \mathcal{Z}_2^3.$$

Hence, the states 1 and 2, which are the only elements of the sets $\mathcal{Z}_1^3$ and $\mathcal{Z}_2^3$ are distinguishable after this decomposition step, which is the third one. Hence, these states are 3-distinguishable and 4 input symbols for a distinguishing input sequence $V(0 \ldots 3)$ have to be found by retracing the decomposition.

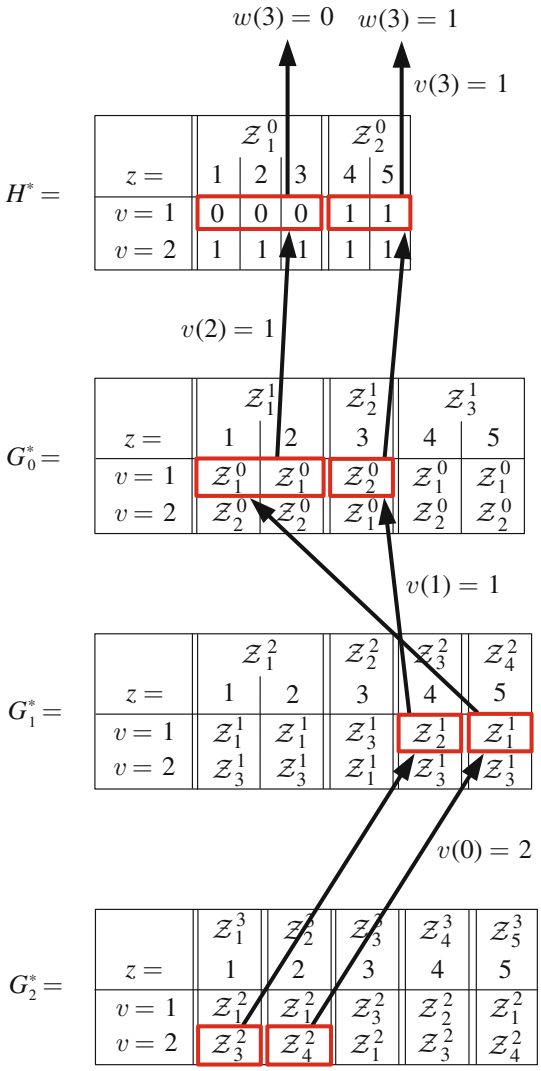


Fig. 11.23 Determination of the distinguishing input sequence of 3-distinguishing state pairs

As Fig. 11.23 shows the input sequence

$$V(0\ldots3) = (2, 1, 1, 1)$$

yields state transitions among the following state sets:

$$1 \xrightarrow{v(0)=2} \mathcal{Z}_3^2 \xrightarrow{v(1)=1} \mathcal{Z}_2^1 \xrightarrow{v(2)=1} \mathcal{Z}_2^0$$
$$2 \xrightarrow{v(0)=2} \mathcal{Z}_4^2 \xrightarrow{v(1)=1} \mathcal{Z}_1^1 \xrightarrow{v(2)=1} \mathcal{Z}_1^0.$$

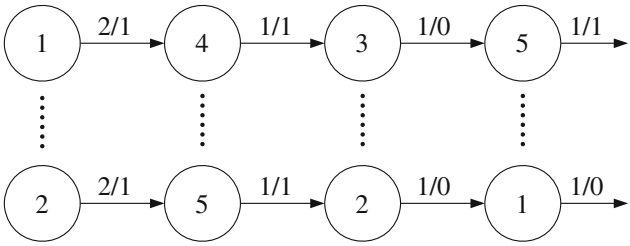


Fig. 11.24 State trajectories for determining whether the automaton is in the initial state 1 or 2

Obviously, the first three input symbols bring the automaton in a pair of 0-distinguishable states. The last input symbol $v(3) = 1$ is used to get two different output symbols $w(3) = 1$ or $w(3) = 0$, where the first symbol indicates that the automaton has started its movement in the state $z(0) = 1$; whereas the second symbol occurs if the automaton has had the initial state $z(0) = 2$.

What the automaton really does for the distinguishing input sequence, can be determined by means of the state transition function G and the output function H. The result is shown in Fig. 11.24. For the first three input symbols, the output symbols generated by the automaton are the same for both initial states. Only the last output symbol can be used as an indicator for determining the initial state in which the automaton has started its movement. This figure has the same structure as Fig. 11.16.

The analysis result can be summarised in the following table, which gives for all state pairs the distinguishing input sequence and shows which last output symbol occurs for both initial states:

z	$\tilde{z}$	$V(0 \ldots k)$	$w(k)$ for $z(0) = z$	$w(k)$ for $z(0) = \tilde{z}$
1	2	(2, 1, 1, 1)	1	0
1	3	(1, 1)	0	1
1	4	1	0	1
1	5	1	0	1
2	3	(1, 1)	0	1
2	4	1	0	1
2	5	1	0	1
3	4	1	0	1
3	5	1	0	1
4	5	(1, 1, 1)	1	0

Note that there may be several distinguishing input sequences, but the sequences shown here have minimum length. □

Algorithm. The method explained in the preceding section, for example automaton, is now formalised to get an algorithm, which proceeds in two main steps:

1. The state set $\mathcal{Z}$ is partitioned into sets of equivalent states:

$$\mathcal{Z} = \cup_{i=1}^{q} \mathcal{Z}_i .$$

As intermediate results, the partitions

$$\mathcal{Z} = \cup_{i=1}^{q_k} \mathcal{Z}_i^k, \quad k = 1, 2, \ldots, N-1$$

are obtained together with the functions H^* and G_k^*, $(k = 0, 1, \ldots, N-3)$ such that Eq. (11.51) or (11.53) and

$$\mathcal{Z}_i = \mathcal{Z}_i^{N-2}, \quad i = 1, 2, \ldots, q$$

hold. If z and $\tilde{z}$ are not equivalent, they belong to different sets $\mathcal{Z}_i$ and $\mathcal{Z}_j$ $(i \neq j)$. Otherwise, no distinguishing input sequence exists.

2. The result of the first step is evaluated backwards in the following way. First, find the number k_e for which the states z and $\tilde{z}$ are elements of a common set $\mathcal{Z}_l^{k_\mathrm{e}-1}$ and of disjoint sets $\mathcal{Z}_i^{k_\mathrm{e}}$ and $\mathcal{Z}_j^{k_\mathrm{e}}$, $(i \neq j)$:

$$k_\mathrm{e}: \; z, \tilde{z} \in \mathcal{Z}_l^{k_\mathrm{e}-1} \quad \text{and} \quad z \in \mathcal{Z}_i^{k_\mathrm{e}}, \; \tilde{z} \in \mathcal{Z}_j^{k_\mathrm{e}} \text{ for some } i \neq j. \tag{11.67}$$

Hence, z and $\tilde{z}$ are k_e-distinguishable. Introduce the new symbols

$$\mathcal{Z}^{k_\mathrm{e}} = \mathcal{Z}_i^{k_\mathrm{e}} \quad \text{and} \quad \tilde{\mathcal{Z}}^{k_\mathrm{e}} = \mathcal{Z}_j^{k_\mathrm{e}}.$$

Second, to find the distinguishing input sequence $V(0 \ldots k_\mathrm{e})$ determine the input $v(k_\mathrm{e} - k) = v_k$ for $k = k_\mathrm{e}, k_\mathrm{e} - 1, \ldots, 1$ such that

$$G_{k-1}^*(\mathcal{Z}^k, v_k) = \mathcal{Z}^{k-1} \neq G_{k-1}^*(\tilde{\mathcal{Z}}^k, v_k) = \tilde{\mathcal{Z}}^{k-1}. \tag{11.68}$$

Finally, choose the input $v(k_\mathrm{e}) = v_0$ such that

$$H^*(\mathcal{Z}^0, v_0) = w \neq H^*(\tilde{\mathcal{Z}}^0, v) = \tilde{w}. \tag{11.69}$$

The result is the distinguishing input sequence

$$V(0 \ldots k_\mathrm{e}) = (v(0), v(1), \ldots, v(k_\mathrm{e})) = (v_{k_\mathrm{e}}, v_{k_\mathrm{e}-1}, \ldots, v_1, v_0). \tag{11.70}$$

These steps can be formally described as the following algorithm:

Algorithm 11.3 *Determination of a distinguishing input sequence*

Given: Deterministic automaton $\mathcal{A}$ and state pair $(z, \tilde{z})$

1. Determine the sets $\mathcal{Z}_i^0$, $(i = 1, \ldots, q_0)$ and the function H^* according to Eq. (11.52).
2. Determine the sets $\mathcal{Z}_i^k$, $(k = 1, 2, \ldots, N-1,\ i = 1, \ldots, q_k)$ and the functions G_k^*, $(k = 1, 2, \ldots, N-2)$ according to Eq. (11.54).

   If z and $\tilde{z}$ belong to the same set $\mathcal{Z}_i^{N-1}$, stop (they are not distinguishable).
3. Determine the length k_e of the distinguishing input sequence according to Eq. (11.67).
4. For $k = k_\mathrm{e}, k_\mathrm{e} - 1, \ldots, 1$ determine v_k such that Eq. (11.68) holds.
5. Determine v_0 by means of Eq. (11.69).

Result: Distinguhishing input sequence (11.70).

Theorem 11.3 (Minimum distinguishing input sequences) *Algorithm 11.3 results in a minimum distinguishing input sequence whenever the states z and $\tilde{z}$ are distinguishable.*

Proof According to Lemma 11.1, the state set partition obtained by Steps 1 and 2 of the algorithm is unique. The states z and $\tilde{z}$ belong to the same set $\mathcal{Z}_i^k$ if and only if they are k-equivalent. Hence, Eq. (11.67) ensures that the length k_e of the input sequence is minimal. Furthermore, the choice of the input symbols according to Eq. (11.68) ensures that the state sequence ends in 0-distinguishing state sets and, finally, the output sequences generated by the automaton $\mathcal{A}$ for the two different initial states z and $\tilde{z}$ distinguish in the last symbol $w(k_\mathrm{e})$. Hence, the input sequence obtained by the algorithm is distinguishing. □

11.5 Diagnosis of Nondeterministic Automata

11.5.1 Method for Testing the Consistency of an I/O Pair with a Nondeterministic Automaton

This section extends the diagnostic method explained in Sect. 11.4 for deterministic automata towards nondeterministic automata

$$\mathcal{N} = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, L_\mathrm{n}, \mathcal{Z}_0).$$

The main problems to be solved concern the new description of the system dynamics by the behavioural relation L_n, which replaces the functions G and H of the deterministic automaton, and the fact that the nondeterminism of the model brings about ambiguities into the diagnostic result with any state transition.

The main idea is again the test of the consistency of the measured I/O pair $(V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e}))$ with the model $\mathcal{N}_f$ of the system subject to fault f. As this test is independent of the fault case, the index f is left out in the following development of the test method. $\mathcal{Z}_0$ is a given set of states, in which the automaton starts its movement.

According to the definition (11.4), an I/O pair (V, W) is consistent with a model $\mathcal{N}$ if the relation

$$(V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e})) \in \mathcal{B}$$

holds, where $\mathcal{B}$ is the behaviour (11.15) of the model $\mathcal{N}$. In order to show clearly how to carry out this test, the elements of the measured sequences are marked by a bar:

$$V(0\ldots k_\mathrm{e}) = (\bar{v}_0, \bar{v}_2, \ldots, \bar{v}_{k_\mathrm{e}}) \tag{11.71}$$

$$W(0\ldots k_\mathrm{e}) = (\bar{w}_0, \bar{w}_2, \ldots, \bar{w}_{k_\mathrm{e}}). \tag{11.72}$$

For these representations, consistency with the model $\mathcal{N}$ claims that there exists a state sequence

$$Z(0\ldots k_\mathrm{e}+1) = (z_0, z_1, \ldots, z_{k_\mathrm{e}+1})$$

such that the relation

$$L_\mathrm{n}(z_{k+1}, \bar{w}_k, z_k, \bar{v}_k) = 1$$

holds for $k = 0, 1, \ldots, k_\mathrm{e}$ and $z_0 \in \mathcal{Z}_0$. Equivalently, the relation

$$\exists Z(0\ldots k_\mathrm{e}+1): \prod_{k=0}^{k_\mathrm{e}} L_\mathrm{n}(z_{k+1}, \bar{w}_k, z_k, \bar{v}_k) = 1 \tag{11.73}$$

has to be valid, which can be represented by summing over all possible state sequences:

$$\sum_{z_{k_\mathrm{e}+1}\in\mathcal{Z}} \sum_{z_{k_\mathrm{e}}\in\mathcal{Z}} \cdots \sum_{z_0\in\mathcal{Z}_0} \prod_{k=0}^{k_\mathrm{e}} L_\mathrm{n}(z_{k+1}, \bar{w}_k, z_k, \bar{v}_k) \geq 1. \tag{11.74}$$

This fact is stated by the following lemma:

Lemma 11.3 *The I/O pair (11.71), (11.72) is consistent with the nondeterministic automaton $\mathcal{N}$ if and only if the condition (11.74) is satisfied.*

Recursive test. The following presents a recursive test of the condition (11.74). The function $p(z', k_e + 1)$ is used as an indicator whether $(p(z', k_e + 1) = 1)$ or not $(p(z', k_e + 1) = 0)$ a state sequence $Z(0 \ldots k_e + 1)$ with the last element $z_{k_e+1} = z'$ exists for which the condition (11.73) is satisfied. This function can be determined recursively as follows:

$$p'(z, 0) = \begin{cases} 1 & \text{if } z \in \mathcal{Z}_0 \\ 0 & \text{else} \end{cases} \tag{11.75}$$

$$p'(z', k+1) = \left\lfloor \sum_{z \in \mathcal{Z}} L_n(z', \bar{w}_k, z, \bar{v}_k) \cdot p'(z, k) \right\rfloor, \quad k = 0, \ldots, k_e, \tag{11.76}$$

where the symbol $\lfloor . \rfloor$ signifies the replacement of any positive integer by 1:

$$\lfloor p \rfloor = \begin{cases} 0 & \text{if } p = 0 \\ 1 & \text{if } p \geq 1. \end{cases}$$

The I/O pair (11.71), (11.72) is consistent with the model $\mathcal{N}$ if and only if the inequality

$$\bar{p} = \left\lfloor \sum_{z' \in \mathcal{Z}} p'(z', k_e + 1) \right\rfloor > 0 \tag{11.77}$$

holds.

As a remark, it should be mentioned that the operation $\lfloor . \rfloor$ reduces the value of the argument to 1 in order to avoid increasing values in the sequence $p(z, k)$, $(k = 0, 1, \ldots, k_e)$. The consistency test only distinguishes between vanishing and non-vanishing values.

Lemma 11.4 *The I/O pair (11.71), (11.72) is consistent with the model $\mathcal{N}$ if and only if the condition (11.77) is satisfied, where $p(z, k_e + 1)$ is obtained by the recursion relation (11.75), (11.76).*

Proof The lemma is proved by showing that the condition (11.77) is satisfied if and only if the condition (11.74) is satisfied, which is necessary and sufficient for the consistency. To do so, define the function $\tilde{p}(z, k)$ as follows:

$$\tilde{p}(z, 0) = \begin{cases} 1 & \text{if } z \in \mathcal{Z}_0 \\ 0 & \text{else} \end{cases} \tag{11.78}$$

$$\tilde{p}(z', k+1) = \sum_{z \in \mathcal{Z}} L_n(z', \bar{w}_k, z, \bar{v}_k) \cdot \tilde{p}(z, k) \quad k = 0, 1, \ldots, k_e. \tag{11.79}$$

Obviously, $p'(z, k) > 0$ holds if and only if $\tilde{p}(z, k) > 0$ is valid. It will be proved now that the equation

$$\tilde{p}(z_{k_e+1}, k_e+1) = \sum_{z_{k_e}\in\mathcal{Z}} \cdots \sum_{z_0\in\mathcal{Z}_0} \prod_{k=0}^{k_e} L_n(z_{k+1}, \bar{w}_k, z_k, \bar{v}_k) \tag{11.80}$$

holds which proves the lemma.

The proof is done by induction. For $k_e = 0$, Eqs. (11.78) and (11.79) yield

$$\begin{aligned}\tilde{p}(z_1, 1) &= \sum_{z_0\in\mathcal{Z}} L_n(z_1, \bar{w}_0, z_0, \bar{v}_0) \cdot \tilde{p}(z_0, 0) \\ &= \sum_{z_0\in\mathcal{Z}_0} L_n(z_1, \bar{w}_0, z_0, \bar{v}_0),\end{aligned}$$

which is identical to Eq. (11.80) for $k_e = 0$.

Now assume that Eq. (11.80) holds for some $k_e = k_e$

$$\tilde{p}(z_{k_e+1}, k_e+1) = \sum_{z_{k_e}\in\mathcal{Z}} \cdots \sum_{z_0\in\mathcal{Z}_0} \prod_{k=0}^{k_e} L_n(z_{k+1}, \bar{w}_k, z_k, \bar{v}_k)$$

and prove that this relation is satisfied for $k_e = k_e + 1$:

$$\tilde{p}(z_{k_e+2}, k_e+2) = \sum_{z_{k_e+1}\in\mathcal{Z}} \cdots \sum_{z_0\in\mathcal{Z}_0} \prod_{k=0}^{k_e+1} L_n(z_{k+1}, \bar{w}_k, z_k, \bar{v}_k). \tag{11.81}$$

A reformulation of the right-hand side of this equation results in

$$\begin{aligned}&\sum_{z_{k_e+1}\in\mathcal{Z}} \sum_{z_{k_e}\in\mathcal{Z}} \cdots \sum_{z_0\in\mathcal{Z}_0} \prod_{k=0}^{k_e+1} L_n(z_{k+1}, \bar{w}_k, z_k, \bar{v}_k) \\ &= \sum_{z_{k_e+1}\in\mathcal{Z}} L_n(z_{k_e+2}, \bar{w}_{k_e+1}, z_{k_e+1}, \bar{v}_{k_e+1}) \\ &\quad \cdot \left(\sum_{z_{k_e}\in\mathcal{Z}} \cdots \sum_{z_0\in\mathcal{Z}_0} \prod_{k=0}^{k_e} L_n(z_{k+1}, \bar{w}_k, z_k, \bar{v}_k) \right) \\ &= \sum_{z_{k_e+1}\in\mathcal{Z}} L_n(z_{k_e+2}, \bar{w}_{k_e+1}, z_{k_e+1}), \bar{v}_{k_e+1}) \cdot \tilde{p}(z_{k_e+1}, k_e+1) \\ &= \tilde{p}(z_{k_e+2}, k_e+2)\end{aligned}$$

which proves Eq. (11.81). □

Test algorithm. The recursive test leads to the following algorithm:

Algorithm 11.4 *Consistency test for nondeterministic automata*

Given: Nondeterministic automaton $\mathcal{N}$
I/O pair (11.71), (11.72)

1. Determine $p'(z, 0)$ by Eq. (11.75)
2. Apply Eq. (11.76) for $k = 0, 1, \ldots, k_e$ to determine $p'(z', k_e + 1)$
3. Test the condition (11.77).

Result: If and only if the condition (11.77) is satisfied, the I/O pair is consistent with the model $\mathcal{N}$

State observation result. As a byproduct of the consistency test, the set $\mathcal{Z}(k_e + 1)$ of states is obtained in which the automaton $\mathcal{N}$ can recide after it has accepted the input sequence $V(0 \ldots k_e)$ and generated the output sequence $W(0 \ldots k_e)$. This set is given by

$$\mathcal{Z}(k+1) = \{z' \in \mathcal{Z} \mid p'(z', k+1) > 0\}, \quad k = 0, 1, \ldots, k_e. \tag{11.82}$$

Hence, all state sequences $Z(0 \ldots k_e)$ considered in the consistency tests end in a state

$$z(k_e + 1) \in \mathcal{Z}(k_e + 1). \tag{11.83}$$

Corollary 11.1 *The I/O pair* $(V(0 \ldots k_e), W(0 \ldots k_e))$ *is consistent with the model* $\mathcal{N}$ *if and only if* $\mathcal{Z}(k_e + 1) \neq \emptyset$ *holds.*

To present the result of the consistency test in a similar way as for deterministic automata, the indicator $p(k)$ is introduced as follows:

$$p(k_e) = \begin{cases} 1 \text{ if } \mathcal{Z}(k_e + 1) \neq \emptyset \\ 0 \text{ else.} \end{cases}$$

11.5.2 Diagnostic Algorithm

The consistency test developed so far has to be applied to a set $\{\mathcal{N}_f \mid f \in \mathcal{F}\}$ of nondeterministic automata to get a fault identification algorithm. This algorithm works online, where the next measured I/O pair $(\bar{v}, \bar{w})$ is processed to get the sets $\mathcal{Z}$ and $\mathcal{Z}'$ of possible current states z or possible future states z', respectively, to decide about the fault candidates. Instead of $\mathcal{Z}$ and $\mathcal{Z}'$, indicators $p(z)$ and $p'(z')$ for the states $z, z' \in \mathcal{Z}$ to be elements of $\mathcal{Z}$ or $\mathcal{Z}'$, respectively, are determined. A model $\mathcal{N}_f$ is consistent with the I/O pair up to the current time horizon k_e if these sets are

non-empty, which is again represented by the indicator

$$p_f(k_\mathrm{e}) = \begin{cases} 1 \text{ if } \mathcal{Z}_f(k_\mathrm{e}+1) \neq \emptyset \\ 0 \text{ else.} \end{cases}$$

With this information, the diagnostic algorithm for nondeterministic automata can be formulated similarly to Algorithm 11.1 as follows:

Algorithm 11.5 *Diagnosis of nondeterministic automata*

Given: Nondeterministic automata $\mathcal{N}_f$, $(f \in \mathcal{F})$
Set of initial states $\mathcal{Z}_0$

Initialisation: $p'_f(z) = \begin{cases} 1 & \text{if } z \in \mathcal{Z}_0 \\ 0 & \text{else} \end{cases}$ for all $f \in \mathcal{F}$
$k_\mathrm{e} = 0$

Loop: 1. Measure the next I/O pair $(\bar{v}, \bar{w})$.

2. Determine $p_f(z) = \left\lfloor \sum_{z' \in \mathcal{Z}} L_\mathrm{n}(z', \bar{w}, z, \bar{v}) \cdot p'_f(z) \right\rfloor$ for all $f \in \mathcal{F}$.

3. Determine $p'_f(z') = \left\lfloor \sum_{z \in \mathcal{Z}} L_\mathrm{n}(z', \bar{w}, z, \bar{v}) \cdot p_f(z) \right\rfloor$ for all $f \in \mathcal{F}$.

4. Set $p_f(k_\mathrm{e}) = \left\lfloor \sum_{z' \in \mathcal{Z}} p'_f(z') \right\rfloor$ for all $f \in \mathcal{F}$.

5. Determine $\mathcal{F}^*(k_\mathrm{e}) = \{f \in \mathcal{F} : p_f(k_\mathrm{e}) = 1\}$.

6. $k_\mathrm{e} := k_\mathrm{e} + 1$
Continue with Step 1.

Result: Set of fault candidates $\mathcal{F}^*(k_\mathrm{e})$ for increasing time horizon k_e

The sets $\mathcal{Z}_f$ and $\mathcal{Z}'_f$ of current states or future states that are necessary to decide about the consistency can be obtained by

$$\begin{aligned} \mathcal{Z}_f &= \{z \in \mathcal{Z} : p_f(z) = 1\} \\ \mathcal{Z}'_f &= \{z' \in \mathcal{Z} : p'_f(z') = 1\}. \end{aligned}$$

Algorithm 11.5 can be used to solve the following fault diagnostic tasks:

- **Fault detection**: If the algorithm is applied to the single model $\mathcal{N}_0$ describing the faultless system, a fault is detected, if after Step 4 the relation $p_0(k_\mathrm{e}) = 0$ holds.

- **Fault identification**: For a set of models $\mathcal{N}_f$, ($f \in \mathcal{F}$), the algorithm yields the set of fault candidates $\mathcal{F}^*(k_e)$, which is the best possible diagnostic result.

Example 11.4 Fault detection of a nondeterministic automaton
Consider the nondeterministc automata $\mathcal{N}_0$ and $\mathcal{N}_1$ whose automaton graphs are depicted in Fig. 11.25. They describe a system in the faultless case or subject to fault $f = 1$, respectively.

The initial state is assumed to belong to the set

$$\mathcal{Z}_0(0 \mid -1) = \{1, 2, 3, 4\} \quad \text{and} \quad \mathcal{Z}_1(0 \mid -1) = \{5, 6, 7, 8\},$$

which yields the initial value of $\mathcal{Z}_0'$ and $\mathcal{Z}_1'$ in the initialisation step of the algorithm. After the first measurement

$$v(0) = 1 \quad \text{and} \quad w(0) = 1$$

has been obtained, Step 2 yields $p_0(z)$ and $p_1(z)$ for all $z \in \mathcal{Z}$ and, hence, the following sets

$$\mathcal{Z}_0(0 \mid 0) = \{1, 2, 3\} = \mathcal{Z}_0 \quad \text{and} \quad \mathcal{Z}_1(0 \mid 0) = \{5, 6, 7\} = \mathcal{Z}_1,$$

which represent all initial states z_0 for which a state transition exists such that the nondeterministic automaton generates the output $w(0) = 1$ for the input $v(0) = 1$. In the automaton graphs this sets can be found by looking for all states $z \in \mathcal{Z}(0 \mid -1)$ in which an edge starts that is labelled with $v = 1$ and $w = 1$. Step 3 the algorithm determines the sets $\mathcal{Z}_0(1 \mid 0)$ and $\mathcal{Z}_1(1 \mid 0)$ which in the notation used above coincide with the sets

$$\mathcal{Z}_0' = \{1, 2, 3, 4\} \quad \text{and} \quad \mathcal{Z}_1' = \{5, 6, 7, 8\}.$$

These are the sets of states in which the automata can be after they have generated the output $w(0) = 1$ for the input $v(0) = 1$. As both automata are consistent with the measurement obtained so far ($p_0(0)) = 1$, $p_1(0) = 1$), the set of fault candidates includes both faults:

$$\mathcal{F}^*(0) = \{0, 1\}.$$

For the next measurement

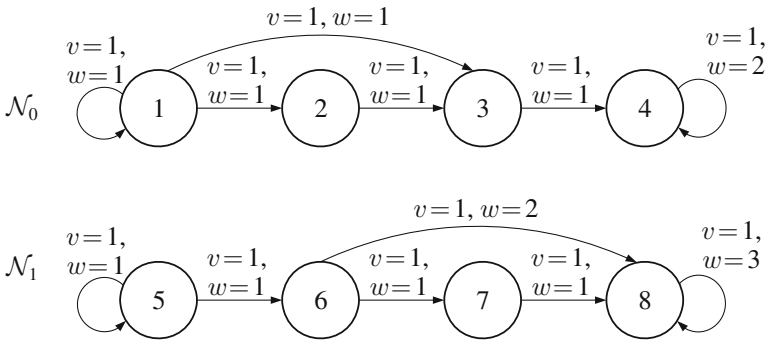


Fig. 11.25 Automaton graph of the example

$$v(1) = 1 \quad \text{and} \quad w(1) = 1$$

the Steps 2 and 3 of the algorithm yield again

$$\mathcal{Z}_0 = \{1, 2, 3\} \quad \text{and} \quad \mathcal{Z}_1 = \{5, 6, 7\}$$

and

$$\mathcal{Z}'_0 = \{1, 2, 3, 4\} \quad \text{and} \quad \mathcal{Z}'_1 = \{5, 6, 7, 8\}.$$

and for

$$v(2) = 1 \quad \text{and} \quad w(2) = 1$$

the same sets. An improvement of the diagnostic result is obtained after the measurement

$$v(3) = 1 \quad \text{and} \quad w(3) = 3$$

has been obtained, because then the results

$$\mathcal{Z}_0 = \emptyset \quad \text{and} \quad \mathcal{Z}_1 = \{8\}$$

show that the measurement sequence is inconsistent with the faultless case and, thus, the set of fault candidates only includes the single fault $f = 1$:

$$\mathcal{F}^*(3) = \{1\}.$$

Hence, the I/O pair (1, 1, 1, 1), (1, 1, 1, 3) yields the diagnostic result that a fault has occurred (the faultless case is inconsistent with the measurements) and that the set of fault candidates only includes, as a single fault, the fault case $f = 1$. □

11.6 State Observation of Stochastic Automata

This and the next section extend the methods developed so far towards stochastic automata

$$\mathcal{S} = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, L, p_0(z)).$$

The investigations are cut into two parts. In the first part described in this section, the state observation problem is solved, which does not only yield a test for the consistency of the measured I/O pair with a stochastic automaton, but also the probability for each state to be the current state of the automaton after the I/O pair has appeared. In the second part presented in the next section, this method is extended to determine the probability distribution over the set $\mathcal{F}$ of all faults considered to get a diagnostic result.

11.6.1 Method for Testing the Consistency of an I/O Pair with a Stochastic Automaton

The basis for finding fault candidates is the check whether the measured I/O pair belongs to the behaviour $\mathcal{B}$, which is defined for the stochastic automaton in Eq. (11.25). The consistency test should not only answer the question whether the I/O pair (V, W) belongs to the behaviour $\mathcal{B}$, but also with which probability this I/O pair occurs for the automaton $\mathcal{S}$. The probability information included in the behavioural relation L should be used to distinguish between I/O pairs that may occur often and those pairs that appear seldom. If this method is applied to the models of the faulty system, it should be possible to distinguish between faults with higher probability and rarely occurring faults.

This section extends the consistency test from nondeterministic towards stochastic automata. As this test is independent of the fault case, the dependency of the model upon the fault $f \in \mathcal{F}$ is omitted in this section.

The consistency test uses the representation (11.71), (11.72) of the I/O pair, where the bar over the symbols indicate that these symbols are measured and, hence, known in the test. The I/O pair defines the values for the stochastic variables $V(k)$ and $W(k)$, $(k = 0, 1, \ldots, k_\mathrm{e})$ that represent the current value of the input and the output signals.

An I/O pair is consistent with the model $\mathcal{S}$ if a state sequence

$$Z(0 \ldots k_\mathrm{e} + 1) = (z_0, z_1, \ldots, z_{k_\mathrm{e}+1}) \tag{11.84}$$

with positive probability exists, which satisfies the relation

$$\exists Z(0 \ldots k_\mathrm{e} + 1) : \prod_{k=0}^{k_\mathrm{e}} L(z_{k+1}, \bar{w}_k \mid z_k, \bar{v}_k) \cdot p_0(z_0) > 0. \tag{11.85}$$

This inequality replaces Eq. (11.73) for the nondeterministic automaton. The last term represents the a-priori probability of the initial state:

$$p_0(z) = \mathrm{Prob}(Z(0) = z).$$

Accordingly, $\mathcal{Z}_0 = \{z \in \mathcal{Z} : p_0(z) > 0\}$ holds.

Analogously to Eq. (11.74) the condition (11.85) can be formulated as

$$\sum_{z_{k_\mathrm{e}+1} \in \mathcal{Z}} \sum_{z_{k_\mathrm{e}} \in \mathcal{Z}} \cdots \sum_{z_0 \in \mathcal{Z}} \prod_{k=0}^{k_\mathrm{e}} L(z_{k+1}, \bar{w}_k \mid z_k, \bar{v}_k) \cdot p_0(z_0) \geq 0. \tag{11.86}$$

Lemma 11.5 *The I/O pair (11.71), (11.72) is consistent with the stochastic automaton $\mathcal{S}$ if and only if the condition (11.86) is satisfied.*

Recursive solution of the state observation problem. If the condition (11.86) is satisfied, there exists a state sequence that appears for the I/O pair (V, W) with positive probability. Hence, there is a non-empty set of states in which the automaton $\mathcal{S}$ resides after this I/O pair has appeared. For the solution of the fault diagnostic problem, it is not only necessary to know whether this set is non-empty, but also with which probability the elements of this set appear as the possible state of the automaton. The following recursive solution of the state observation problem produces this probability distribution.

The probability of the state sequence (11.84) for the I/O pair (11.71), (11.72) is denoted by

$$\begin{aligned}&\text{Prob}(Z(k_\mathrm{e}) = z_{k_\mathrm{e}}, \ldots, Z(0) = z_0 \mid \\ &\quad V(k_\mathrm{e}) = \bar{v}_{k_\mathrm{e}}, \ldots, V(0) = \bar{v}_0,\ W(k_\mathrm{e}) = \bar{w}_{k_\mathrm{e}}, \ldots, W(0) = \bar{w}_0).\end{aligned}$$

As this notation is rather complex, this probability will be abbreviated as

$$\text{Prob}(Z(0 \ldots k_\mathrm{e}) \mid V(0 \ldots k_\mathrm{e}), W(0 \ldots k_\mathrm{e}))$$

in the future with similar notations for other probabilities. The following lemma shows how this probability can be determined.

Lemma 11.6 (Probability distribution of the state sequence) *Consider a stochastic automaton with initial state probability distribution $p_0(z)$ and a consistent I/O pair (V, W). Then*

$$\begin{aligned}&\text{Prob}(Z(0 \ldots k_\mathrm{e}) \mid V(0 \ldots k_\mathrm{e}), W(0 \ldots k_\mathrm{e})) \\ &\quad = \frac{\sum\limits_{z_{k_\mathrm{e}+1}} \prod_{k=0}^{k_\mathrm{e}} L(z_{k+1}, \bar{w}_k \mid z_k, \bar{v}_k) \cdot p_0(z_0)}{\sum\limits_{Z(0 \ldots k_\mathrm{e}+1)} \prod_{k=0}^{k_\mathrm{e}} L(z_{k+1}, \bar{w}_k \mid z_k, \bar{v}_k) \cdot p_0(z_0)}\end{aligned} \tag{11.87}$$

describes the probability that the stochastic automaton has generated the state sequence $Z(0 \ldots k_\mathrm{e})$.

Note that the denominator in Eq. (11.87) does not vanish because the I/O pair is assumed to be consistent and, hence, Eq. (11.86) holds. In the application of Lemma 11.6 first the inequality (11.86) can be checked to find out whether the I/O pair is consistent with the automaton and if this test is successful Eq. (11.87) is applied to determine the probability distribution

$$\text{Prob}(Z(0 \ldots k_\mathrm{e}) \mid V(0 \ldots k_\mathrm{e}), W(0 \ldots k_\mathrm{e}))$$

for all state sequences $Z(0 \ldots k_\mathrm{e}) \in \mathcal{Z}^*$.

Proof According to Bayes' formula, the relation

$$\begin{aligned}&\text{Prob}(Z(0\dots k_\mathrm{e}) \mid V(0\dots k_\mathrm{e}), W(0\dots k_\mathrm{e})) \\ &= \frac{\text{Prob}(Z(0\dots k_\mathrm{e}), V(0\dots k_\mathrm{e}), W(0\dots k_\mathrm{e}))}{\text{Prob}(V(0\dots k_\mathrm{e}), W(0\dots k_\mathrm{e}))}\end{aligned} \tag{11.88}$$

holds, where the inequality $\text{Prob}(V(0\dots k_\mathrm{e}), W(0\dots k_\mathrm{e})) > 0$ holds because the pair (V, W) is assumed to be consistent with the stochastic automaton. Equation (11.88) can be reformulated as

$$\begin{aligned}&\text{Prob}(Z(0\dots k_\mathrm{e}) \mid V(0\dots k_\mathrm{e}), W(0\dots k_\mathrm{e})) \\ &= \frac{\sum\limits_{z_{k_\mathrm{e}+1}} \text{Prob}(z_0, \dots, z_{k_\mathrm{e}+1}, \bar{v}_0, \dots, \bar{v}_{k_\mathrm{e}}, \bar{w}_0, \dots, \bar{w}_{k_\mathrm{e}})}{\sum\limits_{Z(0\dots k_\mathrm{e}+1)} \text{Prob}(z_0, \dots, z_{k_\mathrm{e}+1}, \bar{v}_0, \dots, \bar{v}_{k_\mathrm{e}}, \bar{w}_0, \dots, \bar{w}_{k_\mathrm{e}})}.\end{aligned} \tag{11.89}$$

The probability distribution that appears in the numerator and denominator of (11.89) can be simplified as follows:

$$\text{Prob}(z_0, \dots, z_{k_\mathrm{e}+1}, \bar{v}_0, \dots, \bar{v}_{k_\mathrm{e}}, \bar{w}_0, \dots, \bar{w}_{k_\mathrm{e}}) = \tag{11.90}$$

$$\begin{aligned}&\text{Prob}(z_{k_\mathrm{e}+1}, \bar{w}_{k_\mathrm{e}} \mid z_{k_\mathrm{e}}, \bar{v}_{k_\mathrm{e}}, Z(0\dots k_\mathrm{e}-1), V(0\dots k_\mathrm{e}-1), W(0\dots k_\mathrm{e}-1)) \cdot \\ &\cdot \text{Prob}(z_{k_\mathrm{e}}, \bar{v}_{k_\mathrm{e}}, Z(0\dots k_\mathrm{e}-1), V(0\dots k_\mathrm{e}-1), W(0\dots k_\mathrm{e}-1)) =\end{aligned} \tag{11.91}$$

$$\begin{aligned}&\text{Prob}(z_{k_\mathrm{e}+1}, \bar{w}_{k_\mathrm{e}} \mid z_{k_\mathrm{e}}, \bar{v}_{k_\mathrm{e}}) \cdot \\ &\cdot \text{Prob}(z_0, \dots, z_{k_\mathrm{e}}, \bar{v}_0, \dots, \bar{v}_{k_\mathrm{e}}, \bar{w}_0, \dots, \bar{w}_{k_\mathrm{e}-1})\end{aligned} \tag{11.92}$$

$$= \cdots =$$

$$\begin{aligned}&\text{Prob}(z_{k_\mathrm{e}+1}, \bar{w}_{k_\mathrm{e}} \mid z_{k_\mathrm{e}}, \bar{v}_{k_\mathrm{e}}) \cdot \\ &\cdot \text{Prob}(z_{k_\mathrm{e}}, \bar{w}_{k_\mathrm{e}-1} \mid z_{k_\mathrm{e}-1}, \bar{v}_{k_\mathrm{e}-1}) \cdot \\ &\cdot \dots \cdot \text{Prob}(z_1, \bar{w}_0 \mid z_0, \bar{v}_0) \cdot \text{Prob}(z_0, \bar{v}_0, \dots, \bar{v}_{k_\mathrm{e}}) =\end{aligned} \tag{11.93, 11.94}$$

$$\prod_{k=0}^{k_\mathrm{e}} L(z_{k+1}, \bar{w}_k \mid z_k, \bar{v}_k) \cdot p_0(z_0) \cdot \text{Prob}(\bar{v}_0, \dots, \bar{v}_{k_\mathrm{e}}). \tag{11.95}$$

To obtain Eq. (11.91), Bayes' formula was used. The first probability in (11.91) can be reformulated as (11.92) due to the Markov property (11.27). The second probability distribution in (11.91) and (11.92) has a similar form as (11.90). Only the state z and output w with the highest time index have disappeared in the list of arguments. Hence, the same simplification steps can be carried out several times until Eq. (11.94) is obtained. As z_0 is independent of $\bar{v}_0, \dots, \bar{v}_{k_\mathrm{e}}$ and

$$\text{Prob}(Z(k+1) = z(k+1), W(k) = \bar{w}_k \mid Z(k) = z_k, V(k) = \bar{v}_k) = L(z_{k+1}, \bar{w}_k \mid z_k, \bar{v}_k)$$

represents the behavioural relation of the stochastic automaton, Eq. (11.95) is obtained. Finally, (11.87) results from inserting (11.95) into (11.89) and simplifying the resulting expression. □

From Lemma 11.6, the solution to the observation problem is obtained by determining the conditional probability distributions

$$\begin{aligned} p(z, k_\mathrm{e}) = \mathrm{Prob}(Z(k_\mathrm{e}) &= z | V(0) = v_0, V(1) = v_1, \ldots, V(k_\mathrm{e}) = v_{k_\mathrm{e}}, \\ &\quad W(0) = w_0, W(1) = w_1, \ldots, W(k_\mathrm{e}) = w_{k_\mathrm{e}}) \\ p'(z', k_\mathrm{e}) = \mathrm{Prob}(Z(k_\mathrm{e}+1) &= z' | V(0) = v_0, V(1) = v_1, \ldots, V(k_\mathrm{e}) = v_{k_\mathrm{e}}, \\ &\quad W(0) = w_0, W(1) = w_1, \ldots, W(k_\mathrm{e}) = w_{k_\mathrm{e}}). \end{aligned}$$

They replace the binary indicators $p(z, k_\mathrm{e})$ and $p'(z', k_\mathrm{e})$ defined in Eq. (11.76), which have been used to show whether or not the state $z, z' \in \mathcal{Z}$ can be assumed by the automaton $\mathcal{N}$ at time k_e or $k_\mathrm{e}+1$ if the automaton received the input sequence $V(0 \ldots k_\mathrm{e})$ and generated the output sequence $W(0 \ldots k_\mathrm{e})$.

The results are summarised in the following theorem, which is a direct consequence of Lemma 11.6:

Theorem 11.4 (Solution to the observation problem) *Consider a stochastic automaton with the initial state probability distribution $p_0(z)$. If the I/O pair (V, W) is consistent with the automaton, the current state probability distribution is given by*

$$p(z_{k_\mathrm{e}}, k_\mathrm{e}) = \frac{\sum\limits_{Z(0\ldots k_\mathrm{e}-1)} \sum\limits_{z_{k_\mathrm{e}+1}} \prod_{k=0}^{k_\mathrm{e}} L(z_{k+1}, \bar{w}_k \mid z_k, \bar{v}_k) \cdot p_0(z_0)}{\sum\limits_{Z(0\ldots k_\mathrm{e}+1)} \prod_{k=0}^{k_\mathrm{e}} L(z_{k+1}, \bar{w}_k \mid z_k, \bar{v}_k) \cdot p_0(z_0)} \tag{11.96}$$

and the set of current automaton states by

$$\mathcal{Z}(k_\mathrm{e} \mid k_\mathrm{e}) = \{z_{k_\mathrm{e}} \,:\, p(z_{k_\mathrm{e}}, k_\mathrm{e}) > 0\}. \tag{11.97}$$

Note that for time $k_\mathrm{e} = 0$ Eq. (11.96) yields the a-posteriori probability distribution $p(z, 0), (z \in \mathcal{Z})$ that the automaton is in the initial state z and has generated the output

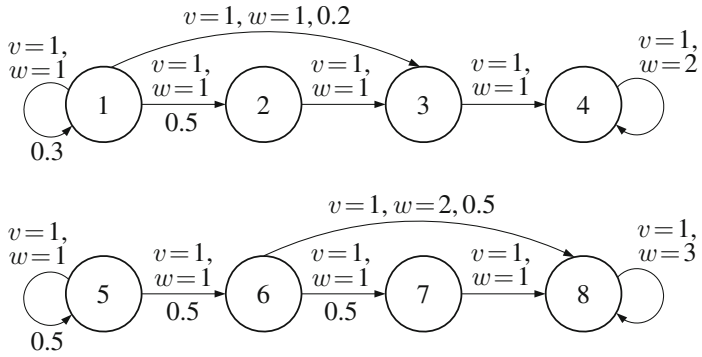


Fig. 11.26 Automaton graph of the example

$\bar{w}_0$ for the input $\bar{v}_0$. This probability is, in general, different from $p_0(z)$, which represents the a-priori probability distribution of the initial state. Hence, the set

$$\mathcal{Z}(0 \mid 0) = \{z \in \mathcal{Z} \,:\, p(z, 0) > 0\}$$

is different from the a-priori information about the initial state represented by

$$\mathcal{Z}(0 \mid -1) = \{z \in \mathcal{Z} \,:\, p_0(z) > 0\} \tag{11.98}$$

and satisfies the relation

$$\mathcal{Z}(0 \mid 0) \subseteq \mathcal{Z}(0 \mid -1).$$

Example 11.5 State observation of a stochastic automaton
Consider the stochastic automaton whose automaton graph is shown in Fig. 11.26 and whose initial state is uniformly distributed:

$$p_0(z) = \frac{1}{8} \quad z = 1, \ldots, 8.$$

The value of the behavioural relation L is indicated at the corresponding edges of the automaton graph unless its value is 1 for deterministic state transitions. The automaton graph

Table 11.1 Probability distribution $\text{Prob}(Z \mid V, W)$ of the example automaton

$k_e = 0$ $V(0...0) = (1)$ $W(0...0) = (1)$		$k_e = 1$ $V(0...1) = (1,1)$ $W(0...1) = (1,1)$		$k_e = 2$ $V(0...2) = (1,1,1)$ $W(0...2) = (1,1,1)$	
$Z(0..0)$	$\text{Prob}(Z\|V,W)$	$Z(0..1)$	$\text{Prob}(Z\|V,W)$	$Z(0..2)$	$\text{Prob}(Z\|V,W)$
(1)	0.1818	(1,1)	0.0923	(1,1,1)	0.0632
(2)	0.1818	(1,2)	0.1538	(1,1,2)	0.1053
(3)	0.1818	(1,3)	0.0615	(1,1,3)	0.0421
(5)	0.1818	(2,3)	0.3077	(1,2,3)	0.3509
(6)	0.0909	(5,5)	0.1538	(5,5,5)	0.1754
(7)	0.1818	(5,6)	0.0769	(5,5,6)	0.0877
		(6,7)	0.1538	(5,6,7)	0.1754
#1	#2	#3	#4	#5	#6

$k_e = 3$ $V(0...3) = (1,1,1,1)$ $W(0...3) = (1,1,1,3)$		$k_e = 4$ $V(0...4) = (1,1,1,1,1)$ $W(0...4) = (1,1,1,3,3)$	
$Z(0...3)$	$\text{Prob}(Z \mid V, W)$	$Z(0...4)$	$\text{Prob}(Z \mid V, W)$
(5,6,7,8)	1	(5,6,7,8,8)	1
#7	#8	#9	#10

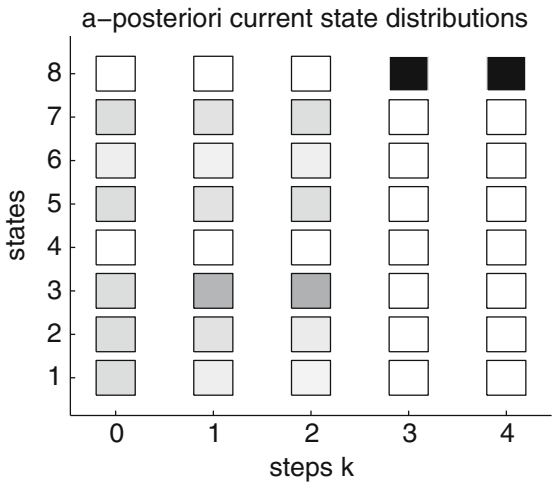


Fig. 11.27 Observation result

consists of two separate parts and it is interesting to see how the probability $p(z, k_e)$ distributes over these two parts for increasing k_e.

All state sequences that occur with non-vanishing probability for the input sequence

$$V(0\ldots4) = (1, 1, 1, 1, 1)$$

and yield the output sequence

$$W(0\ldots4) = (1, 1, 1, 3, 3)$$

are shown in Table 11.1 together with the probabilities

$$\begin{aligned} &\text{Prob}(z_0 \mid \bar{v}_0, \bar{w}_0), \quad \text{Prob}\,(Z(0\ldots1) \mid V(0\ldots1), W(0\ldots1)), \\ &\ldots, \quad \text{Prob}(Z(0\ldots4) \mid V(0\ldots4), W(0\ldots4)), \end{aligned}$$

which are obtained by means of Eq. (11.87). Note that the value of

$$\text{Prob}(Z(0\ldots0) \mid V(0\ldots0), W(0\ldots0)) = \text{Prob}(z_0 \mid \bar{v}_0, \bar{w}_0)$$

which is shown in column #2 differs from the a-priori probability distribution $p_0(z) = \frac{1}{8}$ because this a-posteriori probability includes the information provided by the I/O pair $(\bar{v}_0, \bar{w}_0)$. This is the reason why the states $Z(0) = 4$ and $Z(0) = 8$, both of which are assumed by the automaton with the a-priori probability $p_0(z) = \frac{1}{8}$ do not appear in column #1.

The state probability distribution obtained by means of Eq. (11.96) is shown in Fig. 11.27. The probabilities are depicted in greyscale. Black rectangles symbolise a probability of one, white rectangles a probability of zero. The set $\mathcal{Z}(k_e \mid k_e)$ includes all states z_{k_e} with non-zero probability (grey and black boxes):

$$\begin{aligned}\mathcal{Z}(0\,|\,0) &= \{1,2,3,5,6,7\}\\ \mathcal{Z}(1\,|\,1) &= \{1,2,3,5,6,7\}\\ &\vdots\\ \mathcal{Z}(4\,|\,4) &= \{8\}. \quad \square\end{aligned}$$

Recursive form of the solution. For the application, the elements of the sequences $V(0\ldots k_\mathrm{e})$ and $W(0\ldots k_\mathrm{e})$ appear one after the other for $k_\mathrm{e} = 0, 1, 2, \ldots$ and should be processed in this way. Therefore, the following recursive form of the solution to the state observation problem is important (for a proof cf. [218]). In the representation given, the indicator $p(z, k_\mathrm{e})$ denotes the probability

$$p(z, k_\mathrm{e}) = \mathrm{Prob}(Z(k_\mathrm{e}) = z \,|\, V(0\ldots k_\mathrm{e}),\, W(0\ldots k_\mathrm{e}))$$

that z is the state at time k_e and $p'(z', k_\mathrm{e})$ the probability

$$p'(z', k_\mathrm{e}) = \mathrm{Prob}(Z(k_\mathrm{e}+1) = z' \,|\, V(0\ldots k_\mathrm{e}),\, W(0\ldots k_\mathrm{e}))$$

for z' to be the next state.

Theorem 11.5 (Recursive solution to the state observation problem) *Consider a stochastic automaton with the initial state distribution $p_0(z)$. If the I/O pair (V, W) is consistent with the stochastic automaton, the a-posteriori state probability distribution is given by the recursive relations*

$$p(z, k_\mathrm{e}) = \frac{\sum\limits_{z'} L(z', \bar{w}_{k_\mathrm{e}} \,|\, z, \bar{v}_{k_\mathrm{e}}) \cdot p'(z', k_\mathrm{e}-1)}{\sum\limits_{z',\tilde{z}} L(z', \bar{w}_{k_\mathrm{e}} \,|\, \tilde{z}, \bar{v}_{k_\mathrm{e}}) \cdot p'(z', k_\mathrm{e}-1)}, \quad (k_\mathrm{e} \geq 0) \qquad (11.99)$$

with

$$p'(z', k_\mathrm{e}) = \frac{\sum\limits_{z} L(z', \bar{w}_{k_\mathrm{e}} \,|\, z, \bar{v}_{k_\mathrm{e}}) \cdot p'(z, k_\mathrm{e}-1)}{\sum\limits_{z,\tilde{z}} L(\tilde{z}, \bar{w}_{k_\mathrm{e}} \,|\, z, \bar{v}_{k_\mathrm{e}}) \cdot p'(z, k_\mathrm{e}-1)}, \quad (k_\mathrm{e} > 0) \qquad (11.100)$$

$$p'(z', -1) = p_0(z'), \qquad (k_\mathrm{e} = 0). \quad (11.101)$$

Equation (11.100) is used after the pair $(V(k_\mathrm{e}) = \bar{v}_{k_\mathrm{e}}, W(k_\mathrm{e}) = \bar{w}_{k_\mathrm{e}})$ has been measured. It describes the probability distribution of the future state $Z(k_\mathrm{e}+1) = z'$ by using the information about the movement of the stochastic automaton until time k_e that is included in the I/O pair $(V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e}))$. It is a recursive relation with the initialisation given by Eq. (11.101). Equation (11.100) makes it possible to determine $p'(z', k_\mathrm{e})$, $(z' \in \mathcal{Z})$ for given $p'(z, k_\mathrm{e}-1)$, $(z \in \mathcal{Z})$ and the new measurements $(\bar{v}_{k_\mathrm{e}}, \bar{w}_{k_\mathrm{e}})$. Hence, only the probability distribution $p'(z, k_\mathrm{e}-1)$, $(z \in \mathcal{Z})$ has to be stored in the computer memory, which consists of N values.

Equation (11.99) describes how the prediction from the previous time point has to be corrected after the new measurements $\bar{v}_{k_e}$ and $\bar{w}_{k_e}$ became available. This step can be interpreted as a projection onto the set of those states $Z(k_e) = z$ from which the automaton can have moved when generating the new measurement information $W(k_e) = \bar{w}_{k_e}$. The result of the recursion is the a-posteriori probability distribution $p(z, k_e)$ over all current states $Z(k_e) = z \in \mathcal{Z}$ for the given measurements until time k_e. With this structure, the recursive solution to the state observation problem shows a remarkable similarity to the Kalman filter, which likewise can be decomposed into a prediction and a projection step.

A-priori knowledge about the initial state. In the solution to the state observation problem, the initial state probability distribution $p_0(z)$ is assumed to be known. Since in applications this a-priori knowledge is often not available, $p_0(z)$ is measured or "guessed". The question arises what happens if the a-priori knowledge about z_0 is in conflict with the actual initial state of the stochastic process.

To answer this question, assume that $\hat{p}_0(z)$ denotes the approximate initial state probability distribution and consider the sets

$$\mathcal{Z}_0 = \{z : p_0(z) > 0\} \subseteq \mathcal{Z} \tag{11.102}$$

$$\hat{\mathcal{Z}}_0 = \{z : \hat{p}_0(z) > 0\} \subseteq \mathcal{Z}. \tag{11.103}$$

The stochastic process starts from an initial state $z_0 \in \mathcal{Z}_0$, whereas the observation algorithm assumes that the automaton starts from some state $z_0 \in \hat{\mathcal{Z}}_0$. The a-priori knowledge about the initial state is not in conflict with the real system, if the relation

$$\hat{\mathcal{Z}}_0 \supseteq \mathcal{Z}_0 \tag{11.104}$$

holds true. Then, the solution to the observation problem ensures that the relation

$$Z(k_e) \in \mathcal{Z}(k_e \mid k_e)$$

is valid for all $k_e \geq 0$, i.e. the set of current states determined by the observation algorithm includes the true state of the stochastic process. If the probability distribution $\hat{p}_0(z)$ used in the observation algorithm is different from the real distribution $p_0(z)$, the probability distribution $p(z, k_e)$ obtained by the algorithm is wrong, but it is accepted in practice as solution to the observation problem due to the lack of a better a-priori knowledge.

If, however, condition (11.104) is violated, then it is possible that the probability $\text{Prob}(W(0 \ldots k_e) \mid V(0 \ldots k_e), z_0)$ is zero for all initial states $z_0 \in \hat{\mathcal{Z}}_0$. Consequently, like in the case of an inconsistent I/O pair, the violation of the condition (11.104) makes the denominators in Eqs. (11.99) and (11.100) vanish, which can be used as an indicator to stop the observation algorithm.

This and the preceding remark show that the observation algorithm cannot distinguish between an inconsistent I/O pair and a wrong initial state probability distribution. As a consequence, in an application the set $\hat{\mathcal{Z}}_0$ has to be chosen "large

enough". A secure way is to choose $\hat{p}_0(z)$ so that $\hat{\mathcal{Z}}_0 = \mathcal{Z}$ holds, for example, using the uniform initial state distribution

$$\hat{p}(z) = \text{Prob}(Z(0) = z) = \frac{1}{N} \quad \text{for all} \quad z \in \mathcal{Z}. \tag{11.105}$$

Besides the lack of knowledge of $p_0(z)$ another fact makes it reasonable to use the a-priori probability distribution (11.105). For many stochastic automata, the solution $p(z, k_e)$ of the observation problem is (nearly) independent of $p_0(z)$ for $k_e \geq \bar{k}$ with very small $\bar{k}$. This is particularly true if the set $Z(k_e \mid k_e)$ has only a few elements compared to the cardinality N of the state set $\mathcal{Z}$.

11.6.2 Observation Algorithm

To show how the observation method developed in this section can be applied online, this section presents an observation algorithm, which is based on the recursive solution given in Theorem 11.5. The following symbols are used in the algorithm:

$$\begin{aligned} h(z) &= \sum_{z'} L(z', \bar{w}_{k_e} \mid z, \bar{v}_{k_e}) \cdot p'(z', k_e - 1) \\ p(z) &= \text{Prob}(Z(k_e) = z \mid V(0 \ldots k_e), W(0 \ldots k_e)) \\ p'(z') &= \text{Prob}(Z(k_e + 1) = z' \mid V(0 \ldots k_e), W(0 \ldots k_e)) \end{aligned}$$

Algorithm 11.6 *Observation of stochastic automata*

Given: Stochastic automaton $\mathcal{S}$

Initial state probability distribution $p_0(z)$.

Initialisation: $p'(z) = p_0(z)$ for all $z \in \mathcal{Z}$

$k_e = 0$.

Loop:

1. Measure the current input $\bar{v}$ and output $\bar{w}$.
2. Determine $h(z) = \sum_{z'} L(z', \bar{w} \mid z, \bar{v}) \cdot p'(z')$ for all $z \in \mathcal{Z}$
3. If $\sum_z h(z) = 0$ holds, stop the algorithm (inconsistent I/O pair or wrong initial state distribution).
4. Determine $p(z) := \dfrac{h(z)}{\sum_z h(z)}$ for all $z \in \mathcal{Z}$.

5. Determine $p'(z') := \dfrac{\sum_z L(z', \bar{w} \mid z, \bar{v})\, p'(z)}{\sum_z h(z)}$ for all $z \in \mathcal{Z}$.

6. Determine $\mathcal{Z}(k_e \mid k_e)$ according to Eq. (11.97).

7. $k_e := k_e + 1$
   Continue with Step 1.

Result: $p(z) = \text{Prob}(Z(k_e) = z \mid V(0 \ldots k_e), W(0 \ldots k_e))$ and $\mathcal{Z}(k_e \mid k_e)$ for increasing time horizon k_e.

In Step 3, the consistency of the I/O pair and of the a-priori probability distribution is tested. Steps 4 and 5 use Eq. (11.99) or (11.100), respectively. The test in Step 3 ensures that the denominators of both equations do not vanish.

Note that in each step of the algorithm very easy calculations have to be carried out, which makes the algorithm applicable under relatively strong real-time constraints. Since the algorithm is based on the recursive solution to the state observation problem, its complexity does *not* increase with the length of the measurement sequences V and W. Only the N values of the functions $h(z)$ and $p'(z)$ have to be stored in the memory.

11.6.3 Observability of Stochastic Automata

In this section, a notion of observability is introduced, which takes into account that for stochastic automata the state generally cannot be determined unambiguously. Even if the initial state is known, the automaton may produce an I/O pair that does not allow to track the state trajectory with certainty.

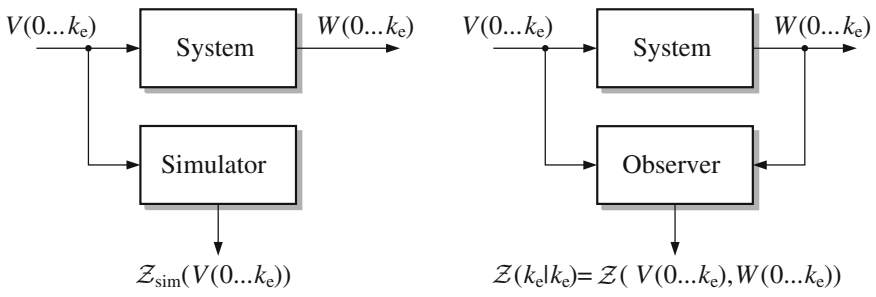


Fig. 11.28 Comparison of simulation and observation

The observability definition is based on a comparison of the results that are obtained by means of simulation and of observation (Fig. 11.28). Roughly speaking, the stochastic automaton is called observable if it is possible to determine the state more precisely by state observation than by simulation. Before defining the observability in this way, simulation and observation have to be compared in more detail.

For both simulation and observation, the initial state distribution $p_0(z)$ and the input sequence V have to be known.

- In **simulation**, the initial state probability distribution is propagated according to the state transition relation G of the stochastic automaton and yields the state probability distribution $\text{Prob}(Z(k_e) = z \mid V(0 \ldots k_e - 1))$, $(z \in \mathcal{Z})$ of the state at the end of the time interval considered. Accordingly, the set of states in which the stochastic automaton recides at time k_e with non-vanishing probability for the given the input sequence $V(0 \ldots k_e)$ is

$$\mathcal{Z}_{\text{sim}}(V(0 \ldots k_e)) = \{z : \text{Prob}(Z(k_e) = z \mid V(0 \ldots k_e)) > 0\}. \tag{11.106}$$

- In **state observation** described by Theorem 11.5 the additional information included in the output sequence W is used to determine the probability

$$\text{Prob}(Z(k_e) = z \mid V(0 \ldots k_e), W(0 \ldots k_e)), \quad z \in \mathcal{Z}$$

  according to Eq. (11.96). The set of states $\mathcal{Z}(k_e \mid k_e)$ is obtained by Eq. (11.97).

As the state observation uses the generated output sequence as additional constraint when determining the set of possible states $Z(k_e)$, the relation

$$\mathcal{Z}_{\text{sim}}(V(0 \ldots k_e)) \supseteq \mathcal{Z}(k_e \mid k_e)$$

holds.

Stochastic unobservability. An automaton is called *unobservable* if for all input sequences V the I/O pair (V, W) does not include more information than the input V alone. This definition can be stated more formally as follows:

Definition 11.4 (*Stochastic unobservability*) A stochastic automaton is called stochastically unobservable if the behavioural relation L can be represented as the product of two functions

$$G : \mathcal{Z} \times \mathcal{Z} \times \mathcal{V} \to [0, 1]$$
$$H : \mathcal{W} \times \mathcal{V} \to [0, 1]$$

such that

$$L(z', w \mid z, v) = G(z' \mid z, v) \cdot H(w \mid v) \tag{11.107}$$

holds for all $z', z \in \mathcal{Z}$, $v \in \mathcal{V}$ and $w \in \mathcal{W}$.

As before, the functions $G(z', z, v)$ and $H(w, v)$ are denoted by $G(z' \mid z, v)$ or $H(w \mid v)$, respectively, because they turn out to be conditional probabilities.

Definition 11.4 has an obvious interpretation. Equation (11.107) says that the output w does not depend on z and, hence, does not provide any information about the current automaton state. Furthermore Eq. (11.107) implies that w does not depend on z' and, hence, the output does not provide any information about the successor state either.

Observability test. Before the consequences of Eq. (11.107) will be discussed it should be mentioned how this definition can be used to test whether a stochastic automaton is stochastically unobservable. It is easy to see that the functions G and H used in Eq. (11.107) are the conditional probabilities defined by Eqs. (11.19) and (11.20). From this fact, it is clear that the decomposition of L in the form (11.107) is found (if it exists) by determining G and H according to Eqs. (11.19) and (11.20) and by testing whether Eq. (11.107) holds. Note that the function H obtained from Eq. (11.20) is not allowed to depend upon z, but $H(w \mid z_i, v) = H(w \mid z_j, v)$ has to hold for all $z_i, z_j \in \mathcal{Z}$, $v \in \mathcal{V}$ and $w \in \mathcal{W}$.

Property of unobservable automata. The following lemma says that the definition of unobservability satisfies the aim to call an automaton observable if the current state can be determined more precisely by state observation than by simulation.

Lemma 11.7 [306] *If the stochastic automaton is stochastically unobservable, then for all consistent I/O pairs (V, W) the results of the state observation problem and of the simulation are identical for all $k_e = 0, 1, 2, \ldots$:*

$$\begin{aligned} \text{Prob}(Z(k_e) = z \mid V(0 \ldots k_e), W(0 \ldots k_e)) &= \text{Prob}(Z(k_e) = z \mid V(0 \ldots k_e)) \\ \mathcal{Z}(k_e \mid k_e) &= \mathcal{Z}_{\text{sim}}(V(0 \ldots k_e)). \end{aligned}$$

Hence, the output sequence W does not include any information about the state trajectory of the automaton that cannot be obtained from the initial state probability $p_0(z)$ and the input sequence V. In this case, the observers described by Theorems 11.4 and 11.5 work as simulators.

Stochastic unobservability of a state set. The stochastic automaton may not satisfy the condition (11.107) for all $z \in \mathcal{Z}$, but only for a subset $\mathcal{G}_z \subseteq \mathcal{Z}$, where $\mathcal{G}_z$ should have at least two elements ($|\mathcal{G}_z| \geq 2$). Then a stochastic automaton is called *stochastically unobservable within a set $\mathcal{G}_z$* if the relation

$$L_{\mathcal{G}_z}(z', w \mid z, v) = G_{\mathcal{G}_z}(z' \mid z, v) \cdot H_{\mathcal{G}_z}(w \mid v) \tag{11.108}$$

holds for all $z, z' \in \mathcal{G}_z$, $v \in \mathcal{V}$, and $w \in \mathcal{W}$, with

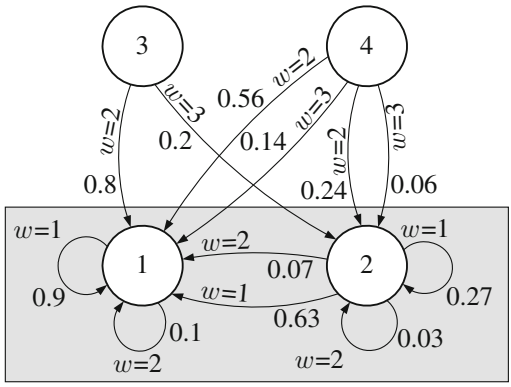


Fig. 11.29 Stochastic automaton with stochastically unobservable set {1, 2}

$$L_{\mathcal{G}_z}(z', w \mid z, v) = \frac{L(z', w \mid z, v)}{\sum_{z' \in \mathcal{G}_z} \sum_{w} L(z', w \mid z, v)} \tag{11.109}$$

$$G_{\mathcal{G}_z}(z' \mid z, v) = \sum_{w} L_{\mathcal{G}_z}(z', w \mid z, v) \tag{11.110}$$

$$H_{\mathcal{G}_z}(w \mid v) = \sum_{z' \in \mathcal{G}_z} L_{\mathcal{G}_z}(z', w \mid z, v). \tag{11.111}$$

In this definition, the functions L, G and H appearing in Definition 11.4 are replaced by $L_{\mathcal{G}_z}$, $G_{\mathcal{G}_z}$ or $H_{\mathcal{G}_z}$, respectively. These functions are normalised as shown in Eqs. (11.109)–(11.111) so as to define conditional probability distributions. For these new functions, Eq. (11.108) is, in principle, the same as Eq. (11.107), but it has only to be satisfied for the states z, z' that belong to the set $\mathcal{G}_z$. Lemma 11.7 holds as long as the automaton remains within the set $\mathcal{G}_z$:

$$z(k) \in \mathcal{G}_z \quad \text{for } k = 0, 1, \ldots, k_e.$$

If the automaton is stochastically unobservable within the set $\mathcal{G}_z$ then the state observer acts as a simulator as long as the state remains in the set $\mathcal{G}_z$.

Example 11.6 Observability of stochastic automata
Consider the stochastic automaton shown in Fig. 11.29, which has the only input $v = 1$. The automaton is not stochastically unobservable according to Definition 11.4. However, the state set $\mathcal{G}_z = \{1, 2\}$ is stochastically unobservable. This can be verified by means of Eq. (11.107) as shown in Table 11.2. Note that $G_{\mathcal{G}_z} \cdot H_{\mathcal{G}_z}$ is identical to $L_{\mathcal{G}_z}$ for all z, z', w and v. □

Stochastic observability. The stochastic observability can now be defined as the property of an automaton not to possess sets of unobservable states:

Definition 11.5 (*Stochastic observability*) A stochastic automaton is called stochastically observable if it does not possess any set $\mathcal{G}_z$ of stochastically unobservable states.

Table 11.2 Test for stochastic unobservability of the state set $\mathcal{G}_z = \{1, 2\}$

		$L_{\mathcal{G}_z}$		$G_{\mathcal{G}_z}$		$H_{\mathcal{G}_z}$		$G_{\mathcal{G}_z} \cdot H_{\mathcal{G}_z}$	
		$z=1$	$z=2$	$z=1$	$z=2$	$z=1$	$z=2$	$z=1$	$z=2$
$z'=1$	$w=1$	0.9	0.63	1.0	0.7	0.9	0.9	0.9	0.63
	$w=2$	0.1	0.07			0.1	0.1	0.1	0.07
	$w=3$	0	0			0	0	0	0
$z'=2$	$w=1$	0	0.27	0	0.3	0.9	0.9	0	0.27
	$w=2$	0	0.03			0.1	0.1	0	0.03
	$w=3$	0	0			0	0	0	0

Consequently, the stochastic automaton is called observable if it is possible to determine every state more precisely by state observation than by simulation. This fact is represented by the following corollary, which follows directly from Lemma 11.7 and Definition 11.5.

Corollary 11.2 *If the stochastic automaton is stochastically observable, the following relation holds:*

$$\mathcal{Z}(k_\mathrm{e} \mid k_\mathrm{e}) \subseteq \mathcal{Z}_\mathrm{sim}(V(0 \ldots k_\mathrm{e})), \quad (k_\mathrm{e} \geq 0). \tag{11.112}$$

Note that the equality sign may hold for some input sequence even if the automaton is observable. The reason for this is given in Sect. 11.6.4.

Remark 11.1 (*Observability test*) The following remarks concern the test of a given stochastic automaton concerning observability. The definition of unobservable state sets implies that if the set $\mathcal{G}_z$ is stochastically unobservable then any subset $\tilde{\mathcal{G}}_z \subset \mathcal{G}_z$ is stochastically unobservable as well, because Eq. (11.108) holds not only for all $z, z' \in \mathcal{G}_z$ but also for all $z, z' \in \tilde{\mathcal{G}}_z$ and the normalisation carried out in Eqs. (11.109)–(11.111) does not influence this result. Hence, the test of a stochastic automaton starts with testing all pairs $z, \bar{z}$ whether or not they are unobservable sets. If no such pair is found, the stochastic automaton does not possess any unobservable set and is, therefore, stochastically observable. If such pairs exist, larger unobservable state sets can be obtained (if they exist) from combinations of such unobservable pairs according to the following corollary. □

Corollary 11.3 *A stochastic automaton is stochastically unobservable within a set $\mathcal{G}_z$ of at least three states ($|\mathcal{G}_z| \geq 3$) if and only if the stochastic automaton is stochastically unobservable within all subsets $\mathcal{G}_z^i \subset \mathcal{G}_z$ of two states ($|\mathcal{G}_z^i| = 2$).*

Therefore, the search for stochastically unobservable sets of states can be reduced to the test of all pairs of states. The stochastic automaton is stochastically observable if no unobservable pair of states is found.

11.6.4 Distinguishing Inputs

In this section, it is investigated under what conditions the observer improves its result by processing the k_eth I/O pair $(\bar{v}_{k_e}, \bar{w}_{k_e})$ in comparison with the result obtained for the I/O pair of length $k_e - 1$. This analysis uses the recursive formulation of the observer given in Theorem 11.5 and compares the observation result with the simulation results obtained by the recursion (11.26). It will be shown that even if the automaton is observable there exist input values v for which the next recursion step of the observer yields the same result as the next recursion step of the simulator. Hence, the movement of the automaton under this input cannot contribute to an improvement of the knowledge about the current state.

To start the analysis, it is assumed that both simulation and observation have obtained the same state probability distribution at time k_e

$$\begin{aligned}&\text{Prob}(Z(k_e)=z \mid V(0\ldots k_e-1), W(0\ldots k_e-1)) = \\ &\quad = \text{Prob}(Z(k_e)=z \mid V(0\ldots k_e-1)) \\ &\quad = p'(z \mid k_e-1),\end{aligned}$$

where both observation and simulation have used the information available until time $k_e - 1$. Consequently, both methods yield the same state set at time k_e:

$$\begin{aligned}\mathcal{Z}(k_e-1 \mid k_e-1)) &= \mathcal{Z}_{\text{sim}}(V(0\ldots k_e-1)) \\ &= \{z : p'(z, k_e-1) > 0\}.\end{aligned}$$

The set of successor states that the automaton can reach at time $k_e + 1$ for the input $\bar{v}_{k_e} = \bar{v}$ is given as follows:

$$\mathcal{Z}_{\text{sim}}(V(0\ldots k_e)) = \{z \mid G(z \mid \tilde{z}, \bar{v}) > 0 \text{ for some } \tilde{z} \in \mathcal{Z}_{\text{sim}}(V(0\ldots k_e-1))\}.$$

In the following, the question should be answered under what condition the k_eth observation step yields a set $\mathcal{Z}(k_e \mid k_e)$ which is a proper subset of $\mathcal{Z}_{\text{sim}}(V(0\ldots k_e))$. The answer can be obtained from the considerations made in Sect. 11.6.3. Lemma 11.7 has shown that state observation yields the same result as simulation if the automaton is unobservable; and hence, the behavioural relation can be decomposed according to Eq. (11.107). This result is applied here for the fixed input $\bar{v}$. If the decomposition (11.107) is possible for the input $\bar{v}_{k_e} = \bar{v}$, simulation and observation lead to the same set of states z_{k_e+1}. This result is summarised in the following corollary. It shows that the equality sign can hold in Eq. (11.112).

Corollary 11.4 *Assume that the state probability distribution $p'(z, k_e-1)$ is known and the stochastic automaton generates the output $\bar{w}_{k_e} = \bar{w}$ for the input $\bar{v}_{k_e} = \bar{v}$. Then simulation and state observation yield the same state probability distribution*

$$\begin{aligned} p'(z, k_\mathrm{e}) &= \mathrm{Prob}(Z(k_\mathrm{e}+1)=z \mid V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e})) \\ &= \mathrm{Prob}(Z(k_\mathrm{e}+1)=z \mid V(0\ldots k_\mathrm{e})) \end{aligned}$$

for all $z \in \mathcal{Z}$ if and only if the relation

$$L(z', \bar{w} \mid z, \bar{v}) = G(z' \mid z, \bar{v}) \cdot H(\bar{w} \mid \bar{v}) \tag{11.113}$$

holds with some constant $H(\bar{w} \mid \bar{v})$ for all $z \in \mathcal{Z}(k_\mathrm{e}-1 \mid k_\mathrm{e}-1)$ and $z' \in \mathcal{Z}_\mathrm{sim}(V(0\ldots k_\mathrm{e}))$.

The condition (11.113) can be tested in each recursion step of the observation algorithm in order to indicate whether the observer works really as an observer or merely as a simulator. If the condition is satisfied, the observer yields the same state probability distribution $\mathrm{Prob}(Z(k_\mathrm{e}+1)=z \mid V(0\ldots k_\mathrm{e}))$ as a simulation step described by Eq. (11.26) (with k_e replacing $k_\mathrm{e}-1$).

This result has interesting consequences concerning the choice of the input. Even if the stochastic automaton is observable according to Definition 11.5, not all individual I/O pairs $(\bar{v}, \bar{w})$ lead to an improvement of the observation result compared to the corresponding simulation step. The reason for this is given by the fact that the observability definition claims that the decomposition of the behavioural relation L according to Eq. (11.107) is impossible for all states $z, z' \in \mathcal{Z}$, all input values $v \in \mathcal{V}$ and all output values $w \in \mathcal{W}$. However, such a decomposition may be possible for the set $\tilde{\mathcal{V}} \subset \mathcal{V}$ of input values even if it is impossible for all $v \in \mathcal{V}$. If the stochastic automaton gets, for some reason, only input values from $\tilde{\mathcal{V}}$ the solution of the observation problem is not better than the simulation result (cf. Lemma 11.7), although the stochastic automaton is observable. Therefore, it is important to know, which input should be used in order to get an improved observation result. These input values are called *distinguishing*. For such input values $\bar{v}$ the decomposition (11.107) is impossible for $v = \bar{v}$ and all z', z and w.

As a consequence, for every given state set $\mathcal{G}_z$ it is possible to partition the input set $\mathcal{V}$ into two sets $\tilde{\mathcal{V}}(\mathcal{G}_z)$ and $\bar{\mathcal{V}}(\mathcal{G}_z)$ such that the decomposition of L is possible for all $v \in \tilde{\mathcal{V}}(\mathcal{G}_z)$:

$$\begin{aligned} L_{\mathcal{G}_z}(z', w \mid z, v) &= G_{\mathcal{G}_z}(z' \mid z, v) \cdot H_{\mathcal{G}_z}(w \mid v) \\ &\qquad \text{for all } \; z', z \in \mathcal{G}_z, \; v \in \tilde{\mathcal{V}}(z), \; w \in \mathcal{W}, \end{aligned} \tag{11.114}$$

whereas such a decomposition of L is impossible for all $v \in \bar{\mathcal{V}}(\mathcal{G}_z)$. From Lemma 11.7, it is obvious that the state observer cannot improve its result at time k_e for states $z_{k_\mathrm{e}}, z(k_\mathrm{e}+1) \in \mathcal{G}_z$ if the stochastic automaton obtains the input $\bar{v}_{k_\mathrm{e}} \in \tilde{\mathcal{V}}(\mathcal{G}_z)$. Hence, $\tilde{\mathcal{V}}(\mathcal{G}_z)$ is the set of distinguishing inputs. Only if an input $v \in \bar{\mathcal{V}}(\mathcal{G}_z)$ is applied, the state observer gets enough information for determining the state probability distribution better than it is determined by simulation.

In an application where the state of the system should be found as quickly as possible, the input has to be chosen at every time instant from the current set of distinguishing inputs. This set has to be determined at time k_e as follows. Select

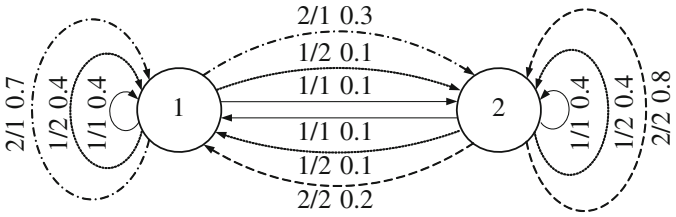


Fig. 11.30 Automaton graph of the example

the set $\mathcal{G}_z$ to include all possible current states $z_{k_e} \in \mathcal{Z}(k_e|k_e - 1)$ and all possible successor states z_{k_e+1}, for which $L(z_{k_e+1}, w|z_{k_e}, v)$ holds for some v and w. Then $\bar{\mathcal{V}}(\mathcal{G}_z)$, which is obtained as described above, is the set of distinguishing inputs form which the current input $V(k_e)$ should be selected. Note that the set $\bar{\mathcal{V}}(\mathcal{G}_z)$ changes from one time step to the next because the set of current state changes. It depends on the practical circumstances whether the input can really be chosen with the aim of state observation or whether the selection of the input has to satisfy other control aims. In any case, the set of distinguishing inputs is the set of preferred input signals as far as state observation is concerned.

If the stochastic automaton is stochastically observable, for every state z there exists at least one distinguishing input v. If, in particular, the decomposition (11.107) is impossible for all $v \in \mathcal{V}$, the stochastic automaton is called *uniformly stochastically observable*. Then, every input is distinguishing and the observation result improves in every recursion step.

Example 11.7 Distinguishing input values
Figure 11.30 shows the automaton graph of a stochastic automaton with two input symbols $v \in \{1, 2\}$ and two output symbols $w \in \{1, 2\}$. Only one output symbol is distinguishing. The edges in Fig. 11.30 are labelled by the I/O pair v/w for which they occur and by the corresponding probability. It can be seen that for $\bar{v} = 1$ the relation (11.113) holds with

$$H(\bar{w}=1 \mid \bar{v}=1) = 0.5 \quad \text{and} \quad H(\bar{w}=2 \mid \bar{v}=1) = 0.5$$

and

$$\begin{aligned} G(z'=1 \mid z=1, \bar{v}=1) &= 0.8, & G(z'=1 \mid z=2, \bar{v}=1) &= 0.2 \\ G(z'=2 \mid z=1, \bar{v}=1) &= 0.2, & G(z'=2 \mid z=2, \bar{v}=1) &= 0.8. \end{aligned}$$

Figures 11.31 and 11.32 show the effect of the non-distinguishing input on the observation result. Starting with a uniform initial state distribution over the whole state set the observation result is shown in Fig. 11.32. It is identical to the simulation result as long as the input is $v = 1$ (time steps $k = 0, 1, \ldots, 7$). Because of the symmetry of $G(z' \mid z, \bar{v})$, the simulation yields uniform distributions for these times steps. Between time steps $k = 8$ and $k = 13$, the distinguishing input $v = 2$ is applied and, hence, the observation result improves immediately. Instead of a uniform probability distribution, Fig. 11.32 shows that the observation results in high probability for the state $z = 2$ at $k = 8, 9$ and for the state $z = 1$ at $k = 10 \ldots 13$.

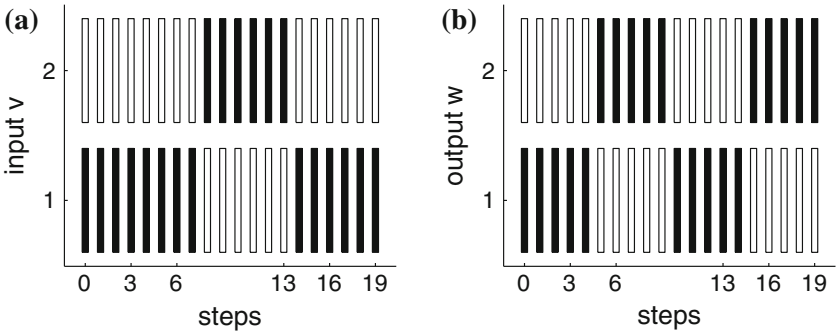
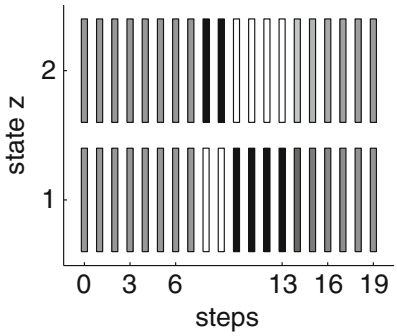


Fig. 11.31 Sequences of input (*left*) and output symbols (*right*)

Fig. 11.32 Observation result



When the input returns to the non-distinguishing input symbol $v = 1$ at time step $k = 14$ the observation falls back into a simulation, quickly loosing all state information. □

11.7 Diagnosis of Stochastic Automata

11.7.1 Principle of Consistency-Based Diagnosis Applied to Stochastic Automata

This section shows how the diagnostic problem can be solved for discrete-event systems described by a set $\{\mathcal{S}_f,\ f \in \mathcal{F}\}$ of stochastic automata. The behaviour of these automata is denoted by $\mathcal{B}_f,\ f \in \mathcal{F}$.

The idea of consistency-based diagnosis is to ask whether the measured I/O pair is consistent with the automaton $\mathcal{S}_f$. If the answer is in the affirmative

$$(V(0 \ldots k_\mathrm{e}), W(0 \ldots k_\mathrm{e})) \in \mathcal{B}_f, \tag{11.115}$$

the fault f may have occurred and is, thus, a fault candidate. In case of a negative answer

$$(V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e})) \notin \mathcal{B}_f, \tag{11.116}$$

the conclusion is that the system is not subjected to the fault f. For stochastic automata, for all fault candidates additional information can be obtained, which is expressed as the probability

$$\mathrm{Prob}(F = f \mid V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e}))$$

that the fault f has occurred if the system has generated the I/O pair $V(0\ldots k_\mathrm{e})$, $W(0\ldots k_\mathrm{e})$. Hence, diagnosing the stochastic automaton means to answer the question:

> With which probability has a fault f occurred if the system has generated the I/O pair $(V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e}))$?

The result is a set $\mathcal{F}^*(k_\mathrm{e})$ of *fault candidates*, which are those faults $f \in \mathcal{F}$ for which the I/O pair with time horizon k_e is consistent with the model $\mathcal{S}_f$:

$$\mathcal{F}^*(k_\mathrm{e}) = \{f \in \mathcal{F} : \mathrm{Prob}(F = f \mid V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e})) > 0\}.$$

11.7.2 Diagnosis of Stochastic Automata with Constant Faults

The diagnostic problem can be solved by observing the internal state for all models $\mathcal{S}_f$ and by selecting all those faults f for which the I/O pair is consistent with the model. Further, the probability of the fault for the measured I/O pair has to be determined. This idea will be explained in this section for constant faults $F = F(k)$ and extended to time-varying faults in the next section.

The idea is first to determine the common probability for the state z to occur at time k_e and the fault f to be present in the system

$$\begin{aligned}\mathrm{Prob}(Z(k_\mathrm{e}) = z, F = f \mid V(0) = \bar{v}_0, V(1) = \bar{v}_1, \ldots, V(k_\mathrm{e}) = \bar{v}_k,\\ W(0) = \bar{w}_0, W(1) = \bar{w}_1, \ldots, W(k_\mathrm{e}) = \bar{w}_k),\end{aligned}$$

where again the measured values of the input and the output are marked by a bar. This probability is abbreviated as $p_f(z, k_\mathrm{e})$. It is used to determine the marginal probability of the fault:

$$\begin{aligned}&\mathrm{Prob}(F = f \mid V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e}))\\ &= \sum_{z\in\mathcal{Z}} \mathrm{Prob}(Z(k_\mathrm{e}) = z, F = f \mid V(0\ldots k_\mathrm{e}), W(0\ldots k_\mathrm{e})),\end{aligned}$$

which will be denoted later by $p_f(k_e)$. To extend the state observation algorithm for stochastic automata developed in Sect. 8.6 three probability distributions are necessary:

$$\begin{aligned} p_f(z, k_e) &= \text{Prob}(Z(k_e) = z, F = f \mid V(0 \ldots k_e), W(0 \ldots k_e)) \\ p'_f(z', k_e) &= \text{Prob}(Z(k_e) = z', F = f \mid V(0 \ldots k_e - 1), W(0 \ldots k_e - 1)). \end{aligned}$$

The fault probability is abbreaviated as

$$p_f(k_e) = \text{Prob}(F = f \mid V(0 \ldots k_e), W(0 \ldots k_e))$$

and obtained as marginal probability of $p_f(z, k_e)$. If $p_f(k_e) = 0$, the I/O pair with time horizon k_e is inconsistent with the stochastic automaton $\mathcal{S}_f$. Otherwise, f is a fault candidate and $p_f(k_e)$ is the probability with which this fault has occurred.

The extension of Eq. (11.99) towards faulty systems yields the diagnostic method, which is summarised in the following theorem:

Theorem 11.6 (Recursive solution to the diagnostic problem) *Consider a set* $\{\mathcal{S}_f, f \in \mathcal{F}\}$ *of stochastic automata. The a-posteriori probabilities of the fault* f *together with a state* z *can be recursively determined as follows:*

$$p_f(z, k_e) = \frac{\sum\limits_{z'} L_f(z', \bar{w}_{k_e} \mid z, \bar{v}_{k_e}) \cdot p'_f(z, k_e - 1)}{\sum\limits_{z, z', f} L_f(z', \bar{w}_{k_e} \mid z, \bar{v}_{k_e}) \cdot p'_f(z, k_e - 1)} \tag{11.117}$$

and

$$p'_f(z', k_e) = \frac{\sum\limits_{z} L_f(z', \bar{w}_{k_e} \mid z, \bar{v}_{k_e}) \cdot p'_f(z, k_e - 1)}{\sum\limits_{z, z', f} L_f(z', \bar{w}_{k_e} \mid z, \bar{v}_{k_e}) \cdot p'_f(z, k_e - 1)} \quad (k_e > 0) \tag{11.118}$$

$$p'_f(z', -1) = p_0(z') \cdot \text{Prob}(F = f), \qquad (k_e = 0). \tag{11.119}$$

where $\text{Prob}(F = f)$ *denotes the a-priori fault probability. The diagnostic result is*

$$p_f(k_e) = \sum_{z \in \mathcal{Z}} p_f(f, k_e). \tag{11.120}$$

This result is used now to extend Algorithm 11.6 for the state observation towards the following algorithm for fault diagnosis:

Algorithm 11.7 *Diagnosis of stochastic automata*

Given: Set of stochastic automata $\{\mathcal{S}_f,\ f \in \mathcal{F}\}$.

Initial state probability distribution $p_0(z)$

Initial fault probability distribution $\text{Prob}(F = f)$.

Initialisation: $p'_f(z) = p_0(z) \cdot \text{Prob}(F = f)$ for all $z \in \mathcal{Z}$ and $f \in \mathcal{F}$

$k_e = 0$.

Loop:

1. Measure the current input $\bar{v}$ and output $\bar{w}$.
2. Determine $h_f(z) = \sum_{z'} L_f(z', \bar{w} \mid z, \bar{v}) \cdot p'_f(z')$ for all $z \in \mathcal{Z}$ and $f \in \mathcal{F}$
3. If $\sum_z h_f(z) = 0$ holds, the fault f is not a fault candidate: $p_f(k_e) = 0$.
   If $\sum_{z,f} h_f(z) = 0$, the I/O pair is inconsistent with all models of the set $\{\mathcal{S}_f,\ f \in \mathcal{F}\}$. Stop the algorithm (an unknown fault has occurred).
4. Determine $p_f(z) = \dfrac{h_f(z)}{\sum\limits_{z,f} h_f(z)}$ for all $z \in \mathcal{Z}$ and $f \in \mathcal{F}$.
5. Determine $p'_f(z') = \dfrac{\sum\limits_z L(z', \bar{w} \mid z, \bar{v})\ p'(z)}{\sum\limits_{z,f} h_f(z)}$ for all $z \in \mathcal{Z}$ and $f \in \mathcal{F}$.
6. Determine $p_f(k_e) = \sum_{z \in \mathcal{Z}} p_f(f, k_e)$ and $\mathcal{F}^*(k_e) = \{f \in \mathcal{F} : p_f(k_e) > 0\}$.
7. $k_e := k_e + 1$
   Continue with Step 1.

Result: Set of fault candidates $\mathcal{F}^*(k_e)$ together with fault probability $p_f(k_e)$ for increasing time horizon k_e.

The algorithm needs to get the a-priori fault probability distribution $\text{Prob}(F(0) = f)$, $(f \in \mathcal{F})$ as an input. If nothing is known about the faults, a uniform distribution

$$\text{Prob}(F(0) = f) = \frac{1}{q}, \quad f \in \mathcal{F}$$

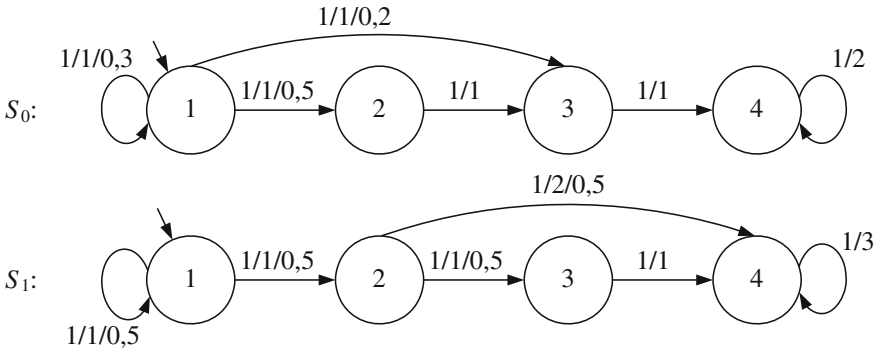


Fig. 11.33 Model of the faultless and the faulty system

may be reasonable, where q is the number of faults considered.

The diagnostic result can be summarised as follows:

- **Fault detection**: If $f_0 \notin \mathcal{F}^*(k_e)$ holds with f_0 denoting the faultless case, then a fault is detected in the system.

- **Fault identification**: The fault f that is present in the system belongs to the set of fault candidates

$$f \in \mathcal{F}^*(k_e)$$

unless it is not considered in the set of faults $\mathcal{F}$. If the set $\mathcal{F}^*(k_e)$ is a singleton, the fault is unambiguously identified. If this set includes more than one element, the probability $p_f(k_e)$, $f \in \mathcal{F}^*(k_e)$ describes with which frequency these faults appear.

Example 11.8 Diagnosis of a stochastic automaton
The automaton in Fig. 11.33 has the state $\tilde{Z} = (z, \quad f)^T$ for the two fault cases $f = 0$ und $f = 1$. Assume that the initial state $z_0 = 1$ is unambiguously known:

$$p_0(z) = \text{Prob}(Z(0) = z) = \begin{cases} 1 & \text{for } z = 1 \\ 0 & \text{else.} \end{cases}$$

For the fault, the a-priori probability distribution is assumed to be uniform:

$$\text{Prob}(F = f) = 0.5 \quad \text{for } f = 0, 1,$$

which means that the diagnostic algorithm starts with the initial probabilities

$$p'_f(z) = \begin{cases} 0.5 & \text{for } z = 1, f = 0, 1 \\ 0 & \text{else.} \end{cases}$$

Table 11.3 shows the diagnostic result for the I/O pair

Table 11.3 Probability distribution of the fault $p_f(k_e)$

	$k_e = 0$ $V(0...0) = (1)$ $W(0...0) = (1)$	$k_e = 1$ $V(0...1) = (1,1)$ $W(0...1) = (1,1)$	$k_e = 2$ $V(0...2) = (1,1,1)$ $W(0...2) = (1,1,1)$
f	$p_f(0)$	$p_f(1)$	$p_f(2)$
0	0.5	0.5714	0.5614
1	0.5	0.4286	0.4386

	$k_e = 3$ $V(0...3) = (1,1,1,1)$ $W(0...3) = (1,1,1,3)$	$k_e = 4$ $V(0...3) = (1,1,1,1,1)$ $W(0...3) = (1,1,1,3,3)$
f	$p_f(3)$	$p_f(4)$
0	0	0
1	1	1

$$V(0\ldots3) = (1,1,1,1,1)$$
$$W(0\ldots3) = (1,1,1,3,3).$$

For the initial state $z_0 = 1$, the first I/O pair ($V(0) = 1, W(0) = 1$) does not give any information about the fault, because both models generate the same output. At time $k_e = 1$, the system can be in one of the following states:

$$\mathcal{Z}_0(1\,|\,0) = \{1,2,3\}$$
$$\mathcal{Z}_1(1\,|\,0) = \{1,2\}$$

The I/O pair ($V(1) = 1, W(1) = 1$) excludes the state transition $2 \to 4$ and, hence, the fault $f = 1$ is less probable than the fault $f = 0$ after the second measurement. After the measurement $W(3) = 3$, the fault $f = 1$ is unambiguously identified. □

11.7.3 Extension to Time-Varying Faults

The diagnostic method developed in the last section for constant faults can be extended to time-varying faults using the model

$$\tilde{\mathcal{S}} = (\tilde{\mathcal{Z}}, \mathcal{V}, \mathcal{W}, \tilde{L}, \text{Prob}(\tilde{z}(0)))$$

given in Eq. (11.35), which includes the information about all fault cases and about the dynamics of the fault $F(k)$, $(k = 0, 1, \ldots, k_e)$. The main idea is to replace the behavioural relation L_f, $(f \in \mathcal{F})$ by the behavioural relation $\tilde{L}$ of the model $\tilde{\mathcal{S}}$.

Then, Eqs. (11.117)–(11.120) have to be replaced by the following relations:

$$p_f(z, k_e) = \frac{\sum_{z', f'} \tilde{L}(z', f', \bar{w}_{k_e} \mid z, \bar{v}_{k_e}, f) \cdot p'_f(z, k_e - 1)}{\sum_{z, z', f, f'} \tilde{L}(z', f', \bar{w}_{k_e} \mid z, \bar{v}_{k_e}, f) \cdot p'_f(z, k_e - 1)} \tag{11.121}$$

$$p'_f(z', k_e) = \frac{\sum_{z, f} \tilde{L}(z', f', \bar{w}_{k_e} \mid z, \bar{v}_{k_e}, f) \cdot p'_f(z, k_e - 1)}{\sum_{z, z', f, f'} \tilde{L}(z', f', \bar{w}_{k_e} \mid z, \bar{v}_{k_e}, f) \cdot p'_f(z, k_e - 1)}, \quad k_e > 0 \tag{11.122}$$

$$p'_f(z', -1) = p_0(z') \cdot \text{Prob}(F = f), \qquad k_e = 0 \tag{11.123}$$

$$p_f(k_e) = \sum_{z \in \mathcal{Z}} p_f(f, k_e). \tag{11.124}$$

Discussion of the diagnostic results. The diagnostic algorithm yields the set $\mathcal{F}^*(k_e)$ of fault candidates for time k_e. Each of the fault separately "explains" why the measured I/O pair occurs. In addition to this set, the probability distribution $p_f(k_e)$ evaluates with which probability each fault candidate represents the real fault.

Under practical circumstances, several heuristic extensions can be made. For example, a threshold s can be fixed and only those faults that occur with a probability higher than s are announced to the human operator in the control room. Then the threshold can be used to adapt the result of the diagnostic algorithm to the certainty with which the behavioural relation of the automaton is known and to the degree of danger that the different faults may have on the system performance. This adaptation of the diagnostic results is analogous to the use of thresholds in the residual evaluation of diagnostic methods for continuous-variable systems described in Chap. 6.

11.7.4 Diagnosability of Stochastic Automata

When solving practical problems, the diagnostic algorithm should provide a set $\mathcal{F}^*(k_e)$ which is a singleton for a possibly small time horizon k_e. Whether or not this is possible, depends on the diagnosability of the stochastic automaton, which is investigated in this section. Note that the diagnosablity is a system property, which depends upon the system dynamics described by the behavioural relation of the automaton and by the measured signals v and w, but does not refer to the diagnostic method applied. Diagnosability claims that the fault f has to be found by *appropriately* using all the information available.

It is not easy to find conditions under which the system is diagnosable, because the question whether a fault can be detected does not only depend on the system dynamics but also on the initial state and on the input sequence. However, the frequent discussions among theoreticans and people from different application fields on "hidden faults" that do not influence the measurement sequence and, thus,

cannot be found by any diagnostic algorithm, and discussions on the fact that different faults have to bring about different effects on the system behaviour if they should be discriminated, show that diagnosability is an important practical issue.

In this section, the results on the observability of stochastic automata presented in Sect. 11.6.3 will be used to define and analyse the diagnosability of automata. Like in Sect. 11.6.3, the starting point is the investigation under what conditions the automaton is not diagnosable.

Definition 11.6 (*Stochastic undiagnosability*) A stochastic automaton $\tilde{\mathcal{S}}$ with behavioural relation

$$\tilde{L}(z', f', w \mid z, v, f) = L(z', w \mid z, v, f) \cdot G_{\mathrm{f}}(f' \mid f)$$

is called stochastically undiagnosable if it satisfies the property

$$L(z', f', w \mid z, v, f) = L(z', w \mid z, v) \tag{11.125}$$

for all $z', z \in \mathcal{Z}$, $w \in \mathcal{W}$, $v \in \mathcal{V}$ and $f \in \mathcal{F}$.

Clearly, under the condition (11.125) the state and output sequences of the stochastic automaton are independent of the fault f, because the fault does no longer appear in L and, hence, the fault cannot be "seen" from the measured I/O pair. In analogy to Lemma 11.7, it can be proved that for undiagnosable automata the diagnostic result coincides with the result obtained by simulation of the faulty behaviour:

Lemma 11.8 *If the stochastic automaton is stochastically undiagnosable, then for all input sequences V and for all output sequences W the diagnostic result is identical to the simulation result:*

$$\mathrm{Prob}(f(k_{\mathrm{e}}) \mid V(0 \ldots k_{\mathrm{e}}), W(0 \ldots k_{\mathrm{e}})) = \mathrm{Prob}(f(k_{\mathrm{e}}) \mid V(0 \ldots k_{\mathrm{e}})). \tag{11.126}$$

The left-hand side of Eq. (11.126) is the result obtained from the diagnostic algorithm. As the fault f does not depend on the input v, the right-hand side of Eq. (11.126) is given by the relation

$$\begin{aligned}
&\mathrm{Prob}(F(k_{\mathrm{e}}) = f \mid V(0 \ldots k_{\mathrm{e}})) \\
&= \sum_{F(0 \ldots k_{\mathrm{e}}-1)} G_{\mathrm{f}}(f \mid f(k_{\mathrm{e}}-1)) \cdot G_{\mathrm{f}}(f(k_{\mathrm{e}}-1) \mid f(k_{\mathrm{e}}-2)) \cdot \ldots \\
&\qquad \cdot G_{\mathrm{f}}(f(1) \mid f(0)) \cdot \mathrm{Prob}(F(0) = f(0)),
\end{aligned}$$

which predicts the state of the fault model $\mathcal{S}_f$. The fault changes with increasing time horizon k_{e} and so does the probability distribution

$$\mathrm{Prob}(F(k_{\mathrm{e}}) = f \mid V(0 \ldots k_{\mathrm{e}})).$$

However, the only information used for simulation is the state transition relation G_f of the fault model. As the diagnostic algorithm uses further information given by the output sequence W it is reasonable to expect that the diagnostic result is better than the simulation result. The lemma says that for stochastically undiagnosable automata this expectation is not met. The diagnostic algorithm cannot improve the simulation result.

A given stochastic automaton is generally not completely stochastically undiagnosable according to Eq. (11.125), but there may exist one or more state sets $\mathcal{G}_z \subset \mathcal{Z}$ and one or more fault sets $\mathcal{G}_f \subset \mathcal{F}$ such that the behaviour within the set $\mathcal{G}_z$ does not depend on the faults $f \in \mathcal{G}_f$. Then Eq. (11.125) does not hold for all z, z' and f, but for all $z, z' \in \mathcal{G}_z$ and all $f \in \mathcal{G}_f$. If a non-empty fault set $\mathcal{F}$ can be found such that Eq. (11.125) is satisfied for all $z, z' \in \mathcal{G}_z$, the set $\mathcal{G}_z$ is called a *stochastically undiagnosable state set*. If the stochastic automaton does not possess such a state set, it is called diagnosable.

Definition 11.7 (*Stochastic diagnosability*) A stochastic automaton is called stochastically diagnosable if it does not possess any stochastically undiagnosable state set.

It is obvious from the investigations above that for stochastically diagnosable systems the diagnostic algorithm yield better results than a simulation of the behaviour of the fault model.

Example 11.9 Diagnosability of stochastic automata
The stochastic automaton depicted in Fig. 11.33 is not stochastically undiagnosable. Nevertheless, the set of faults $\mathcal{F}$ is stochastically undiagnosable within the set of states $\mathcal{G}_z = \{1, 2, 3\}$. Hence, as long as the system is not in state $z = 4$ nor has the possibility to go to this state within one time step, no information about the fault can be obtained. This result can be seen from Example 11.8. The fault $f = 1$ is proved not to exist at time $k_\mathrm{e} = 3$ when the output $w = 3$ occurs, which proves that the automaton is in the state $\tilde{z} = (4, 2)^\mathrm{T}$. □

Example 11.10 Diagnosis of a stochastic automata
As an example, consider the task system to diagnose a fault by means of the automata shown in Figs. 11.34 and 11.35. Both automata together describe the behavioural relation $L_f(z', w \mid z, v)$. The fault is assumed to be constant.

A diagnosability check for $v = 1$ yields the result that the stochastic automaton is not diagnoseable with respect to the set of faults $\mathcal{F} = \{1, 2\}$ within all states $\mathcal{G}_z = \mathcal{Z}$. For $v = 2$ the automaton is diagnosable.

Three experiments are considered. First, the input is fixed at $v = 2$ and the fault is $f = 1$. An experiment with the initial state $z = 6$ yields the output sequence shown in the left part of Fig. 11.36. A second experiment with the same initial state is made for $v = 2$ and $f = 2$ resulting in the output sequence shown in the middle part of Fig. 11.36, and a third experiment with $v = 1$ and $f = 1$ leads to the right part of Fig. 11.36.

The diagnostic results corresponding to the three experiments are shown in Fig. 11.37. It can be seen that the fault is isolated for $v = 2$, but for $v = 1$ the diagnostic result is merely a simulation of the initially known fault distribution, which results for the given automaton in a sequence of uniform distributions. The result is obtained because the automaton is undiagnosable for $v = 1$ in the whole state set $\mathcal{Z}$. □

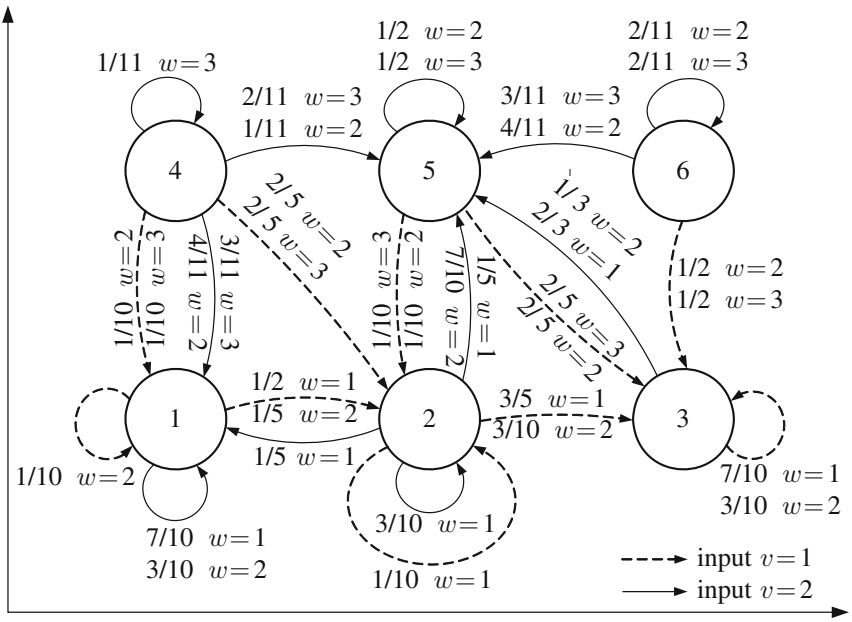


Fig. 11.34 Automaton graph for fault $f = 1$

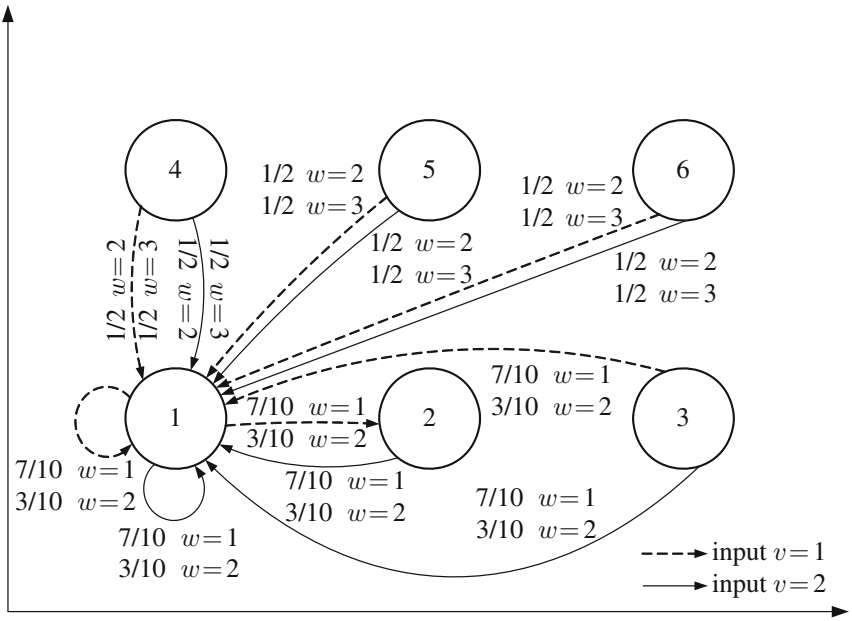


Fig. 11.35 Automaton graph for fault $f = 2$

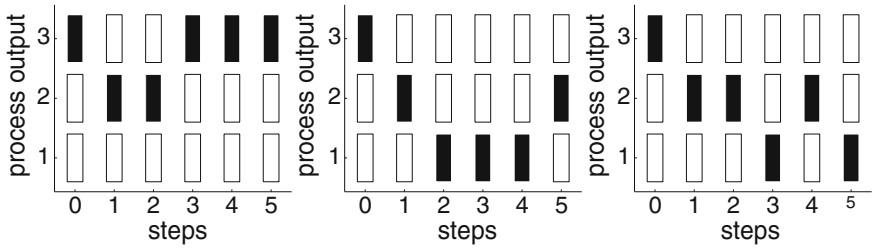


Fig. 11.36 Output sequences for $v = 2$, $f = 1$ (*left*), $v = 2$, $f = 2$ (*middle*) and $v = 1$, $f = 1$ (*right*)

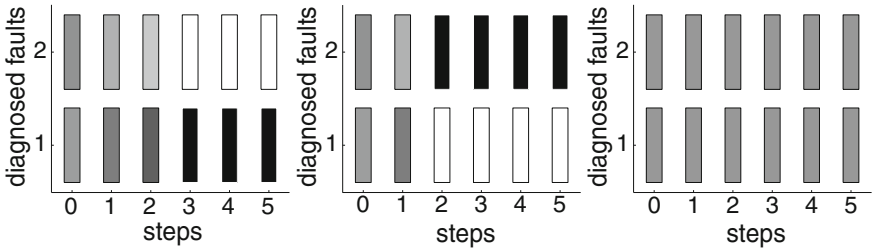


Fig. 11.37 Diagnostic results for the three experiments shown in Fig. 11.36 in the same order

11.8 Exercises

Exercise 11.1 Observability of stochastic automata
Assume that the current state probability distribution is given for time k_e and denoted by $\text{Prob}(z(k_e))$. Prove the following fact: If at time $k_e + 1$ an input $v(k_e + 1)$ and an output $w(k_e + 1)$ occur for which a decomposition (11.107) is possible for the state set

$$\mathcal{Z}(k_e \mid k_e) = \{z : \text{Prob}(Z(k_e) = z) > 0\}$$

then in Steps 4 and 5 of the observation algorithm the same results are obtained as by simulating the automaton behaviour according to Eq. (11.26). □

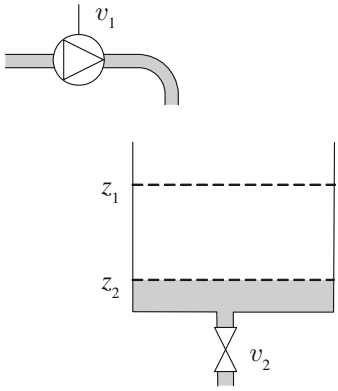
Exercise 11.2 Diagnosis of fixed faults
How can the Algorithm 11.7 be simplified if the fault is known not to change over time? □

Exercise 11.3 Diagnosis of a batch reactor
The reactor shown in Fig. 11.38 is used within a larger batch process, where it is filled and emptied in order to bring a certain amount of liquid into another reactor. For the behaviour of the reactor only the empty and the full state is distinguished, where the liquid level is above the higher or below the lower border shown in the figure. These states are denoted by z_1 and z_2.

To fill the reactor, the pump is switched on (input v_1), to empty the reactor, the input v_2 opens the valve. A security check ensures that the pump is not switched on if the valve is open.

1. Describe the reactor by a deterministic automaton

Fig. 11.38 Batch reactor



2. The fault f_1 breaks the pump. Extend you model in order to describe the reactor for the faultless and the faulty operation mode.
3. The faultless reactor remains faultless with the probability of 99 % in all state transitions that are caused by switching the pump or opening and closing the valve. Extend you model to get a stochastic automaton, which reflects this information. □

Exercise 11.4 Diagnosis of nondeterministic automata
How can the diagnostic method developed in this chapter be simplified if instead of a stochastic automaton a nondeterministic automaton is used to describe the system under consideration? Do the sets of fault candidates obtained by both methods distinguish? □

11.9 Bibliographical Notes

The first results concerning state observation of discrete-event systems occurred in connection with the supervisory control theory developed in [284] where the supervisor has to reconstruct the current state of the system from partially measurable states or events. Reference [198] defined the notion of the observable language and developed results on the existence of the combined supervisory control and the observation problem given. Reference [55] showed that for supervisory control the problem of state observation can be reformulated as an event observation problem.

Observability of discrete-event-systems. The classical observability definition has been given in [56] and was used also, for example, in the textbooks [51, 320]. According to this definition a stochastic automaton is called *semi-deterministic* or observable if for all states z the successor state $z' = \varphi(z, v, w)$ can be unambiguously determined if the current state z, the current input v and current output w are known. This definition is useful only if it can be *assumed* that the automaton state z at some time k is precisely known. Then, the future sequence of states starting in z can be unambiguously determined. However, as long as this assumption is not satisfied, the notion

of observability does not say anything about the solvability of the observation problem. Similar remarks hold true for other observability definitions like the one given in [259] which likewise claim that the automaton state should be unambiguously determined.

Several papers have been published about the connection of state observation and fault diagnosis for discrete-event systems. Some of the aspects discussed are summarised in [95].

Diagnosability and diagnostic methods. The diagnosis of discrete-event systems was the subject of a steadily increasing number of papers in the past with the references [12, 194, 277, 319] as early papers on the diagnostic problem for Petri nets, and [179, 197, 216, 218, 297, 306] for nondeterministic or stochastic automata. Conditions on the automaton under which the fault can be uniquely determined have been derived in [197, 297]. If the input to the automaton should satisfy certain requirements to avoid the situation where the system reaches forbidden states, the diagnosability conditions appear to be stronger as shown in [268, 269]. A combination of Petri net and automata theoretic approaches is described in [52].

In the stochastic setting, there are different definitions of diagnosability. In [366], a stochastic automaton is said to be diagnosable if on all state trajectories the fault can be detected and, hence, the fault is eventually unambiguously identified. In [218], diagnosability means that the fault changes the I/O behaviour and leads to an increase in its probability. Reference [179] describes detectability of a fault as the ability to estimate the current state of a system with increasing certainty. Reference [277] outlined the connections between discrete-event models and logical descriptions, which opens the way to apply diagnostic methods elaborated in the field of artificial intelligence to discrete-event systems (for survey cf. [139] or [212]). However, most of the diagnostic methods developed in artificial intelligence can only be used for static system descriptions, whereas the methods that have been developed here allow to diagnose dynamical systems far from their equilibrium state.

Based on Algorithm 11.7 for the diagnosis of stochastic automata, a specific sensor and actuator supervision system has been developed in [219].

The results on the diagnosability of deterministic automata developed in Sect. 11.4 have been published in [215]. The method for finding distinguishing inputs developed in Sect. 11.4.5 is similar to the one described in [128] but uses an alternative notation and, thus, directly extends a method for determining equivalent states of deterministic automata. Details can be found in [303, 304].

Extensions. The issue of complexity reduction of the diagnosis of automata has been considered in [210]. The basic idea was to lump unobservable states of the model together because during the movement in such sets of states the observation or diagnostic algorithm does not gain any additional information about the system. It has been shown that this method of complexity reduction can be applied for nondeterministic automata. However, for stochastic automata the complexity reduction brings about biased diagnostic results.

The results reported in Sect. 11.5 have been developed in [218]. All proofs of the results given here can be found in this reference.

The extension of the diagnostic method to remote diagnosis, where data loss in the network has to be tolerated, is described in [114, 302].

The methods explained in this chapter can be extended to timed automata as shown in [351].

The introduction to automata theory and the Exercise 11.3 follow the textbook [209].

Chapter 12
Diagnosis of I/O Automata Networks

Abstract This chapter is devoted to complex discrete-event systems that are represented by input–output automata networks. A method for decentralised diagnosis is developed where the diagnostic units have only access to the model and the measurement sequences of a subsystem. It guarantees that the diagnostic result is complete in spite of the lack of information about the subsystem interactions. Completeness means that the set of faults found by the diagnostic units include all fault candidates. Sufficient conditions for the autonomy of subsystems and for the kind of asynchronous state transitions are derived for which the result of decentralised diagnosis coincides with the result of centralised diagnosis.

12.1 Centralised Versus Decentralised Diagnosis of Discrete-Event Systems

Fault diagnosis is the task to decide, on the basis of measurement sequences and a model, whether a dynamical system is subject to some fault and, if so, to identify the fault. This task leads to highly complex diagnostic algorithms if the system under consideration has to be described by a discrete-event model, because such models generally have an exponential complexity with respect to the number of states, input symbols and output symbols.

To overcome this complexity, several structured diagnostic schemes have been elaborated. In *distributed diagnosis* the system is represented by a set of interacting components. Accordingly, the diagnostic unit consists of a set of local units which have access to different event streams. As the system components interact due to their physical couplings, the diagnostic units are allowed to exchange messages about their observations or their diagnostic results with the aim to merge the local result and to get the best possible result for the overall system. This scheme necessitates an extensive information exchange which should compensate the lack of information that is locally measurable.

Decentralised diagnosis. In contrast to this, in *decentralised diagnosis* separate diagnostic units are used that cannot communicate and have only access to the model

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8_12

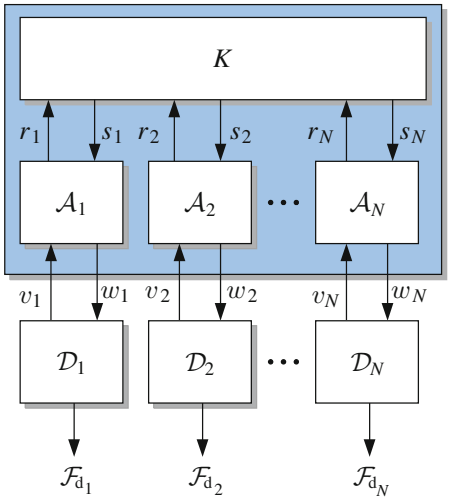


Fig. 12.1 Decentralised diagnosis of interconnected discrete-event systems

of a subsystem and the measurement sequences made for this subsystem (Fig. 12.1). Hence, the overall complexity of the diagnostic method is much lower than that of centralised or distributed diagnosis. However, due to the lack of information about the influence of the component interactions, the diagnostic results are, in general, weaker than the results of centralised or of fully coordinated distributed diagnosis.

In decentralised diagnosis, N local diagnostic units $\mathcal{D}_i$ are used, each of which has only access to the I/O pair of the ith subsystem:

$$\begin{aligned} V_i(0 \ldots k_\mathrm{e}) &= (v_i(0), v_i(1), \ldots, v_i(k_\mathrm{e})) \\ W_i(0 \ldots k_\mathrm{e}) &= (w_i(0), w_i(1), \ldots, w_i(k_\mathrm{e})). \end{aligned}$$

Furthermore, the ith diagnostic unit knows only the model $\mathcal{A}_i$ of the associated subsystem. It solves the following problem:

Diagnostic problem for the i-th subsystem

Given:	Automaton $\mathcal{A}_i$ describing the subsystem
	I/O pair $(V_i(0 \ldots k_\mathrm{e}), W_i(0 \ldots k_\mathrm{e}))$
Find:	Local diagnostic result $\mathcal{F}_{\mathrm{d}i}$

This problem is solved by the ith local diagnostic unit $\mathcal{D}_i$. The results $\mathcal{F}_{\mathrm{d}i} \subseteq \mathcal{F}_i$ $(i = 1, 2, \ldots, N)$ of all units are lumped together to obtain the overall diagnostic result denoted by $\mathcal{F}_\mathrm{d} \subseteq \mathcal{F}$. An important issue is to ensure the diagnostic result to be complete, which means that the relation

$$\mathcal{F}_\mathrm{c} \subseteq \mathcal{F}_\mathrm{d} \tag{12.1}$$

between the centralised and the decentralised diagnostic results has to be valid.

The main aim of this chapter is to elaborate a decentralised diagnostic method for networks of deterministic I/O automata. The model used distinguishes considerably from those used in the literature (cf. bibliographical notes at the end of the chapter). First, the fault is represented here as an additional argument of the state transition function and not by an unobservable event. Second, networks of I/O automata are used in the generalised version, where the automata are coupled through interconnection signals.

The aim is to develop a decentralised diagnostic method such that the diagnostic result is complete. The property of completeness that will be defined more clearly in Definition 12.2 means, in principle, that all faults that can be found by the centralised diagnostic unit by evaluating the I/O sequences in the best possible way are also found by the decentralised diagnostic scheme.

To reach this aim, the following two problems have to be solved:

1. Develop a diagnostic method for the decentralised diagnostic units that uses only the I/O pair and the model of a subsystem.
2. Develop a method to merge the diagnostic results obtained by the local diagnostic units to obtain a diagnostic result for the overall system.

Centralised diagnosis. In order to characterise the validity of the diagnostic result obtained by the decentralised scheme, the centralised diagnosis of the automaton network will be considered as the alternative way of solution. The overall system has the vector inputs and outputs

$$\boldsymbol{v}(k) = \begin{pmatrix} v_1(k) \\ v_2(k) \\ \vdots \\ v_N(k) \end{pmatrix} \quad \text{and} \quad \boldsymbol{w}(k) = \begin{pmatrix} w_1(k) \\ w_2(k) \\ \vdots \\ w_N(k) \end{pmatrix}$$

(Fig. 12.2). The component faults are summarised in the vector

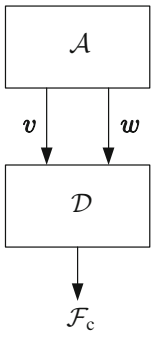


Fig. 12.2 Centralised diagnosis

$$\boldsymbol{f} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{pmatrix}$$

that belongs to the set

$$\mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2 \times \cdots \times \mathcal{F}_N .$$

In centralised diagnosis a single diagnostic unit $\mathcal{D}$ obtains the I/O pair of the overall system

$$\begin{aligned} \boldsymbol{V}(0 \ldots k_\mathrm{e}) &= (\boldsymbol{v}(0), \boldsymbol{v}(1), \ldots, \boldsymbol{v}(k_\mathrm{e})) \\ \boldsymbol{W}(0 \ldots k_\mathrm{e}) &= (\boldsymbol{w}(0), \boldsymbol{w}(1), \ldots, \boldsymbol{w}(k_\mathrm{e})) \end{aligned}$$

for the time horizon $0 \ldots k_\mathrm{e}$ and the task is to find the fault $\boldsymbol{f} \in \mathcal{F}$.

Diagnostic problem for the overall system

Given:	Automaton $\mathcal{A}$ describing the overall system
	I/O pair $(\boldsymbol{V}(0 \ldots k_\mathrm{e}), \boldsymbol{W}(0 \ldots k_\mathrm{e}))$
Find:	Diagnostic result $\mathcal{F}_\mathrm{c}$

The result is a set $\mathcal{F}_\mathrm{c} \subseteq \mathcal{F}$ of fault vectors $\boldsymbol{f}$.

12.2 Representation of Complex Systems by I/O Automata Networks

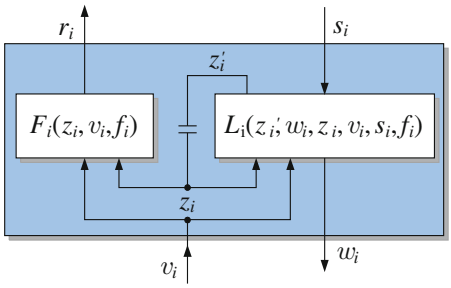
12.2.1 Composite Systems to Be Diagnosed

Consider discrete-event systems composed of N subsystems that interact through the signals r_i and s_i (Fig. 12.1). Every subsystem may be faulty, what is described by the fault symbols $f_i \in \mathcal{F}_i$ associated with the ith subsystem. It is assumed that the fault does not change while the diagnostic method is applied. Consequently, faults can be dealt with as constant subsystem parameters. The faultless case is represented by the fault symbol $f_i = 0$.

The subsystems are described by deterministic I/O automata (Fig. 12.3)

$$\mathcal{A}_i = (\mathcal{Z}_i, \mathcal{V}_i, \mathcal{W}_i, \mathcal{S}_i, \mathcal{R}_i, \mathcal{F}_i, F_i, L_i, \mathcal{Z}_{i0})$$

Fig. 12.3 Subsystem model



that have the following components:

- $\mathcal{Z}_i$ - state set
- $\mathcal{V}_i$ - input alphabet
- $\mathcal{W}_i$ - output alphabet
- $\mathcal{S}_i$ - coupling input alphabet
- $\mathcal{R}_i$ - coupling output alphabet
- $\mathcal{F}_i$ - set of faults
- $F_i : \mathcal{Z}_i \times \mathcal{V}_i \times \mathcal{F}_i \to \mathcal{R}_i$ - coupling function
- $L_i : \mathcal{Z}_i \times \mathcal{W}_i \times \mathcal{Z}_i \times \mathcal{V}_i \times \mathcal{S}_i \times \mathcal{F}_i \to \{0, 1\}$ - state transition function
- $\mathcal{Z}_{i0}$ - set of possible initial states ($\mathcal{Z}_{i0} \neq \emptyset$).

The function F_i determines the interconnection output $r_i(k)$ in dependence upon the input symbol $v_i(k)$, the state $z_i(k)$ of the ith subsystem, and the fault f_i:

$$r_i(k) = F_i(z_i(k), v_i(k), f_i). \tag{12.2}$$

The state transition function represents all 6-tuples $(z_i', w_i, z_i, v_i, s_i, f_i)$ that can occur for the subsystem. The relation

$$L_i(z_i', w_i, z_i, v_i, s_i, f_i) = 1$$

means that the ith subsystem subject to fault f_i changes its state from z_i towards z_i' while generating the output w_i if it gets the input symbol v_i and the interconnection input s_i.

The interconnection outputs r_i and inputs s_i of all components form the vector signals

$$\boldsymbol{r}(k) = \begin{pmatrix} r_1(k) \\ r_2(k) \\ \vdots \\ r_N(k) \end{pmatrix} \quad \text{and} \quad \boldsymbol{s}(k) = \begin{pmatrix} s_1(k) \\ s_2(k) \\ \vdots \\ s_N(k) \end{pmatrix}.$$

The interactions of the components are described by the function K:

$$\boldsymbol{s}(k) = K(\boldsymbol{r}(k)).$$

The overall system is denoted by $\mathcal{A} = K(\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_N)$, which symbolises the fact that the component automata are coupled via the interaction relation K. It is represented by the deterministic automaton

$$\mathcal{A} = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, \mathcal{F}, L, \mathcal{Z}_0)$$

that has the following components:

- $\mathcal{Z} = \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_N$ - state set
- $\mathcal{V} = \mathcal{V}_1 \times \cdots \times \mathcal{V}_N$ - input alphabet
- $\mathcal{W} = \mathcal{W}_1 \times \cdots \times \mathcal{W}_N$ - output alphabet
- $\mathcal{F} = \mathcal{F}_1 \times \cdots \times \mathcal{F}_N$ - set of faults
- $L: \ \mathcal{Z} \times \mathcal{W} \times \mathcal{Z} \times \mathcal{V} \times \mathcal{F} \to \{0, 1\}$ - state transition function
- $\mathcal{Z}_0 = \mathcal{Z}_{10} \times \cdots \times \mathcal{Z}_{N0}$ - set of possible initial states ($\mathcal{Z}_0 \neq \emptyset$).

12.2.2 Model of the Overall System

This section shows how the component models $\mathcal{A}_i$ and the interconnection relation K can be combined to get a deterministic automaton

$$\mathcal{A} = (\mathcal{Z}, \mathcal{V}, \mathcal{W}, L, \mathcal{Z}_0)$$

that describes the overall system in an equivalent way. Like in Sect. 11.6.1 the system is considered without reference to a fault $\boldsymbol{f}$.

The interconnection outputs of all subsystems yield the vector

$$\boldsymbol{r}(k) = \begin{pmatrix} r_1(k) \\ r_2(k) \\ \vdots \\ r_N(k) \end{pmatrix} = \begin{pmatrix} F_1(z_1(k), v_1(k)) \\ F_2(z_2(k), v_2(k)) \\ \vdots \\ F_N(z_N(k), v_N(k)) \end{pmatrix} \tag{12.3}$$

(cf. Eq. (12.2) with deleted symbol f_i). The interconnection relation

$$\boldsymbol{s}(k) = K(\boldsymbol{r}(k))$$

is decomposed into N relations

$$s_i(k) = K_i(\boldsymbol{r}(k)) \quad k = 1, 2, \ldots, N,$$

which determine the interconnection input s_i of the subsystems.

The dynamics of the ith subsystem are described by the relation

$$L_i(z_i(k+1), w_i(k), z_i(k), v_i(k), K_i(\boldsymbol{r}(k))) = 1.$$

Note that the function L_i depends through the vector $\boldsymbol{r}(k)$ upon all states z_i and inputs v_i, $(i = 1, 2, \ldots, N)$. A state transition

$$\begin{pmatrix} z_1(k) \\ z_2(k) \\ \vdots \\ z_N(k) \end{pmatrix} \xrightarrow{\boldsymbol{v}(k)/\boldsymbol{w}(k)} \begin{pmatrix} z_1(k+1) \\ z_2(k+1) \\ \vdots \\ z_N(k+1) \end{pmatrix}$$

occurs in the overall system if all state transition functions L_i, $(i = 1, 2, \ldots, N)$ of the subsystems are satisfied, which is true if the condition

$$\prod_{i=1}^{N} L_i(z_i(k+1), w_i(k), z_i(k), v_i(k), K_i(\boldsymbol{r}(k))) = 1$$

is met. In the usual notation where the successor state is indicated by the prime, the composition rule is written in the following concise form:

$$L\left(\begin{pmatrix} z_1' \\ z_2' \\ \vdots \\ z_N' \end{pmatrix}, \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix}, \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{pmatrix}, \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix}\right) = \prod_{i=1}^{N} L_i(z_i', w_i, z_i, v_i, K_i(\boldsymbol{r})). \tag{12.4}$$

By using Eq. (12.3), the interconnection signal $\boldsymbol{r}$ can be eliminated from this equation.

System with two components. If the system has only two components, the interaction relation may have the simple form

$$\begin{pmatrix} s_1(k) \\ s_2(k) \end{pmatrix} = K(\boldsymbol{r}(k)) = \begin{pmatrix} r_2(k) \\ r_1(k) \end{pmatrix}$$

(Fig. 12.4). The overall system has the state transition function L, which is obtained from Eq. (12.4) as follows:

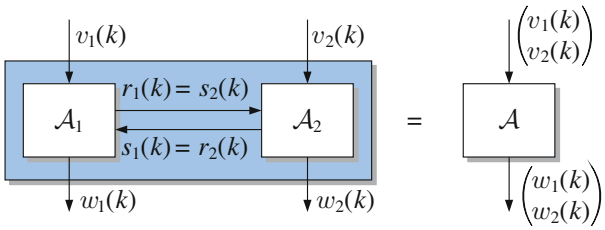


Fig. 12.4 Network with two I/O automata

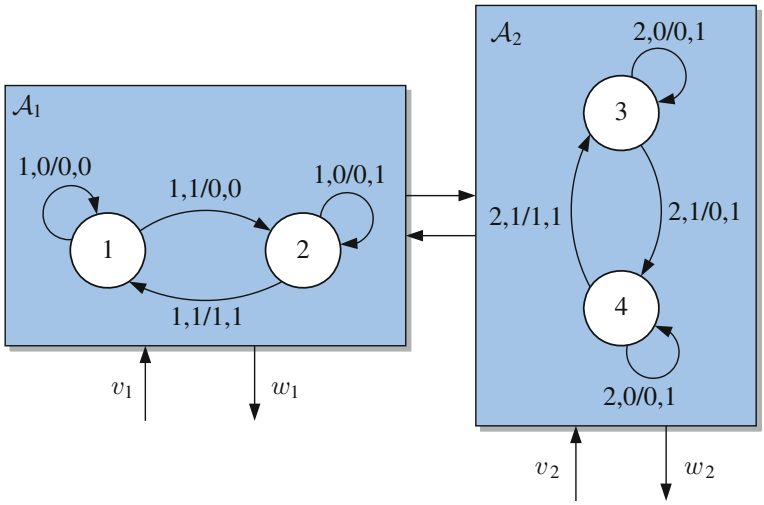


Fig. 12.5 Automata network

$$L\left(\begin{pmatrix} z_1' \\ z_2' \end{pmatrix}, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}, \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}, \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}\right) =$$

$$L_1(z_1', w_1, z_1, v_1, F_2(z_2, v_2)) \cdot L_2(z_2', w_2, z_2, v_2, F_1(z_1, v_1)).$$

Example 12.1 Automata network with two components
In the automata network shown in Fig. 12.5 the components are defined for the sets

$$\begin{aligned} \mathcal{Z}_1 &= \{1, 2\}, & \mathcal{Z}_2 &= \{2, 4\} \\ \mathcal{Z}_{10} &= \{1, 2\}, & \mathcal{Z}_{20} &= \{4\} \\ \mathcal{V}_1 &= \{1\}, & \mathcal{V}_2 &= \{2\} \\ \mathcal{W}_1 &= \{0, 1\}, & \mathcal{W}_2 &= \{0, 1\} \\ \mathcal{S}_1 = \mathcal{S}_2 &= \mathcal{R}_1 = \mathcal{R}_2 = \{0, 1\}. \end{aligned} \tag{12.5}$$

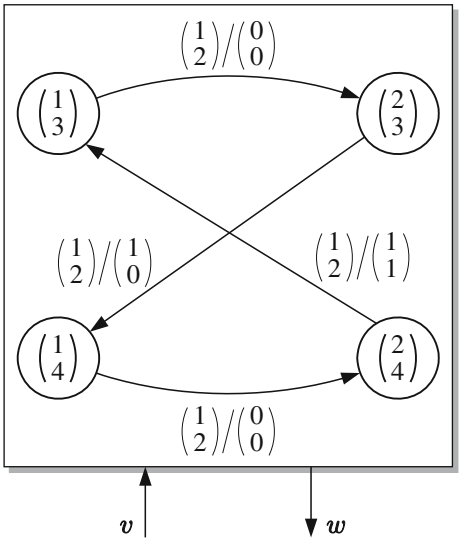


Fig. 12.6 Equivalent deterministic automaton

The state transition functions L_1 and L_2 are given by automaton graphs in Fig. 12.5 whose edges have the labels $v_i, s_i/w_i, r_i$.

The subsystem models are set up in such a way that the interconnection input $s_i = 1$ allows a state transition whereas the input $s_i = 0$ blocks a state transition in subsystem i

$$L_i(z_i, 0, z_i, v_i, 0) = 1$$

(cf. the loops at all states in the automaton graphs).

The interconnection of the two subsystems leads to the automaton shown in Fig. 12.6. The application of the composition rule is demonstrated for determining the state transition function L for the states $z_1 = 1, z_2 = 3$ and the inputs $v_1 = 1, v_2 = 2$:

$$\begin{aligned} & L\left(\begin{pmatrix} z_1' \\ z_2' \end{pmatrix}, \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}, \begin{pmatrix} 1 \\ 3 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \end{pmatrix}\right) \\ = \; & L_1(z_1', w_1, 1, 1, F_2(3, 2)) \cdot L_2(z_2', w_2, 3, 2, F_1(1, 1)) \\ = \; & L_1(z_1', w_1, 1, 1, 1) \cdot L_2(z_2', w_2, 3, 2, 0). \end{aligned}$$

The first factor is non-zero for $z_1' = 2$ and $w_1 = 0$ as shown by the edge $1 \overset{1,1,/0,0}{\longrightarrow} 2$ of the automaton graph of $\mathcal{A}_1$ in Fig. 12.5. The second factor is non-zero for $z_2' = 3$, $w_2 = 0$. Hence,

$$L\left(\begin{pmatrix} 2 \\ 3 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 3 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \end{pmatrix}\right) = 1$$

holds, which corresponds to the edge

$$\begin{pmatrix} 1 \\ 3 \end{pmatrix} \xrightarrow{\begin{pmatrix} 1 \\ 2 \end{pmatrix} / \begin{pmatrix} 0 \\ 0 \end{pmatrix}} \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

in the automaton graph in Fig. 12.6. The other state transitions of the overall system are determined in the same way. □

12.3 Decentralised Consistency Test

This section develops a method for the decentralised consistency test. The problem is to determine whether or not the I/O pair (11.71), (11.72) is consistent with the model $\mathcal{A}$ of the overall system. In the decentralised scheme, N units are used to carry out the test, each of which has only access to the I/O pair and the model of a subsystem. The test is first explained for a centralised scheme.

12.3.1 Consistency Test for the Overall System

If the full information (overall model and overall I/O pair) is available, the consistency can be checked as described by Lemma 11.4. Accordingly, the test result is positive if and only if Eq. (11.77) is satisfied

$$p_{\mathrm{c}} = \sum_{z \in \mathcal{Z}} p'(z', k_{\mathrm{e}} + 1) > 0, \tag{12.6}$$

where $p'(z', k_{\mathrm{e}}+1)$ is determined recursively by Eqs. (11.75), (11.76). The test result is called p_{c} with the index "c" indicating that a centralised test scheme is used. The test can be accomplished by means of Algorithm 11.4.

As a by-product, the set of all states, in which the overall system can be after accepting the given input sequence and producing the given output sequence is obtained as follows:

$$\mathcal{Z}_{\mathrm{c}}(k_{\mathrm{e}} + 1) = \{z' \in \mathcal{Z} \mid p'(z', k_{\mathrm{e}} + 1) > 0\}.$$

Again, the index "c" indicates that this set is the result of a centralised consistency test.

12.3.2 Consistency Test for the Subsystems

In a decentralised test scheme, the ith test unit has access only to the model $\mathcal{A}_i$ of the ith subsystem and to the I/O pair

$$V_i(0 \ldots k_e) = (\bar{v}_{0i}, \bar{v}_{1i}, \ldots, \bar{v}_{k_e i}) \tag{12.7}$$

$$W_i(0 \ldots k_e) = (\bar{w}_{0i}, \bar{w}_{1i}, \ldots, \bar{w}_{k_e i}). \tag{12.8}$$

Like in the I/O pair of the overall system, the measured values are marked by a bar.

Completeness of the consistency test. For the subsystem test, Algorithm 11.4 has to be extended because the subsystem performance depends upon the interconnection input s_i which is unknown for the decentralised test unit (cf. Fig. 12.1). This extension has to be done in such a way that the overall test result is complete in the sense defined as follows.

Denote the result of the consistency test accomplished by the ith unit by p_{di}, where $p_{di} > 0$ indicates consistency of the I/O pair (V_i, W_i) of the ith subsystem with the subsystem model $\mathcal{A}_i$ and $p_{di} = 0$ inconsistency. All such results together will be merged to get the test result p_d for the overall system, where again $p_d > 0$ indicates that all I/O pairs of all subsystems together are consistent with the automata network and $p_d = 0$ holds otherwise.

Definition 12.1 (*Completeness of the consistency test*) The consistency test whose result is denoted by p_d is said to be *complete* if the relation

$$p_c > 0 \;\Rightarrow\; p_d > 0$$

holds.

This definition allows a deviation of the result p_d from the true result p_c obtained by the centralised scheme. However, for all I/O pairs $(\boldsymbol{V}, \boldsymbol{W})$ that are consistent with the overall model $\mathcal{A}$ the decentralised scheme has to give a positive test result. On the other hand, the definition allows that the decentralised scheme delivers a positive result for inconsistent I/O pairs.

The attribute "complete" is used here in coincidence with the same attribute for the diagnostic result that is obtained by using complete consistency tests.

Description of a complete consistency test. As an extension of Eqs. (11.75) and (11.76) the following indicator functions p_i, $(i = 1, 2, \ldots, N)$ are defined:

$$p_i'(z_i, 0) = \begin{cases} 1 & \text{if } z_i \in \mathcal{Z}_{i0} \\ 0 & \text{else} \end{cases} \tag{12.9}$$

$$p_i'(z_i', k+1) = \left\lfloor \sum_{z_i \in \mathcal{Z}_i} \sum_{s_i \in \mathcal{S}_i} L_i(z_i', \bar{w}_{ki}, z_i, \bar{v}_{ki}, s_i)\, p_i'(z_i, k) \right\rfloor, \tag{12.10}$$

$(k = 0, 1, \ldots, k_e)$. The consistency result for the ith subsystem is

$$p_{di} = \left\lfloor \sum_{z_i' \in \mathcal{Z}_i} p_i'(z_i', k_e + 1) \right\rfloor . \tag{12.11}$$

All indicator functions together yield

$$p_d = \prod_{i=1}^{N} p_{di} . \tag{12.12}$$

Theorem 12.1 *If the I/O pair is consistent with the overall system $\mathcal{A}$, the indicator function p_d is positive:*

$$p_d > 0. \tag{12.13}$$

Proof The theorem is proved by showing that the relation

$$p_c \leq p_d \tag{12.14}$$

holds. Then the inequality (12.13) is satisfied whenever the I/O pair is consistent with the overall system, which is indicated by a positive left-hand side of Eq. (12.14).

The proof is done by induction over the length k_e of the I/O pairs. For $k_e = 0$, Eqs. (11.75) and (11.76) yield

$$\begin{aligned} p_c &= \left\lfloor \sum_{z' \in \mathcal{Z}} p'(z', 1) \right\rfloor \\ &= \left\lfloor \sum_{z' \in \mathcal{Z}} \sum_{z \in \mathcal{Z}_0} L(z', \bar{w}_0, z, \bar{v}_0) \right\rfloor . \end{aligned}$$

Equations (12.9) and (12.10) lead to the expression

$$\begin{aligned} p_d &= \prod_{i=1}^{N} p_{di} \\ &= \prod_{i=1}^{N} \left\lfloor \sum_{z_i' \in \mathcal{Z}_i} p_i'(z_i', 1) \right\rfloor \\ &= \prod_{i=1}^{N} \left\lfloor \sum_{z_i' \in \mathcal{Z}_i} \left\lfloor \sum_{z_i \in \mathcal{Z}_{0i}} \sum_{s_i \in \mathcal{S}_i} L(z_i', \bar{w}_{0i}, z_i, \bar{v}_{0i}, s_i) \right\rfloor \right\rfloor \end{aligned}$$

$$= \left\lfloor \prod_{i=1}^{N} \sum_{z_i' \in \mathcal{Z}_i} \sum_{z_i \in \mathcal{Z}_{0i}} \sum_{s_i \in \mathcal{S}_i} L(z_i', \bar{w}_{0i}, z_i, \bar{v}_{0i}, s_i) \right\rfloor .$$

As all factors are non-negative, the following inequality holds

$$\begin{aligned} & \left\lfloor \prod_{i=1}^{N} \sum_{z_i' \in \mathcal{Z}_i} \sum_{z_i \in \mathcal{Z}_{0i}} \sum_{s_i \in \mathcal{S}_i} L(z_i', \bar{w}_{0i}, z_i, \bar{v}_{0i}, s_i) \right\rfloor \\ \geq & \left\lfloor \sum_{z_i' \in \mathcal{Z}_i} \sum_{z_i \in \mathcal{Z}_{0i}} \prod_{i=1}^{N} L(z_i', \bar{w}_{0i}, z_i, \bar{v}_{0i}, K_i(\boldsymbol{r})) \right\rfloor \qquad (12.15) \\ = & \left\lfloor \sum_{z' \in \mathcal{Z}} \sum_{z \in \mathcal{Z}_0} L(\boldsymbol{z}', \bar{\boldsymbol{w}}_0, \boldsymbol{z}, \bar{\boldsymbol{v}}_0) \right\rfloor \\ = & \; p_{\mathrm{c}} \end{aligned}$$

since $K_i(\boldsymbol{r})$ is a specific element of the set $\mathcal{S}_i$.

For the induction step, assume that Eq. (12.14) holds for some $k_{\mathrm{e}} = k_{\mathrm{e}}$ and prove that it is also valid for $k_{\mathrm{e}} = k_{\mathrm{e}} + 1$. This proof can be done in a similar way, where again the inequality (12.15) is the crucial step that deletes the summation over all interconnection input symbols s_i. □

Remark 12.1 The decentralised test result p_{d} is the best possible result that is complete and can be determined if the test units have only access to a subsystem model and the I/O pair of the subsystem. Then the inequality (12.15) has to be used to remove the dependence of p_{d} upon the interconnection inputs $s_i \in \mathcal{S}_i$ that are unknown. □

Decentralised consistency test. For the decentralised consistency test, the following algorithm has to be carried out by all local test units. The results $p_{\mathrm{d}i}$ obtained by these units have to be lumped together to the overall test result given by Eq. (12.12).

Algorithm 12.1 *Consistency test of the i th subsystem*

Given: Automaton $\mathcal{A}_i$
I/O pair (12.7), (12.8)

1. Determine $p_i'(z_i, 0)$ by Eq. (12.9)

2. Apply Eq. (12.10) for $k = 0, 1, \ldots, k_{\mathrm{e}}$ to determine $p_i'(z_i', k_{\mathrm{e}} + 1)$

3. Determine $p_{\mathrm{d}i}$ by Eq. (12.11).

Result: Test result $p_{\mathrm{d}i}$ for the ith subsystem

The result for the overall system obtained by the N decentralised consistency test units is obtained by Eq. (12.12).

12.3.3 State Observation Result

The results $p_{\mathrm{d}i}$, $(i = 1, \ldots, N)$ generated by all the decentralised test units also provide a set of states in which the subsystems can be at time $k_\mathrm{e} + 1$:

$$\mathcal{Z}_{\mathrm{d}i}(k_\mathrm{e} + 1) = \{z_i' \in \mathcal{Z}_i \mid p_i'(z_i', k_\mathrm{e} + 1) > 0\}.$$

Hence, the state of the overall system is known to be in the set

$$\mathcal{Z}_\mathrm{d}(k_\mathrm{e} + 1) = \mathcal{Z}_{\mathrm{d}1}(k_\mathrm{e} + 1) \times \mathcal{Z}_{\mathrm{d}2}(k_\mathrm{e} + 1) \times \cdots \times \mathcal{Z}_{\mathrm{d}N}(k_\mathrm{e} + 1)$$

which satisfies the relation

$$\mathcal{Z}_\mathrm{d}(k_\mathrm{e} + 1) \supseteq \mathcal{Z}_\mathrm{c}(k_\mathrm{e} + 1). \tag{12.16}$$

Accordingly, the decentralised units yield a superset of the set of states $\boldsymbol{z}(k_\mathrm{e} + 1)$ in which the overall system can reside at time $k_\mathrm{e} + 1$.

12.4 Centralised Versus Decentralised Diagnosis

12.4.1 Completeness of the Diagnostic Result

The consistency test is now applied for solving the diagnostic problem stated in Sect. 12.1. The model of the subsystems and of the overall system have now to be dependent upon the system fault $\boldsymbol{f}$ or the subsystem fault f_i, respectively. Consequently, the state transition functions considered now have the fault as additional argument.

The aim is to determine a set of faults $\mathcal{F}_\mathrm{d}$ that is equal or a superset of the set $\mathcal{F}^*$ of fault candidates.

Definition 12.2 (*Completeness of the diagnostic result*) The diagnostic result $\mathcal{F}_\mathrm{d}$ is said to be *complete* if the relation

$$\mathcal{F}_\mathrm{d} \supseteq \mathcal{F}^* \tag{12.17}$$

holds.

Hence, a complete diagnostic result is given by a set of faults in which all faults occur for which the I/O pair is consistent with the model of the overall system.

12.4.2 Centralised Diagnosis

The diagnostic problem of the overall system is solved by using Algorithm 11.4 for the model $\mathcal{A}(\boldsymbol{f})$ successively for all faults $\boldsymbol{f} \in \mathcal{F}$. The notation $\mathcal{A}(\boldsymbol{f})$ differs from the symbol $\mathcal{A}_f$ used in Chap. 11 for a model that describes a system subject to fault f, because here the index is used to identify the subsystem concerned. If the I/O pair is found to be consistent with the model for some fault $\boldsymbol{f}$, then $\boldsymbol{f}$ is an element of the solution set $\mathcal{F}_\mathrm{c}$.

As, according to Lemma 11.4, the consistency test developed for the overall system gives the best possible result, the set $\mathcal{F}_\mathrm{c}$ coincides with the set of fault candidates $\mathcal{F}^*$. That is, the diagnostic method yields the best possible diagnostic result.

Algorithm 12.2 *Centralised fault diagnosis of the overall system*

Given: Automaton $\mathcal{A}(\boldsymbol{f})$ describing the system for all faults $\boldsymbol{f} \in \mathcal{F}$
I/O pair (11.71), (11.72)

1. For all $\boldsymbol{f} \in \mathcal{F}$ test the consistency of the I/O pair with the model $\mathcal{A}(\boldsymbol{f})$ by Algorithm 11.4.

2. If the I/O pair is consistent with the model for some fault $\boldsymbol{f} \in \mathcal{F}$, include $\boldsymbol{f}$ as new element into the set $\mathcal{F}_\mathrm{c}$

Result: Set of faults $\mathcal{F}_\mathrm{c}$

Theorem 12.2 (Completeness of the overall diagnostic result) *The diagnostic result obtained by Algorithm 12.2 is complete. Moreover, the relation*

$$\mathcal{F}_\mathrm{c} = \mathcal{F}^* \tag{12.18}$$

holds.

Equation (12.18) shows that the diagnostic result is not only complete but also precise in the sense that it does not include any fault that is not a fault candidate.

12.4.3 Decentralised Diagnosis

Decentralised diagnosis solves the diagnostic problem for the components separately and merges the results to get the diagnostic result for the overall system.

The decentralised diagnostic unit of the ith subsystem tests the consistency of the I/O pair (V_i, W_i) of the subsystem with the subsystem model $\mathcal{A}_i(f)$ set up for all subsystems faults $f \in \mathcal{F}_i$. If the consistency condition is satisfied, the fault f is inserted into the diagnostic result $\mathcal{F}_{\mathrm{d}i}$.

As the decentralised diagnostic algorithm has no information about the interconnection input s_i, the condition $p_{\mathrm{d}i} > 0$ for the indicator $p_{\mathrm{d}i}$ obtained from Eq. (12.11) is sufficient but not necessary for the consistency of the overall I/O pair with the overall model. Hence, a complete but not precise diagnostic result is obtained by the decentralised diagnostic unit.

Algorithm 12.3 *Decentralised fault diagnosis of the ith subsystem*

Given: Automaton $\mathcal{A}_i(f)$ describing the ith subsystem for all faults $f \in \mathcal{F}_i$
I/O pair (12.7), (12.8)

1. For all $f \in \mathcal{F}_i$ test the consistency of the I/O pair with the model $\mathcal{A}_i(f)$ by Algorithm 12.1.

2. If the I/O pair is consistent with the model $\mathcal{A}_i(f)$ for some fault $f \in \mathcal{F}_i$, include f as new element into the set $\mathcal{F}_{\mathrm{d}i}$

Result: Set of faults $\mathcal{F}_{\mathrm{d}i}$

The diagnostic result $\mathcal{F}_{\mathrm{d}}$ for the overall system is obtained from the diagnostic results $\mathcal{F}_{\mathrm{d}i}$ of the subsystems as follows:

$$\mathcal{F}_{\mathrm{d}} = \mathcal{F}_{\mathrm{d}1} \times \mathcal{F}_{\mathrm{d}2} \times \cdots \times \mathcal{F}_{\mathrm{d}N}. \tag{12.19}$$

Theorem 12.1 yields the following statement about the completeness of this diagnostic result.

Theorem 12.3 (Completeness of decentralised diagnosis) *The result $\mathcal{F}_{\mathrm{d}}$ obtained by the decentralised diagnostic units is complete:*

$$\mathcal{F}_{\mathrm{d}} \supseteq \mathcal{F}^*. \tag{12.20}$$

The diagostic result includes, in general, faults that are no fault candidates.

12.5 System Properties and Simplification of Diagnosis

12.5.1 Aim of Analysis

This section deals with properties of the automata network that allow to use a decentralised diagnostic scheme without a deterioration of the diagnostic result due to the lack of information about the interconnections of the subsystems. If the system possesses—at least temporarily—these properties, then the decentralised diagnostic units can test the consistency of the local I/O pairs with the model of the subsystems and get the same result as the centralised unit, which uses the overall I/O pair and the model of the overall system.

The main idea of the further considerations is to look for situations in which a subsystem makes state transitions independently of other subsystems. Two situations are investigated in more detail:

- **Subsystem autonomy**: A subsystem has an (temporal) autonomy if its state transitions are not influenced by the interconnection input. Then the consistency of the I/O pair of the subsystem does not depend on the unknown interconnection signals.
- **Asynchronous state transitions**: If a subsystem is caused to make a state transition by its local input whereas the other subsystems do not get any input and, thus, do not make state transitions, then the consistency of the local I/O pair is not influenced by subsystem interactions.

As the consistency test is the basis for any diagnostic method, the rest of this section is concerned with the consistency test for a model that has no reference to any fault.

12.5.2 Autonomy of Subsystems

Autonomy is the property that a subsystem behaves independently of the state or state transitions of the other subsystems although it is coupled with these subsystems within the automata network.

Definition 12.3 (*Subsystem autonomy*) A subsystem $\mathcal{A}_i$ in state $z_i \in \mathcal{Z}_i$ is said to behave *autonomously* from other subsystems if there exists a function $\tilde{L}_i$ such that the equation

$$L_i(z_i', w_i, z_i, v_i, s_i) = \tilde{L}_i(z_i', w_i, z_i, v_i) \tag{12.21}$$

holds for this state z_i, all $v_i \in \mathcal{V}_i$ and all $s_i \in \mathcal{S}_i$.

That is, the state transition $z_i \xrightarrow{v_i/w_i} z_i'$ does not depend on the interconnection input s_i. Denote the set of states z_i for which the autonomy condition (12.21) is satisfied by $\mathcal{Z}_i^{\text{aut}}$:

$$\mathcal{Z}_i^{\text{aut}} = \{z_i \in \mathcal{Z}_i \mid \exists \tilde{L}_i : \; L_i(z_i', w_i, z_i, v_i, s_i) = \tilde{L}_i(z_i', w_i, z_i, v_i) \\ \text{for all} \;\; v_i \in \mathcal{V}_i, s_i \in \mathcal{S}_i\}.$$

If Eq. (12.21) holds for all $z_i \in \mathcal{Z}_i$, the ith subsystem acts completely independently of the other subsystems and can be analysed completely separately from the remainder of the composite system. This is the trivial case where the decentralised diagnostic unit $\mathcal{D}_i$ gets the best possible diagnostic result. The more interesting case concerns subsystems where the autonomy condition (12.21) is satisfied for some but not all states z_i. In this case the decentralised unit $\mathcal{D}_i$ gets the best possible result as long as the subsystem state remains in the set $\mathcal{Z}_i^{\text{aut}}$.

Two autonomous subsystems. The simplification that results from autonomous state transitions is considered now for an automata network with two subsystems. Assume that at time k the decentralised test units together have obtained the same consistency result as the centralised test unit, which means that the relation

$$p_1'(z_1, k) \cdot p_2'(z_2, k) = p'\left(\begin{pmatrix} z_1 \\ z_2 \end{pmatrix}, k\right) \tag{12.22}$$

holds. That is, the subsystems are in states that occur in the sets

$$\mathcal{Z}_1(k) = \{z_1 \in \mathcal{Z}_1 \mid p_1'(z_1, k) > 0\}$$
$$\mathcal{Z}_2(k) = \{z_2 \in \mathcal{Z}_2 \mid p_2'(z_2, k) > 0\}$$

and the automata network in one of the states of the set

$$\mathcal{Z}(k) = \{\boldsymbol{z} \in \mathcal{Z} \mid p'(\boldsymbol{z}, k) > 0\} = \mathcal{Z}_1(k) \times \mathcal{Z}_2(k).$$

Assume furthermore that both subsystems satisfy the autonomy conditions (12.21) for all $z_i \in \mathcal{Z}_i(k)$:

$$L_1(z_1', w_1, z_1, v_1, s_1) = \tilde{L}_1(z_1', w_1, z_1, v_1) \quad \text{for all} \;\; z_1 \in \mathcal{Z}_1(k), v_1 \in \mathcal{V}_1, s_1 \in \mathcal{S}_1$$
$$L_2(z_2', w_2, z_2, v_2, s_2) = \tilde{L}_2(z_2', w_2, z_2, v_2) \quad \text{for all} \;\; z_2 \in \mathcal{Z}_2(k), v_2 \in \mathcal{V}_2, s_2 \in \mathcal{S}_2.$$

Then the consistency tests carried out by the centralised unit applying Eq. (11.76) and the decentralised units using Eq. (12.10) lead to the same result as the following equations show:

$$p_1(z_1', k+1) \cdot p_2(z_2', k+1)$$
$$= \left\lfloor \sum_{z_1 \in \mathcal{Z}_1} \sum_{s_1 \in \mathcal{S}_1} L_1(z_1', \bar{w}_{k1}, z_1, \bar{v}_{k1}, s_1) \cdot p_1(z_1, k) \right\rfloor$$

$$\cdot \left\lfloor \sum_{z_2 \in \mathcal{Z}_2} \sum_{s_2 \in \mathcal{S}_2} L_2(z_2', \bar{w}_{k2}, z_2, \bar{v}_{k2}, s_2) \cdot p_2(z_2, k) \right\rfloor$$

$$= \left\lfloor \sum_{z_1 \in \mathcal{Z}_1(k)} \sum_{s_1 \in \mathcal{S}_1} L_1(z_1', \bar{w}_{k1}, z_1, \bar{v}_{k1}, s_1) \right\rfloor$$

$$\cdot \left\lfloor \sum_{z_2 \in \mathcal{Z}_2(k)} \sum_{s_2 \in \mathcal{S}_2} L_2(z_2', \bar{w}_{k2}, z_2, \bar{v}_{k2}, s_2) \right\rfloor$$

$$= \left\lfloor \sum_{z_1 \in \mathcal{Z}_1(k)} \tilde{L}_1(z_1', \bar{w}_{k1}, z_1, \bar{v}_{k1}) \cdot \sum_{z_2 \in \mathcal{Z}_2(k)} \tilde{L}_2(z_2', \bar{w}_{k2}, z_2, \bar{v}_{k2}) \right\rfloor$$

$$p(\boldsymbol{z}', k+1)$$

$$= \left\lfloor \sum_{z \in \mathcal{Z}} L(\boldsymbol{z}', \bar{\boldsymbol{w}}_{k1}, \boldsymbol{z}, \bar{\boldsymbol{v}}_{k1}) \cdot p(\boldsymbol{z}, k) \right\rfloor$$

$$= \left\lfloor \sum_{z \in \mathcal{Z}(k)} L(\boldsymbol{z}', \bar{\boldsymbol{w}}_{k1}, \boldsymbol{z}, \bar{\boldsymbol{v}}_{k1}) \right\rfloor$$

$$= \left\lfloor \sum_{z_1 \in \mathcal{Z}_1(k)} L_1(z_1', \bar{w}_{k1}, z_1, \bar{v}_{k1}, K_1(\boldsymbol{r})) \cdot \sum_{z_2 \in \mathcal{Z}_2(k)} L_2(z_2', \bar{w}_{k2}, z_2, \bar{v}_{k2}, K_2(\boldsymbol{r})) \right\rfloor$$

$$= \left\lfloor \sum_{z_1 \in \mathcal{Z}_1(k)} \tilde{L}_1(z_1', \bar{w}_{k1}, z_1, \bar{v}_{k1}) \cdot \sum_{z_2 \in \mathcal{Z}_2(k)} \tilde{L}_2(z_2', \bar{w}_{k2}, z_2, \bar{v}_{k2}) \right\rfloor .$$

Hence the lack of information under which the decentralised units have to test the consistency of the I/O pairs of the subsystems does not deteriorate the result in comparison to the centralised test.

Series connection of two subsystems. The situation changes if only one of the two subsystems acts autonomously. Assume, as an example, that Subsystem 1 satisfies the autonomy condition (12.21) whereas the state transition of Subsystem 2 depends upon the interconnection signal s_2. Then the automata network consists of a series connection of the two subsystems. Consider again the situation described by Eq. (12.22). Then the decentralised scheme yields the following result:

$$p_1'(z_1', k+1) = \left\lfloor \sum_{z_1 \in \mathcal{Z}_1(k)} \sum_{s_1 \in \mathcal{S}_1} L_1(z_1', \bar{w}_{k1}, z_1, \bar{v}_{k1}, s_1) \right\rfloor$$

$$
\begin{aligned}
&= \left\lfloor \sum_{z_1 \in \mathcal{Z}_1(k)} \tilde{L}_1(z_1', \bar{w}_{k1}, z_1, \bar{v}_{k1}) \right\rfloor \\
&= \left\lfloor \sum_{z_1 \in \mathcal{Z}_1(k)} L_1(z_1', \bar{w}_{k1}, z_1, \bar{v}_{k1}, K_1(\boldsymbol{r})) \right\rfloor \\
p_2'(z_2', k+1) &= \left\lfloor \sum_{z_2 \in \mathcal{Z}_2(k)} \sum_{s_2 \in \mathcal{S}_2} L_2(z_2', \bar{w}_{k2}, z_2, \bar{v}_{k2}, s_2) \right\rfloor \\
&\geq \left\lfloor \sum_{z_2 \in \mathcal{Z}_2(k)} L_2(z_2', \bar{w}_{k2}, z_2, \bar{v}_{k2}, K_2(\boldsymbol{r})) \right\rfloor .
\end{aligned}
$$

The comparison with the result produced by the centralised test unit gives the following inequality:

$$
\begin{aligned}
p'(\boldsymbol{z}', k+1) &= \left\lfloor \sum_{z_1 \in \mathcal{Z}_1(k)} L_1(z_1', \bar{w}_{k1}, z_1, \bar{v}_{k1}, K_1(\boldsymbol{r})) \right. \\
&\qquad \left. \cdot \sum_{z_2 \in \mathcal{Z}_2(k)} L_2(z_2', \bar{w}_{k2}, z_2, \bar{v}_{k2}, K_2(\boldsymbol{r})) \right\rfloor \\
&\leq p_1'(z_1', k+1) \cdot p_2'(z_2', k+2).
\end{aligned}
$$

Note that the deterioration of the test result only concerns Subsystem 2. This becomes obvious if the sets of states are compared in which the system can be at time $k+1$. The decentralised scheme delivers the sets

$$
\begin{aligned}
\mathcal{Z}_1(k+1) &= \{z_1 \in \mathcal{Z}_1 \mid p_1'(z_1, k+1) > 0\} \\
\mathcal{Z}_2(k+1) &= \{z_2 \in \mathcal{Z}_2 \mid p_2'(z_2, k+1) > 0\},
\end{aligned}
$$

whereas the centralised test results in the set

$$
\mathcal{Z}(k+1) = \{\boldsymbol{z} \in \mathcal{Z} \mid p'(\boldsymbol{z}, k+1) > 0\}.
$$

The projection of the set $\mathcal{Z}(k+1)$ onto the first component coincides with the set $\mathcal{Z}_1(k+1)$ obtained by the decentralised unit of Subsystem 1:

$$
P_{\mathcal{Z}_1}(\mathcal{Z}(k+1)) = \left\{ z_1 \in \mathcal{Z}_1 \;\middle|\; \exists \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \in \mathcal{Z}(k+1) \right\} = \mathcal{Z}_1(k+1).
$$

However, the projection onto the state set $\mathcal{Z}_2$ shows the deterioration of the consistency test result due to the lack of coupling information:

$$P_{\mathcal{Z}_2}(\mathcal{Z}(k+1)) \subseteq \mathcal{Z}_2(k+1).$$

12.5.3 Asynchronous State Transitions

When considering an I/O pair $(\boldsymbol{V}, \boldsymbol{W})$ it had been assumed until now that the components of the vectors $\boldsymbol{v}(k)$ and $\boldsymbol{w}(k)$ represent input symbols that occur at all subsystems simultaneously. The input $\boldsymbol{v}(k)$ causes synchronous state transitions of the subsystems, which in turn generate simultaneously the output symbols that occur in the vector $\boldsymbol{w}(k)$.

This section shows that the methods described so far are also applicable for automata networks that perform asynchronous state transitions. To represent such state transitions, the empty symbol ε is introduced as additional elements of the input and output alphabets of all subsystems including the alphabets of the interconnection signals and of the overall systems. This symbol indicates that "no information" is transmitted by the corresponding signal:

$$\begin{aligned} \mathcal{V}_1 = \{\varepsilon, 1\}, \quad \mathcal{V}_2 = \{\varepsilon, 2\} \\ \mathcal{W}_1 = \{\varepsilon, 0, 1\}, \quad \mathcal{W}_2 = \{\varepsilon, 0, 1\} \\ \mathcal{S}_1 = \mathcal{S}_2 = \mathcal{R}_1 = \mathcal{R}_2 = \{\varepsilon, 0, 0\}. \end{aligned}$$

Several situations can be taken into account when extending the subsystem models to accept empty input symbols:

- **No state transition**: If for the state transition function the relation

$$L_i(z_i, \varepsilon, z_i, \varepsilon, s_i) = 1 \quad \text{for all } z_i \in \mathcal{Z}_i, s_i \in \mathcal{S}_i \tag{12.23}$$

  holds, a subsystem that gets the empty input symbol does not change its state, $(z_i' = z_i)$ and does not generate any output symbol.
- **State transition caused by other subsystems**: If the state transition function has the property

$$\exists z_i', w_i: \quad L_i(z_i', w_i, z_i, \varepsilon, s_i) = 1 \quad \text{for some } z_i \in \mathcal{Z}_i, s_i \in \mathcal{S}_i, \tag{12.24}$$

  the state transition $z_i \to z_i'$ of the ith subsystem is caused by the input symbol s_i. The state transition is initiated by the subsystem that generates the interconnection input s_i.
- **Blocked state transitions**: If the state transition function satisfies the condition

$$L_i(z_i, \varepsilon, z_i, v_i, \varepsilon) = 1 \quad \text{for all } z_i \in \mathcal{Z}_i, v_i \in \mathcal{V}_i, \tag{12.25}$$

then the state transition of the ith subsystem can be blocked by other subsystems that make the interconnection input s_i to the ith subsystem equal to the empty symbol. This blockage can be caused by the jth subsystem that gets the empty input signal $v_j = \varepsilon$ and transfers this blockage via its interconnection output

$$r_j = F_j(z_j, \varepsilon) = \varepsilon$$

towards the interconnection input $s_i = \varepsilon$ of the ith subsystem.

The properties (12.23) and (12.25) may be satisfied for some instead of all subsystem states or subsystem inputs. Then the phenomena described above hold only in specific situations.

Example 12.2 Fault detection of two asynchronous subsystems
Consider again the two coupled automata of Example 12.1. The state transition functions L_1 and L_2 are given by automaton graphs in Fig. 12.5 whose edges have the labels $v_i, s_i/w_i, r_i$.

In addition to the properties investigated in Example 12.1 the following behaviour of the automata network in case of empty inputs is introduced now. If Subsystem 1 gets a nonempty input signal it carries out a state transition, which is an asynchronous transition if Subsystem 2 does not get any nonempty input. In case of an empty input, Subsystem 1 blocks any state transition of Subsystem 2. This situation is described by the following relations that hold in addition to the state transitions shown in the figure:

$$\begin{aligned}
F_1(z_1, \varepsilon) &= \varepsilon && (12.26)\\
F_2(z_2, \varepsilon) &= \varepsilon \\
L_1(z_1, \varepsilon, z_1, \varepsilon, s_1) &= 1 && (12.27)\\
L_1(z_1', w_i, z_1, v_1, \varepsilon) &= L_1(z_1', w_i, z_1, v_1, 1) && (12.28)\\
L_2(z_2, \varepsilon, z_2, \varepsilon, s_2) &= 1 && (12.29)\\
L_2(z_2, \varepsilon, z_2, v_2, \varepsilon) &= 1. && (12.30)
\end{aligned}$$

These formulas hold true for all free variables occurring in the functions considered. Both the blockage of a state transition and a state transition where the successor state coincides with its predecessor are symbolised by a state transition $z_i \to z_i'$. However, in the former case the output $w_i = \varepsilon$ is generated whereas in the latter case some output $w_i \neq \varepsilon$ is issued.

Centralised diagnosis. The consistency of the I/O pair

$$\begin{aligned}
\boldsymbol{V} &= \left(\begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} \varepsilon \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ \varepsilon \end{pmatrix} \right) \\
\boldsymbol{W} &= \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \varepsilon \\ \varepsilon \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ \varepsilon \end{pmatrix} \right)
\end{aligned}$$

is tested first by the centralised unit by means of Algorithm 11.4. The model of the overall system, which has the state set

$$\mathcal{Z} = \left\{ \begin{pmatrix} 1 \\ 3 \end{pmatrix}, \begin{pmatrix} 1 \\ 4 \end{pmatrix}, \begin{pmatrix} 2 \\ 3 \end{pmatrix}, \begin{pmatrix} 2 \\ 4 \end{pmatrix} \right\}$$

is built by Eq. (12.4) for the arguments that occur during the test. The test result $p'(z, k)$ is written down only for those states z for which it has a positive value.

1. Due to Eq. (12.5), the initialisation yields

$$p'\left(\begin{pmatrix} 1 \\ 4 \end{pmatrix}, 0\right) = 1$$
$$p'\left(\begin{pmatrix} 2 \\ 4 \end{pmatrix}, 0\right) = 1.$$

2. For the inputs $v_1(0) = 1$, $v_2(0) = 2$ and outputs $w(0) = 0$, $w_2(0) = 0$ the following results are obtained:

$$\begin{aligned} & p'(z', 1) \\ &= \left\lfloor \sum_{z \in \mathcal{Z}} L\left(z', \begin{pmatrix} 0 \\ 0 \end{pmatrix}, z, \begin{pmatrix} 1 \\ 2 \end{pmatrix}\right) \right\rfloor \cdot p'(z, 0) \\ &= \left\lfloor L\left(z', \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 4 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \end{pmatrix}\right) + L\left(z', \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 2 \\ 4 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \end{pmatrix}\right) \right\rfloor \\ &= \lfloor L_1(z'_1, 0, 1, 1, F_2(4, 2)) \cdot L_2(z'_2, 0, 4, 2, F_1(1, 1)) \\ &\quad + L_1(z'_1, 0, 2, 1, F_2(4, 2)) \cdot L_2(z'_2, 0, 4, 2, F_1(2, 1)) \rfloor \\ &= \lfloor L_1(z'_1, 0, 1, 1, 1) \cdot L_2(z'_2, 0, 4, 2, 0) + L_1(z'_1, 0, 2, 1, 1) \cdot L_2(z'_2, 0, 4, 2, 1) \rfloor. \end{aligned}$$

   In the last line, $L_1(z'_1, 0, 2, 1, 1)) = 0$ holds, because Subsystem 1 generates the output $w(0) = 1$ if it gets in state $z_1 = 2$ the inputs $v_1 = 1, s_1 = 1$. The preceding line is equal to one for $z'_1 = 2$ and $z'_2 = 4$:

$$p'\left(\begin{pmatrix} 2 \\ 4 \end{pmatrix}, 1\right) = 1.$$

3. For the inputs $v_1(1) = \varepsilon$, $v_2(1) = 2$ only Subsystem 2 is forced from outside to make a state transition. The following calculations show that in accordance with the earlier description of the network properties for empty inputs both subsystems are blocked by the empty input signal to Subsystem 1:

$$
\begin{aligned}
p'(z',2) &= \left\lfloor \sum_{z\in\mathcal{Z}} L\left(z', \begin{pmatrix}\varepsilon\\ \varepsilon\end{pmatrix}, z, \begin{pmatrix}\varepsilon\\ 2\end{pmatrix}\right)\right\rfloor \cdot p(z,1) \\
&= \left\lfloor L\left(z', \begin{pmatrix}\varepsilon\\ \varepsilon\end{pmatrix}, \begin{pmatrix}2\\ 4\end{pmatrix}, \begin{pmatrix}\varepsilon\\ 2\end{pmatrix}\right)\right\rfloor \\
&= \lfloor L_1(z_1', \varepsilon, 2, \varepsilon, 1) \cdot L_2(z_2', \varepsilon, 4, 2, \varepsilon)\rfloor
\end{aligned}
$$

(cf. Eq. (12.26)). The state transition functions (12.27), (12.30) of the subsystems yield $z_1' = z_1 = 2$ and $z_2' = z_2 = 4$ as the only successor states for which the product is non-zero:

$$
p'\left(\begin{pmatrix}2\\ 4\end{pmatrix}, 2\right) = 1.
$$

The blockage of both subsystems by the empty input to Subsystem 1 can be seen in the empty outputs of both components, which indicate that no transitions have occurred.

4. For $v_1(2) = 1$, $v_2(2) = 2$ both subsystems may change their states. The evaluation of the I/O pair leads to

$$
\begin{aligned}
p'(z',3) &= \left\lfloor \sum_{z\in\mathcal{Z}} L\left(z', \begin{pmatrix}1\\ 1\end{pmatrix}, z, \begin{pmatrix}1\\ 2\end{pmatrix}\right) \cdot p'(z,2)\right\rfloor \\
&= \left\lfloor L\left(z', \begin{pmatrix}1\\ 1\end{pmatrix}, \begin{pmatrix}2\\ 4\end{pmatrix}, \begin{pmatrix}1\\ 2\end{pmatrix}\right)\right\rfloor \\
&= \lfloor L_1(z_1', 1, 2, 1, 1) \cdot L_2(z_2', 1, 4, 2, 1)\rfloor
\end{aligned}
$$

and

$$
p'\left(\begin{pmatrix}1\\ 3\end{pmatrix}, 3\right) = 1.
$$

5. For the input $v_1(3) = 1$, $v_2(3) = \varepsilon$ Subsystem 1 makes an asynchronous state transition. With the outputs $w_1(3) = 0$ and $w_2(3) = \varepsilon$ one gets

$$
\begin{aligned}
p'(z',4) &= \left\lfloor \sum_{z\in\mathcal{Z}} L\left(z', \begin{pmatrix}0\\ \varepsilon\end{pmatrix}, z, \begin{pmatrix}1\\ \varepsilon\end{pmatrix}\right) \cdot p'(z,3)\right\rfloor \\
&= \left\lfloor L\left(z', \begin{pmatrix}0\\ \varepsilon\end{pmatrix}, \begin{pmatrix}1\\ 3\end{pmatrix}, \begin{pmatrix}1\\ \varepsilon\end{pmatrix}\right)\right\rfloor \\
&= \lfloor L_1(z_1', 0, 1, 1, \varepsilon) \cdot L_2(z_2', \varepsilon, 3, \varepsilon, 0)\rfloor \\
&= \lfloor L_1(z_1', 0, 1, 1, 1) \cdot L_2(z_2', \varepsilon, 3, \varepsilon, 0)\rfloor
\end{aligned}
$$

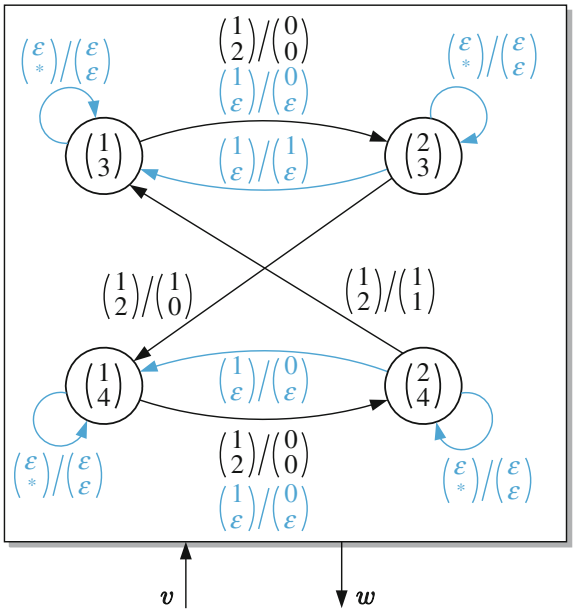


Fig. 12.7 Overall system with asynchronous state transitions

(cf. Eq. (12.28)). Equation (12.29) leads to

$$p'\left(\binom{2}{3}, 4\right) = 1.$$

The same results are obtained if first, the overall system model is built (Fig. 12.7) and second, the I/O pair of the overall system is tested with respect to this model. Then, however, the determination of the interconnection signals does not become obvious as intermediate result, which is important for the comparison of the two schemes later on.

The figure shows which state transitions may occur in the network due to the asynchronous movement of Subsystem 1. The asterisk denotes any input v_2.

Decentralised diagnosis. Now consider the decentralised units. Algorithm 12.1 yields for Subsystem 1 the following results:

1. The initialisation gives

$$\begin{aligned} p_1'(1,0) &= 1 \\ p_1'(2,0) &= 1. \end{aligned}$$

2. For $v_1(0) = 1$ and $w_1(0) = 0$ the algorithm delivers the result

$$p_1'(z_1', 1) = \left\lfloor \sum_{z_1 \in \mathcal{Z}_1} \sum_{s_1 \in \mathcal{S}_1} L_1(z_1', 0, z_1, 1, s_1) \cdot p_1'(z_1, 0) \right\rfloor$$

$$= \left\lfloor \sum_{s_1 \in \mathcal{S}_1} \left(L_1(z_1', 0, 1, 1, s_1) + L_1(z_1', 0, 2, 1, s_1) \right) \right\rfloor. \quad (12.31)$$

The first addend is non-zero for $z_1' = 1$ and $z_1' = 2$ whereas the second is non-zero for $z_1' = 2$. Hence, the result of this step is

$$p_1'(1, 1) = 1$$
$$p_1'(2, 1) = 1.$$

3. For $v_1(1) = \varepsilon$ and $w_1(1) = \varepsilon$ the algorithm yields

$$p_1'(z_1', 2) = \left\lfloor \sum_{s_1 \in \mathcal{S}_1} \left(L_1(z_1', \varepsilon, 1, \varepsilon, s_1) + L_1(z_1', \varepsilon, 2, \varepsilon, s_1) \right) \right\rfloor,$$

which according to Eq. (12.27) is non-zero for $z_1' = z_1$ and leads to

$$p_1'(1, 2) = 1$$
$$p_1'(2, 2) = 1.$$

No state transition has occurred.

4. For the I/O pair $v_1(2) = 1$, $w_1(2) = 1$ the sum

$$p_1'(z_1', 3) = \left\lfloor \sum_{s_1 \in \mathcal{S}_1} \left(L_1(z_1', 1, 1, 1, s_1) + L_1(z_1', 1, 2, 1, s_1) \right) \right\rfloor$$

is non-zero only for $z_1' = 1$:

$$p_1'(1, 3) = 1.$$

Hence, the state $z_1(3) = 1$ is unambiguously determined.

5. For the I/O pair $v_1(3) = 1$, $w_1(3) = 0$ the relation

$$p_1'(z_1', 4) = \left\lfloor \sum_{s_1 \in \mathcal{S}_1} L_1(z_1', 0, 1, 1, s_1) \right\rfloor$$

leads to two possible successor states z_1':

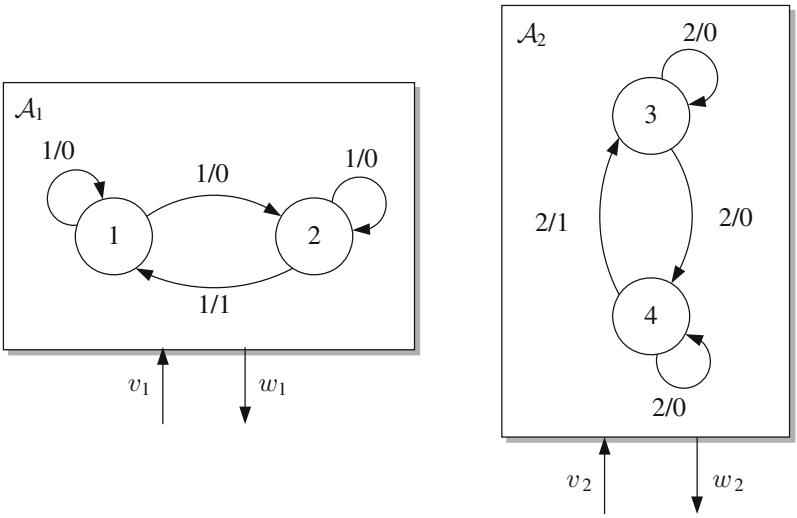


Fig. 12.8 Model of the isolated subsystems

$$p_1'(1,4) = 1$$
$$p_1'(2,4) = 1.$$

Due to the summation over the interconnection input s_1, the test algorithm deals with Subsystem 1 in such a way that it allows all state transitions that are possible for some interconnection input. Further, the interconnection output r_1 does not play any role in the test. This means that instead of the model $\mathcal{A}_1$ shown in the left part of Fig. 12.7 the simplified model depicted in the left part of Fig. 12.8 is used. This model results from the former one after deleting the information about the interconnection input s_1 and the interconnection output r_1, both of which are written as part of the labels of the state transitions. The result is a nondeterministic automaton.

For Subsystem 2 the decentralised test unit uses Algorithm 12.1 to get the following results:

1. The initialisation gives

$$p_2'(4,0) = 1.$$

2. For $v_2(0) = 2$ and $w_2(0) = 0$ the result is

$$p_2'(z_2', 1) = \left\lfloor \sum_{z_2 \in \mathcal{Z}_2} \sum_{s_2 \in \mathcal{S}_2} L_2(z_1', 0, z_2, 1, s_2) \cdot p_2'(z_2, 0) \right\rfloor$$
$$= \left\lfloor \sum_{s_2 \in \mathcal{S}_2} L_2(z_2', 0, 4, 2, s_2) \right\rfloor$$

which leads to

$$p_2'(4,1) \;=\; 1.$$

3. For $v_2(1) = 2$ and $w_2(1) = \varepsilon$ the relation

$$p_2'(z_2',2) \;=\; \left\lfloor \sum_{s_2 \in \mathcal{S}_2} L_2(z_2', \varepsilon, 4, 2, s_2) \right\rfloor$$

and Eq. (12.30) yield a positive result for $s_2 = \varepsilon$, for which $z_2' = z_2 = 4$ holds and, thus,

$$p_2'(4,2) \;=\; 1.$$

The fact that no state transition occurred can be recognised by the decentralised unit due to the empty output.

4. The I/O pair $v_2(2) = 2$, $w_2(2) = 1$ leads to

$$p_2'(z_2',3) \;=\; \left\lfloor \sum_{s_2 \in \mathcal{S}_2} L_2(z_2', 1, 4, 2, s_2) \right\rfloor$$

which is non-zero only for $z_2' = 3$.

$$p_2'(3,3) \;=\; 1.$$

5. Finally, the I/O pair $v_2(3) = \varepsilon$, $w_2(3) = \varepsilon$ leads to

$$p_2'(z_2',4) \;=\; \left\lfloor \sum_{s_2 \in \mathcal{S}_2} L_2(z_2', \varepsilon, 3, \varepsilon, s_2) \right\rfloor$$

and together with Eq. (12.29)

$$p_2'(3,4) \;=\; 1.$$

The calculations described above show that asynchronous state transitions can be considered by the consistency test.

Comparison of the results. The deviations of the results obtained by the decentralised units from the outcome of the centralised unit can be seen by comparing the state sets in which the system is known to be after the evaluation of the I/O pair for the time horizon $0 \ldots k$ with $k = 0, 1, \ldots, 4$:

k	$\mathcal{Z}(k)$	$\mathcal{Z}_1(k) \times \mathcal{Z}_2(k)$
0	$\left\{\begin{pmatrix}1\\4\end{pmatrix}, \begin{pmatrix}2\\4\end{pmatrix}\right\}$	$= \left\{\begin{pmatrix}1\\4\end{pmatrix}, \begin{pmatrix}2\\4\end{pmatrix}\right\}$
1	$\left\{\begin{pmatrix}2\\4\end{pmatrix}\right\}$	$\subset \left\{\begin{pmatrix}1\\4\end{pmatrix}, \begin{pmatrix}2\\4\end{pmatrix}\right\}$
2	$\left\{\begin{pmatrix}2\\4\end{pmatrix}\right\}$	$\subset \left\{\begin{pmatrix}1\\4\end{pmatrix}, \begin{pmatrix}2\\4\end{pmatrix}\right\}$
3	$\left\{\begin{pmatrix}1\\3\end{pmatrix}\right\}$	$= \left\{\begin{pmatrix}1\\3\end{pmatrix}\right\}$
4	$\left\{\begin{pmatrix}2\\3\end{pmatrix}\right\}$	$\subset \left\{\begin{pmatrix}1\\3\end{pmatrix}, \begin{pmatrix}2\\3\end{pmatrix}\right\}$

The deterioration of the observation result for the time $k = 1$ is brought about by the lack of information about the interconnection signals for the decentralised test units. The set $\mathcal{Z}(k)$ is determined by the centralised unit by processing the I/O pair

$$(\boldsymbol{v}(0), \boldsymbol{w}(0)) = \left(\begin{pmatrix}1\\2\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}\right).$$

The result is that the system can have been at time $k = 0$ only in the state

$$z(0) = \begin{pmatrix}1\\4\end{pmatrix}$$

and jumped to the state

$$z(1) = \begin{pmatrix}2\\4\end{pmatrix}$$

(cf. Fig. 12.6). The decentralised test by means of the model of Subsystem 1 and the I/O pair $(v_1(0), w_1(0)) = (1, 0)$ leads to Eq. (12.31), which is non-zero for $z_1' = 1$ and $z_1' = 2$. These states can be reached by choosing the interconnection input $s_1 = 0$ or $s_1 = 1$, respectively. Only if the model and the I/O pair of Subsystem 2 were known it could be recognised that the interconnection input is fixed to $s_1 = 1$, and, hence Subsystem 1 cannot jump to the state $z_1' = 1$. Similar considerations explain why for $k = 2$ and $k = 4$ the results of the decentralised test are worse than the centralised result. □

12.5.4 Extensions

Nondeterministic systems. The comparison of the centralised and decentralised test schemes can be extended to non-deterministic discrete-event systems without any problems. For non-deterministic systems, for a given state z and input v more than one successor state z' may occur. With respect to the ith subsystem this means that for a given triple (z_i, v_i, s_i) the relation

$$L_i(z_i', w_i, z_i, v_i, s_i) = 1$$

may hold for several states $z_i' \in \mathcal{Z}_i$ and output symbols $w_i \in \mathcal{W}_i$. The definition of the system behaviour as well as of the consistency of an I/O pair with the model remains the same.

In the most general form, the subsystem models are represented by a function that includes the coupling function F_i and the state transition function L_i. This function

$$L_i : \mathcal{Z} \times \mathcal{W} \times \mathcal{R} \times \mathcal{Z} \times \mathcal{V} \times \mathcal{S} \to \{0, 1\} \tag{12.32}$$

associates with any tupel

$$(z_i', w_i, r_i, z_i, v_i, s_i)$$

the value 1 if the subsystem may jump from the current state z_i towards the state z_i' if it gets the input v_i and the interconnection input s_i and generates the output w_i and the interconnection output r_i. Then the representation of the composition rule (12.4) changes towards

$$L\left(\begin{pmatrix} z_1' \\ z_2' \\ \vdots \\ z_N' \end{pmatrix}, \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix}, \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{pmatrix}, \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix}\right) = \left\lfloor \sum_{r_1 \in \mathcal{R}_1} \sum_{r_2 \in \mathcal{R}_2} \dots \sum_{r_N \in \mathcal{R}_N} \prod_{i=1}^{N} L_i(z_i', w_i, r_i, z_i, v_i, K_i(\boldsymbol{r})) \right\rfloor .$$

General subsystem models. In the subsystem models considered in this chapter the interconnection output r_i of the ith subsystem depends upon the current state z_i and the input v_i, but not on the interconnection input s_i. In the non-deterministic case, the interconnection output can also depend on the successor state z_i' and on the output w_i that is generated during the state transition. The latter case is already included in the definition of the function L_i by Eq. (12.32).

In all these cases, the automaton network need not be well-defined. An automata network is called well-defined if the overall network belongs to the same class of

automata as the component models [211]. This is particularly important for deterministic automata, which are the main concern of this chapter. For this class of automata, the network has to be tested to see whether it is well-defined. An "algebraic loop" may destroy the deterministic behaviour if for an overall state $\boldsymbol{z}$ and input $\boldsymbol{v}$ of the network the interconnection signals turn out not to be unambiguously defined. Then, no or more than one state transitions are possible which violates the property of the overall system to have a unambiguously defined successor state for all current states and inputs.

This problem has been mentioned in [84] and dealt with in more detail in [211] for deterministic systems. The references show that additional tests within the consistency test elaborated here are necessary to decide about the well posedness of the overall system. This test can only be made with the information about the overall system. Therefore, it can only be included in the centralised consistency test whereas the decentralised tests get wrong results if the system turns out not to be well-defined because this situation cannot be recognised by any decentralised test scheme due to the lack of information about the overall system.

12.6 Exercises

Exercise 12.1 Modelling of a mountain railway
Figure 12.9 shows a bird's eye view on a mountain railway together with the discrete-event models of the two cabins. For the input 1 a cabin can move downward (from right to left in the figure) and for the input 2 upward. The rails between both stations are partitioned into five

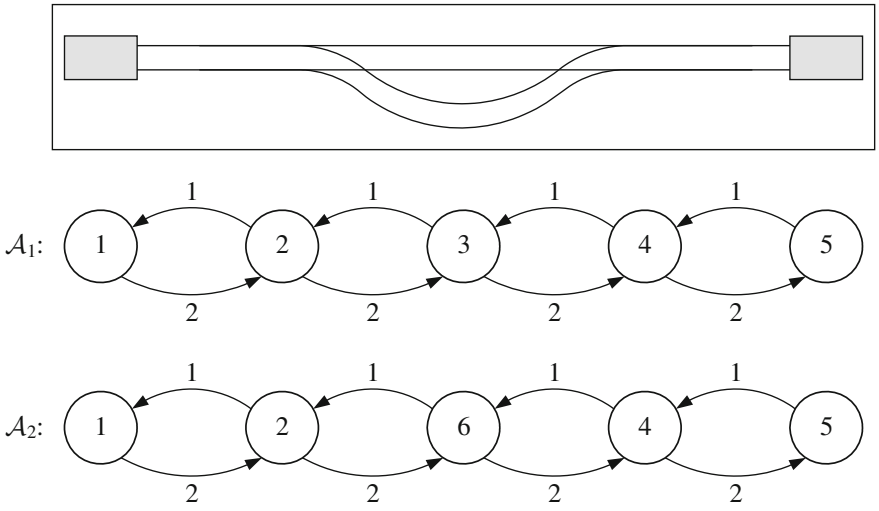


Fig. 12.9 Sketch and subsystem models of a mountain railway

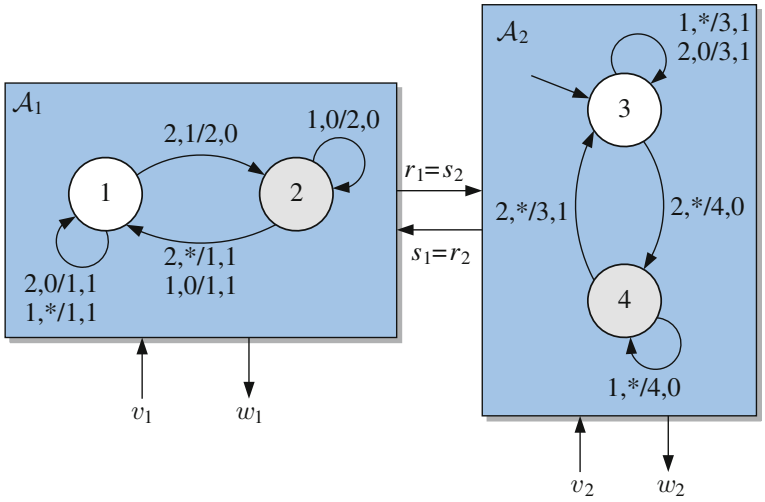


Fig. 12.10 Composite system model

zones each of which corresponds to a state of the automata. For the passing place, the cabin 2 goes over the rail corresponding to state 6 whereas cabin 1 uses the rail for state 3.

As usual, the cabins cannot move independently, but are coupled through a rope. This coupling should be represented by using the coupling inputs s_i and coupling output r_i of the cabin models together with an interconnection model. Extend the given subsystem models to get a model of the railway as an I/O automata network. Build a deterministic automaton showing the overall system and show that the cabins cannot simultaneously assume the same state (rail zone). □

Exercise 12.2 Diagnosis of an interconnected discrete-event system
Consider the system modelled by the I/O automata network shown in Fig. 12.10. The system is designed so that the two subsystems represented by the automata $\mathcal{A}_1$ and $\mathcal{A}_2$ cannot reach the states 2 and 4 simultaneously.

1. Combine the models shown in the figure to get a deterministic automaton of the overall system. Show that the specification mentioned above is satisfied.
2. The fault f_2 appearing in the automaton $\mathcal{A}_2$ makes the coupling output r_2 identical to 1 ($r_2(k) = 1$) independently of the automaton state. Change the model $\mathcal{A}_2$ accordingly. What happens in the model of the overall system?
3. Use decentralised diagnosers to detect the fault f_2. Is it possible to select an input sequence for both subsystems such that the fault is detected before the subsystems reach simultaneously the states 2 and 4?
4. Consider now a sensor fault, which makes the output of the automaton $\mathcal{A}_1$ constant: $w_1(k) = 1$. Can decentralised diagnosers detect this fault? Select, if possible, distinguishing input sequences for both subsystems. □

12.7 Bibliographical Notes

There are different lines of research for the diagnosis of composite systems. References [75, 236, 237, 348, 349] propose *distributed* diagnostic schemes where the local units send their results to some global diagnostic unit (coordinator) that merges the local results to get the best possible result for the overall system. Similarly, the method described in [123] for systems described by Petri nets relies on an exchange of information about the occurrence of observable events, particularly about events over which the components interact. In [41] a distributed diagnostic method is proposed with two diagnostic units where one of the units communicates information about its event sequence to the other one. According to the communication protocol, information is exchanged only if the message transferred can be expected to improve the diagnostic result. All these schemes aim at recovering the best possible result, which would be obtained by a centralised diagnostic unit with access to the overall model and all measurement information. Reference [61] describes an approach where several diagnostic units work in parallel and combine their local decisions to get the global diagnostic result.

On the other hand, *decentralised* diagnosis deals with the situation that there are no information links among the diagnostic units of the subsystems. References [235, 381] propose decentralised diagnostic methods and investigate the relation of the results obtained by decentralised or centralised diagnostic approaches. Only in special cases, both results coincide. Reference [281] introduces the notion of *co-diagnosability* as the property that every fault is detected by at least one decentralised diagnostic unit after a finite event sequence. This notion has been extended in [283] to safe co-diagnosability, which describes the property that if a system is faulty, there exists at least on decentralised diagnoser that detects the fault within a bounded delay and before safety specifications are violated. The conditions derived show that these properties exist only in very restrictive classes of systems.

References [269, 270] developed a hierarchical structure to answer the question, which faults can be detected by the higher level diagnostic unit.

The majority of papers deal with automata networks where the couplings among the components are described by the simultaneous appearance of some events in all components. The model used in this chapter consists of I/O automata for all subsystems, that are coupled in the generalised version introduced in [84, 214], where the automata are coupled through interconnection signals. Hence, the class of systems considered here is similar to *active systems* investigated in [13], although the subsystem interactions are instantaneous here. Asynchronous state transitions are possible in the modelling approach used here, whereas [235, 236, 237] refer to networks, in which the components carry out synchronous state transitions.

The extension towards interconnected stochastic discrete-event systems has been made in, for example, [200], where a decentralised scheme of diagnosers is considered and the notion of co-diagnosability is extended to the stochastic setting.

Appendix A
Some Prerequisites on Vectors and Matrices

Homogeneous system. Let $\boldsymbol{A}$ be an $m \times n$ matrix, $\boldsymbol{A} \in |\mathcal{R}^{m\times n}$ and $\boldsymbol{x}$ an n vector $\boldsymbol{x} \in |\mathcal{R}^n$. The system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{O}$$

is called a *homogeneous system*.

- Every homogeneous system has the solution $\boldsymbol{x} = \boldsymbol{0}$. This solution is called the trivial solution.
- If the homogeneous system has fewer equations than unknowns, it has an infinite number of solutions; in particular, it has a nontrivial solution.

Vector space. A collection V of n-vectors is a *vector space* if V satisfies the following properties:

1. (*Closure under vector addition*) The sum of two vectors u and v in V is a vector $u + v$ that belongs to V.
2. (*Closure under scalar multiplication*) The product of a vector v in V with any scalar c is a vector cv that belongs to V.

If V satisfies these properties, it is said to be *closed* under vector addition and scalar multiplication.

Rank of a matrix. Let $\boldsymbol{A}$ be an $m \times n$ matrix and let $\tilde{\boldsymbol{A}}$ be its Gauss-reduced form. The rank of $\boldsymbol{A}$ is the number of non-zero rows of $\tilde{\boldsymbol{A}}$. It is equal to the number of linearly independent rows or columns in matrix $\boldsymbol{A}$.

Subspace. A vector space V is a *subspace* of a vector space W if every vector in V also belongs to W.

Let $\boldsymbol{A} \in |\mathcal{R}^{m\times n}$ be an $m \times n$ matrix.

Definition. *The subspace of* $|\mathcal{R}^n$ *consisting of all solutions to the homogeneous linear system* $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ *is called the null space of the matrix* $\boldsymbol{A}$, *and is denoted by* $N(\boldsymbol{A})$.

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8

Definition. *The subspace of* $|\mathcal{R}^m$ *consisting of all vectors* $\boldsymbol{b}$ *for which the system* $\boldsymbol{Ax} = \boldsymbol{b}$ *is consistent is called the column space of the matrix* $\boldsymbol{A}$, *and is denoted by* $C(\boldsymbol{A})$.

Example Column space
Let

$$\boldsymbol{A} = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 6 & 9 \end{pmatrix}.$$

The null space of $\boldsymbol{A}$ is the subspace of $|\mathcal{R}^3$ consisting of all vectors of the form

$$\boldsymbol{x} = t \begin{pmatrix} 2 \\ -1 \\ 0 \end{pmatrix}^{\mathrm{T}} + u \begin{pmatrix} 3 \\ 0 \\ -1 \end{pmatrix}'$$

for real numbers t and u. The column space of $\boldsymbol{A}$ is the subspace of $|\mathcal{R}^2$ consisting of all vectors of the form

$$\boldsymbol{b} = v \begin{pmatrix} 1 \\ 3 \end{pmatrix}^{\mathrm{T}}$$

for real numbers v. □

Four fundamental subspaces. Let $\boldsymbol{A}$ be an $m \times n$ matrix. The four fundamental subspaces of $\boldsymbol{A}$ are

- the *column space* of $\boldsymbol{A}$, denoted by $C(\boldsymbol{A})$.
- the *nullspace* of $\boldsymbol{A}$, denoted by $N(\boldsymbol{A})$.
- the *row space* of $\boldsymbol{A}$, which is the column space of $\boldsymbol{A}^{\mathrm{T}}$ and denoted by $R(\boldsymbol{A})$.
- the *left nullspace* of $\boldsymbol{A}$, which is the nullspace of $\boldsymbol{A}^{\mathrm{T}}$. It contains all vectors $\boldsymbol{y}$ such that $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y} = 0$, and is denoted by $N(\boldsymbol{A}')$.

Fundamental theorem of linear algebra. Let $\boldsymbol{A}$ be an $m \times n$ matrix of rank r. Then

- The column space of $\boldsymbol{A}$ has dimension r.
- The nullspace of $\boldsymbol{A}$ has dimension $n - r$.
- The rowspace of $\boldsymbol{A}$ has dimension r.
- The left nullspace of $\boldsymbol{A}$ has dimension $m - r$.

Example Four fundamental subspaces
Let

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 6 & 9 \end{bmatrix}.$$

- The (right) null space of $\boldsymbol{A}$ has a basis $\left\{ \begin{pmatrix} 2 \\ -1 \\ 0 \end{pmatrix}^{\mathrm{T}}, \begin{pmatrix} 3 \\ 0 \\ -1 \end{pmatrix}' \right\}$. It is a subspace of $|\mathcal{R}^3$ of dimension 2.
- The column space of $\boldsymbol{A}$ has a basis $\left\{ \begin{pmatrix} 1 \\ 3 \end{pmatrix}^{\mathrm{T}} \right\}$. It is a subspace of $|\mathcal{R}^2$ of dimension 1.
- The row space of $\boldsymbol{A}$ has a basis $\left\{ \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}^{\mathrm{T}} \right\}$. It is a subspace of $|\mathcal{R}^3$ of dimension 1.
- The (left) null space of $\boldsymbol{A}$ is the set of solutions to the matrix equation $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A} = \boldsymbol{0}'$. It has a basis $\left\{ \begin{pmatrix} -3 \\ 1 \end{pmatrix}' \right\}$ and is a subspace of $|\mathcal{R}^2$ of dimension 1. □

Orthogonal subspaces. Two subspaces V and W of the vector space $|\mathcal{R}^n$ are *orthogonal* if every vector $\boldsymbol{v} \in V$ is orthogonal to every vector $\boldsymbol{w} \in W$; that is, $\boldsymbol{v}^{\mathrm{T}}\boldsymbol{w} = 0$ for all $\boldsymbol{v}$ and $\boldsymbol{w}$.

Orthogonal complement. Given a subspace V of $|\mathcal{R}^n$, the space of all vectors orthogonal to V is called the *orthogonal complement* of V, and is denoted $V^{\perp}$.

Jacobian and other derivatives. Derivatives of functions with respect to vectors are employed when nonlinear systems are linearised. In the dynamical vector equation

$$\dot{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}), \quad \boldsymbol{x} \in |\mathcal{R}^n, \boldsymbol{u} \in |\mathcal{R}^m \tag{A.1}$$

we introduce $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \bar{\boldsymbol{x}}$ and $\tilde{\boldsymbol{u}} = \boldsymbol{u} - \bar{\boldsymbol{u}}$ where the overbar indicates the value (functional) about which the system is to be to linearised. Taylor expansion of Eq. (A.1) yields

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \frac{\mathrm{d}\bar{\boldsymbol{x}}}{\mathrm{d}t} + \frac{\mathrm{d}\tilde{\boldsymbol{x}}}{\mathrm{d}t} = \boldsymbol{g}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}) + \frac{\partial \boldsymbol{g}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})}{\partial \boldsymbol{x}}\tilde{\boldsymbol{x}} + \frac{\partial \boldsymbol{g}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})}{\partial \boldsymbol{u}}\tilde{\boldsymbol{u}} + \cdots$$

and the first-order expansion defines the linearised system,

$$\frac{\mathrm{d}\tilde{\boldsymbol{x}}}{\mathrm{d}t} = \frac{\partial \boldsymbol{g}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})}{\partial \boldsymbol{x}}\tilde{\boldsymbol{x}} + \frac{\partial \boldsymbol{g}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})}{\partial \boldsymbol{u}}\tilde{\boldsymbol{u}} = \boldsymbol{A}\tilde{\boldsymbol{x}} + \boldsymbol{B}\tilde{\boldsymbol{u}}.$$

The *Jacobian* $\frac{\partial \boldsymbol{g}(\bar{\boldsymbol{x}},\bar{\boldsymbol{u}})}{\partial \boldsymbol{u}}$ is the $n \times m$ matrix

$$\frac{\partial \boldsymbol{g}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})}{\partial \boldsymbol{u}} = \begin{pmatrix} \frac{\partial \boldsymbol{g}_1}{\partial u_1} & \frac{\partial \boldsymbol{g}_1}{\partial u_2} & \cdots & \frac{\partial \boldsymbol{g}_1}{\partial u_m} \\ \frac{\partial \boldsymbol{g}_2}{\partial u_1} & \frac{\partial \boldsymbol{g}_2}{\partial u_2} & \cdots & \frac{\partial \boldsymbol{g}_2}{\partial u_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{g}_n}{\partial u_1} & \frac{\partial \boldsymbol{g}_n}{\partial u_2} & \cdots & \frac{\partial \boldsymbol{g}_n}{\partial u_m} \end{pmatrix} \tag{A.2}$$

of partial derivatives of the entries of g.

Other useful derivatives with respect to a vector v are

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}v}(\boldsymbol{A}\boldsymbol{v}) &= \boldsymbol{A} \\ \frac{\mathrm{d}}{\mathrm{d}v}\left(\boldsymbol{v}^{\mathrm{T}}\boldsymbol{A}\right) &= \boldsymbol{A}^{\mathrm{T}} \\ \frac{\mathrm{d}}{\mathrm{d}v}\left(\boldsymbol{v}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{v}\right) &= \boldsymbol{v}^{\mathrm{T}}\left(\boldsymbol{A} + \boldsymbol{A}^{\mathrm{T}}\right). \end{aligned}$$

Appendix B
Notions of Probability Theory

Introduction. In this appendix, different notions of probability theory that are used or mentioned in Chap. 6 are briefly reviewed. Gaussian and χ^2-distributed random variables are first presented. Next a simple hypothesis testing problem is stated and solved by the so-called χ^2 test. The statistics of the empirical mean is analysed in the subsequent section. Finally, the last part of the appendix, which is also the main one, is presenting a review of notions on continuous- and discrete-time random processes.

Gaussian distributed random variables. A normally distributed random variable x with mean μ and variance σ^2 is characterised by its probability density function, $p(x)$ which has the form

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{B.1}$$

To indicate that the probability law of x, $\mathcal{L}(x)$, is the Gaussian law with mean μ and variance σ^2, one uses the notation $\mathcal{L}(x) = \mathcal{N}(\mu, \sigma^2)$.

The probability density function of a n-dimensional Gaussian random vector $\boldsymbol{x}$ with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{Q}$ has the form

$$p(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^n \det \boldsymbol{Q}}} \exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{Q}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})}{2}\right) \tag{B.2}$$

Remark (*Notations*) The symbol x ($\boldsymbol{x}$) has two different meanings above. It denotes either a random variable (vector) or the argument of the probability density function

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8

associated to the considered random variable (vector). The distinction should be clear from the context. □

χ^2 test. We often encounter that a sum of squares of Gaussian variables are calculated and used as a test statistics. This is the case for the GLR test for a Gaussian sequence, see Remark 7.2.4. This generic test is formulated as follows.

Problem (χ *hypothesis testing*) Given $\boldsymbol{x}_{\mathrm{m}}$, a realisation of a normally distributed random vector $\boldsymbol{x}$ of dimension n, with variance $\boldsymbol{Q}$, determine whether $\boldsymbol{x}_{\mathrm{m}}$ is a realisation of a random vector $\boldsymbol{x}$ with zero mean or non-zero mean, namely choose between the following two hypotheses:

- $\mathcal{H}_0$: $\mathcal{L}(\boldsymbol{x}) = \mathcal{N}(0,\, \boldsymbol{Q})$
- $\mathcal{H}_1$: $\mathcal{L}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu},\, \boldsymbol{Q})$ where $\boldsymbol{\mu}$ is an unknown non-zero vector

The index "m" used above stands for "measured".

To solve the problem, it suffices to realise that the random variable $\xi = \boldsymbol{x}^{\mathrm{T}} \boldsymbol{Q}^{-1} \boldsymbol{x}$ has a χ^2 distribution under hypothesis $\mathcal{H}_0$. Since $\boldsymbol{Q}$ is positive definite, it can be factorised as $\boldsymbol{Q} = \tilde{\boldsymbol{Q}} \tilde{\boldsymbol{Q}}^{\mathrm{T}}$ by Choleski factorisation. Set $\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{Q}}^{-1} \boldsymbol{x}$, and consider the variance of $\tilde{\boldsymbol{x}}$ under hypothesis $\mathcal{H}_0$:

$$\mathrm{E}\,(\tilde{\boldsymbol{x}} \tilde{\boldsymbol{x}}^{\mathrm{T}}) = \tilde{\boldsymbol{Q}}^{-1} \mathrm{E}\,(\boldsymbol{x} \boldsymbol{x}') \tilde{\boldsymbol{Q}}^{-T} = \tilde{\boldsymbol{Q}}^{-1} \tilde{\boldsymbol{Q}} \tilde{\boldsymbol{Q}}^{\mathrm{T}} \tilde{\boldsymbol{Q}}^{-T} = \boldsymbol{I}_n,$$

where the superscript $-T$ stands for inverse transpose. As the variance of $\tilde{x}$ is the identity matrix, its components, $\tilde{\boldsymbol{x}}_i, i = 1, \ldots, n$ are independent and $\mathcal{L}(\tilde{\boldsymbol{x}}_i) = \mathcal{N}(0, 1)$ under hypothesis $\mathcal{H}_0$. Finally, noticing that

$$\xi = \boldsymbol{x}^{\mathrm{T}} \boldsymbol{Q}^{-1} \boldsymbol{x} = \tilde{\boldsymbol{x}}^{\mathrm{T}} \tilde{\boldsymbol{x}} = \sum_{i=1}^{n_z} \tilde{x}_i^2,$$

one deduces that ξ has a $\chi^2(n)$ distribution under hypothesis $\mathcal{H}_0$.

Thus a standard χ^2-test can be used to check whether $\xi_{\mathrm{m}} = \boldsymbol{x}_{\mathrm{m}}^{\mathrm{T}} \boldsymbol{Q}^{-1} \boldsymbol{x}_{\mathrm{m}}$ is a realisation of a $\chi^2(n)$ random variable. This test relies on the χ^2 statistical table that provides, for a given probability of false alarm, α (namely the probability of choosing $\mathcal{H}_1$ while $\mathcal{H}_0$ is true), a threshold h such that:

$$\mathrm{Prob}(\xi \leq h) = 1 - \alpha, \tag{B.3}$$

where $\mathrm{Prob}(\xi \leq h)$ denotes the probability of the event $\xi \leq h$.

The χ^2-test then simply amounts to the following operations:

Algorithm B.4 *χ^2-Test*

Given: $\boldsymbol{x}_{\mathrm{m}}$, a realisation of the n-dimensional normally distributed random vector $\boldsymbol{x}$ with variance $\boldsymbol{Q}$,

α, a probability of false alarm.

Determine:

1. The threshold h that fulfils (B.3) from the χ^2 statistical table.
2. $\xi_{\mathrm{m}} = \boldsymbol{x}_{\mathrm{m}}^{\mathrm{T}} \boldsymbol{Q}^{-1} \boldsymbol{x}_{\mathrm{m}}$.

Output: Accept $\mathcal{H}_0$ if $\xi_{\mathrm{m}} \leq h$.
Accept $\mathcal{H}_1$ otherwise.

Statistics of the empirical mean. Consider a data sample $\{x_{\mathrm{m},1}, \ldots, x_{\mathrm{m},n}\}$ where the $x_{\mathrm{m},i}$, $(i = 1, \ldots, n)$ are realisations of the random variables x_i. The latter are assumed to be independent and identically distributed (IID). Let $m = \mathrm{E}(x_i)$ and $\sigma^2 = \mathrm{E}((x_i - m)^2)$ be respectively the mean and the variance of x_i ($\mathrm{E}(.)$ denotes the expectation of the considered random variable). The empirical mean of the data sample is defined as

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i .$$

The mean and the variance of $\bar{x}$ are given by

$$\begin{aligned} \mathrm{E}(\bar{x}) &= m \\ \mathrm{E}((\bar{x} - m)^2) &= \frac{\sigma^2}{n}. \end{aligned}$$

In the sequel, the particular case where the distribution of the x_i is Gaussian is considered, namely $\mathcal{L}(x_i) = \mathcal{N}(m, \sigma^2)$. This implies $\mathcal{L}(\bar{x}) = \mathcal{N}(m, \frac{\sigma^2}{n})$.

To be able to characterise the quality of the estimate of the mean provided by the empirical mean one resorts to the following result. Define $T_{n-1} = \frac{\bar{x}-m}{q}\sqrt{n-1}$, where $q^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$ is the empirical variance of the sample. Then it can be proven that T_{n-1} is a Student random variable with $n-1$ degrees of freedom. From this fact, the confidence region for the mean can be characterised by

$$\bar{x}_{\mathrm{m}} - t_{\alpha/2} \frac{q_{\mathrm{m}}}{\sqrt{n-1}} < m < \bar{x}_{\mathrm{m}} + t_{\alpha/2} \frac{q_{\mathrm{m}}}{\sqrt{n-1}}, \tag{B.4}$$

where

- $\bar{x}_{\mathrm{m}}$ and q_{m} are realisations of the random variables $\bar{x}$ and q respectively, namely $\bar{x}_{\mathrm{m}} = \frac{1}{n}\sum_{i=1}^{n} x_{m,i}$ and $q_{\mathrm{m}}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_{m,i} - \bar{x}_{\mathrm{m}})^2$
- $t_{\alpha/2}$ is such that $\int_{-t_{\alpha/2}}^{t_{\alpha/2}} f_T(x)\mathrm{d}x = \alpha$ with
  - $f_T(x)$: the probability density function associated to a Student random variable with $n-1$ degrees of freedom
  - α: the probability that m lies in the considered interval (B.4)

Remark (*Notation conventions*) A different notation has been used above to denote a random variable or vector and a specific realisation of this variable or vector. In most of the other sections, this distinction is not indicated explicitly; the considered object should be clear from the context. □

Stochastic processes. A stochastic (or random) process is defined as a mapping that associates with each time instant t in a set $\mathcal{T}$ a random variable $x(t)$ (or a random vector in the case of a vector random process). A random process thus appears as an infinite set of random variables. One can distinguish continuous- and discrete-time stochastic processes according as the variable t belongs to a continuum of values or the set $\mathcal{T}$ is made of the sampling instants. In the latter case, $\mathcal{T} = \{\ldots, -T, 0, T, 2T, \ldots\}$ or $\mathcal{T} = \{\ldots, -1, 0, 1, 2, \ldots\}$ when the sampling period T is chosen as the time unit. Discrete-time stochastic processes are also called random on stochastic sequences.

The notion of random process is used to model quantities for which there is no way to predict an exact value at a future instant of time. This is typically the case for measurement noise for instance. The outcome of an experiment generates a specific outcome or realisation of a random process, which is a function of time.

Distribution function and probability density function. A random process is completely characterised by its nth-order distribution function (also called cumulative distribution function), for an arbitrary n, defined as

$$F(x_1, t_1; \ldots; x_n, t_n) = \mathrm{Prob}(x(t_1) \leq x_1, \ldots, x(t_n) \leq x_n),$$
$$t_i \neq t_j,\ x_1, \ldots, x_n \in \mathbb{R}, \quad t_1, \ldots, t_n \in \mathbb{R} \text{ or } Z,$$

where

$$\mathrm{Prob}(x(t_1) \leq x_1, \ldots, x(t_n) \leq x_n)$$

is the probability that the events $(x(t_1) \leq x_1), \ldots, (x(t_n) \leq x_n)$ are observed in a realisation of $x(t)$.

In particular, the first-order (cumulative) distribution function is defined as $F(x,t) = \mathrm{Prob}(x(t) \leq x)$. It is thus the probability of the event $(x(t) \leq x)$.

The probability density function of the random process $x(t)$ is defined as the derivative of the first-order cumulative distribution function with respect to x: $p(x,t) = \frac{\partial F(x,t)}{\partial x}$ and the nth order probability density function is:

$$p(x_1, t_1; \cdots ; x_n, t_n) = \frac{\partial^n F(x_1, t_1; \cdots ; x_n, t_n)}{\partial x_1 \cdots \partial x_n}$$

Moments of stochastic processes. The properties of a random process $x(t)$ can be fully characterised by its moments defined, for the nth-order moment as

$$M_x^n(t) = \int_{-\infty}^{\infty} x^n p(x, t) \mathrm{d}x.$$

The first-order moment of a random process $x(t)$ is called its mean or its expected value,

$$\mu_x(t) = \int_{-\infty}^{\infty} x p(x, t) \mathrm{d}x.$$

This operation is often denoted $\mu_x(t) = \mathrm{E}\,(x(t))$ where "E" stands for the expectation operator. In the vector case, $\boldsymbol{\mu}_x(t)$ is a vector with the same dimension as $\boldsymbol{x}(t)$ of which the ith entry is obtained by computing the above integral for the ith component of $\boldsymbol{x}(t)$. In that case, $p(\boldsymbol{x}, t)$ is a scalar function of vector $\boldsymbol{x}$ and time t (see for instance the probability density function of a Gaussian random process in (B.6) below).

Similarly, the mean square is defined as the second-order moment

$$ms_x(t) = \int_{-\infty}^{\infty} x^2 p(x, t) \mathrm{d}x = \mathrm{E}\,(x^2(t)).$$

The variance of the stochastic process $x(t)$ is the centred second order moment of $x(t)$ ("centred" indicates that the mean is subtracted from $x(t)$),

$$\sigma_x^2(t) = \int_{-\infty}^{\infty} (x - \mu_x(t))^2 p(x, t) \mathrm{d}x = \mathrm{E}\,((x(t) - \mu_x(t))^2).$$

For a vector stochastic process, this expression takes the form

$$\begin{aligned} \boldsymbol{Q}_x(t) &= \int_{-\infty}^{\infty} (\boldsymbol{x}(t) - \boldsymbol{\mu}_x(t))(\boldsymbol{x} - \boldsymbol{\mu}_x(t))^{\mathrm{T}} p(\boldsymbol{x}, t)\, \mathrm{d}\boldsymbol{x} \\ &= \mathrm{E}\,((\boldsymbol{x}(t) - \boldsymbol{\mu}_x(t))(\boldsymbol{x}(t) - \boldsymbol{\mu}_x(t))^{\mathrm{T}}). \end{aligned}$$

$\boldsymbol{Q}_x(t)$ is thus a matrix of which entry (i, j) is given by

$$\int_{-\infty}^{\infty} (x_i - \mu_{x,i})(x_j - \mu_{x,j}) p(\boldsymbol{x}, t)\, \mathrm{d}\boldsymbol{x}.$$

The integral is actually a multiple integral; integration is performed over each component of $\boldsymbol{x}$.

The autocorrelation function of a random process $\boldsymbol{x}(t)$ is the joint moment of the random vectors $\boldsymbol{x}(t_1)$ and $\boldsymbol{x}(t_2)$,

$$\boldsymbol{R}_{xx}(t_1, t_2) = \mathrm{E}\,(\boldsymbol{x}(t_1)\boldsymbol{x}(t_2)^{\mathrm{T}}) = \int_{-\infty}^{\infty} \boldsymbol{x}_1 \boldsymbol{x}_2^{\mathrm{T}}\, p(\boldsymbol{x}_1, t_1; \boldsymbol{x}_2, t_2)\, \mathrm{d}\boldsymbol{x}_1\, \mathrm{d}\boldsymbol{x}_2$$

and the (auto)covariance of $\boldsymbol{x}(t)$ is given as

$$\boldsymbol{C}_{xx}(t_1, t_2) = \mathrm{E}\,((\boldsymbol{x}(t_1) - \boldsymbol{\mu}(t_1))(\boldsymbol{x}(t_2) - \boldsymbol{\mu}(t_2))^{\mathrm{T}}).$$

This generalises to the cross-covariance of two random processes $\boldsymbol{x}(t)$ and $\boldsymbol{y}(t)$ defined as

$$\boldsymbol{C}_{xy}(t_1, t_2) = \mathrm{E}\,((\boldsymbol{x}(t_1) - \boldsymbol{\mu}_x(t_1))(\boldsymbol{y}(t_2) - \boldsymbol{\mu}_y(t_2))).$$

Most often in engineering applications, one resorts to the first- and second-order moments of the stochastic processes; however, these are not sufficient to fully characterise a random process, except if this process is normally distributed. In the latter case, $p(x, t)$ corresponds to the well-known Gaussian distribution with mean μ_x and variance σ_x^2 (or $\boldsymbol{Q}_x$ in the vector case),

$$p(x, t) = \frac{1}{\sigma_x(t)\sqrt{2\pi}} \exp\left(-\frac{(x - \mu_x(t))^2}{2\sigma_x^2(t)}\right), \tag{B.5}$$

and for an n-dimensional Gaussian random process

$$p(\boldsymbol{x}, t) = \frac{1}{\sqrt{(2\pi)^n \det \boldsymbol{Q}_x(t)}} \exp\left(-\frac{(\boldsymbol{x} - \boldsymbol{\mu}_x(t))^{\mathrm{T}}\, \boldsymbol{Q}_x(t)^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_x(t))}{2}\right) \tag{B.6}$$

Remark (*Discrete-time random process*) The above definitions are valid for both continuous-time and discrete-time stochastic processes. Below, when specific expressions must be considered for discrete-time processes, the superscript "d" will be used to indicate explicitly that moments or other quantities are associated to discrete-time random processes. □

Stationary random process. A stochastic process is said to be stationary (in the strict sense) if its probability density functions of any order are not affected by a shift in the time origin, namely

$$p(x_1, t_1; x_2, t_2; \ldots; x_n, t_n) = p(x_1, t_1 + \epsilon; \ldots; x_n, t_n + \epsilon)$$

for any real ϵ and any integer n.

On the other hand, a weakly stationary random process has the following properties:

- its mean does not change with time.
- its covariance $C(t_1, t_2)$ depends on t_1 and t_2 only through the difference $t_2 - t_1$. Thus $C(t_1, t_2) = C(t_2 - t_1) = \mathrm{E}\,((x(t + t_2 - t_1) - \mu_x)(x(t) - \mu_x))$.

A random process with the above two properties is also sometimes said to be stationary in the wide sense.

Stationarity in the strict sense implies weak stationarity, but the converse is not true, except if $x(t)$ has a Gaussian distribution.

Clearly, the probability density function of a Gaussian weakly stationary random process is given by (B.5) or (B.6) in which the argument t is cancelled.

Empirical mean, variance and covariance of a stochastic process. Often in practice, only one or a few outcomes of an experiment are available, and one has to infer from such data a model of the observed phenomenon. In particular, one has to deduce the properties of measurement noise from a realisation of this noise. A classical approach to do this is to estimate the moments of the corresponding random process by time averages. This amounts to assuming that time averages are equal to ensemble averages (i.e. expected values); it is the *ergodic hypothesis*. Roughly speaking, this hypothesis holds true for a stochastic process $x(t)$ if, as τ increases, the random variables (or vectors) $x(t)$ and $x(t+\tau)$ become uncorrelated. This situation is quite general in practice.

For a weakly stationary ergodic continuous-time stochastic process, the following equalities hold

$$\begin{aligned}
\boldsymbol{\mu}_x &= \mathrm{E}\,(\boldsymbol{x}(t)) = \lim_{T\to\infty} \frac{1}{2T}\int_{-T}^{T} \boldsymbol{x}(t)\mathrm{d}t \\
\boldsymbol{R}_{xx}(\tau) &= \mathrm{E}\,(\boldsymbol{x}(t+\tau)\boldsymbol{x}^{\mathrm{T}}(t)) = \lim_{T\to\infty} \frac{1}{2T}\int_{-T}^{T} \boldsymbol{x}(t+\tau)\boldsymbol{x}^{\mathrm{T}}(t)\mathrm{d}t \\
\boldsymbol{C}_{xx}(\tau) &= \mathrm{E}\,((\boldsymbol{x}(t+\tau)-\boldsymbol{\mu}_x)(\boldsymbol{x}(t)-\boldsymbol{\mu}_x)^{\mathrm{T}}) \\
&= \lim_{T\to\infty} \frac{1}{2T}\int_{-T}^{T} (\boldsymbol{x}(t+\tau)-\boldsymbol{\mu}_x)(\boldsymbol{x}(t)-\boldsymbol{\mu}_x)^{\mathrm{T}}\mathrm{d}t.
\end{aligned}$$

In practice, only a finite number, say N, of samples of a realisation of a stochastic process is available. The integral in the above expressions can be estimated from the data set $\{\boldsymbol{x}(1),\ldots,\boldsymbol{x}(N)\}$ as

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}(i)$$

$$\hat{Q} = \hat{C}(0) = \frac{1}{N}\sum_{i=1}^{N} (\boldsymbol{x}(i)-\hat{\boldsymbol{\mu}})(\boldsymbol{x}(i)-\hat{\boldsymbol{\mu}})^{\mathrm{T}}.$$

Spectral density or power spectrum. In classical control engineering, transfer functions are extremely useful computational and analysis tools. To exploit them in stochastic control theory, the notion of power spectrum (or spectral density) is introduced. By definition, the power spectrum of the continuous-time (discrete-time)

stochastic process $x(t)$ is the Fourier transform (discrete Fourier transform) of its autocovariance function,

$$S_{xx}(\omega) = \int_{-\infty}^{\infty} C_{xx}(\tau) \exp(-\mathrm{j}\omega\tau)\mathrm{d}\tau \tag{B.7}$$

($S^d_{xx}(\omega) = \sum_{n=-\infty}^{\infty} C^d_{xx}(n) \exp(-\mathrm{j}\omega n)$). For a random vector, $\boldsymbol{S}_{xx}(\omega)$ is a matrix of which each entry is the Fourier transform of the corresponding entry in $\boldsymbol{C}_{xx}$.

The name "spectral density" used for $S_{xx}(\omega)$ is due to the fact that the variance of the stochastic process can be recovered through

$$\sigma_x^2 = \frac{1}{2\pi} \int_{-\infty}^{\infty} S_{xx}(\omega)\, \mathrm{d}\omega \tag{B.8}$$

($\sigma_x^{d2} = \frac{1}{2\pi} \int_{-\pi}^{\pi} S^d_{xx}(\omega)\, \mathrm{d}\omega$). Equation (B.8) is obtained by taking the inverse Fourier transform of (B.7) and setting the time shift equal to zero in the result.

Example Computation of the spectral density function
The autocovariance function of a random process $v(t)$ is given by

$$C_{vv}(\tau) = \sigma^2 \mathrm{e}^{-\beta|\tau|},$$

where σ^2 and β are known variables. The spectral density function for the $v(t)$ process is

$$S_{vv}(\omega) = \frac{2\sigma^2\beta}{\omega^2 + \beta^2}.$$

Evaluation of (B.8) yields

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{2\sigma^2\beta}{\omega^2 + \beta^2}\, \mathrm{d}\omega = \frac{\sigma^2\beta}{\pi} \left(\frac{1}{\beta} \tan^{-1}\left(\frac{\omega}{\beta}\right) \right)_{-\infty}^{\infty} = \sigma^2 \tag{B.9}$$

as expected since $C_{vv}(0) = \sigma^2$. □

Example Band-limited noise through low-pass filter
Given a stable low-pass filter with the transfer function

$$H(s) = \frac{\alpha}{s + \alpha} \tag{B.10}$$

with input $w(t)$, which is a band-limited random signal with autocorrelation function

$$R_{ww}(\tau) = \sigma_w^2 \mathrm{e}^{-\beta|\tau|}.$$

The output of the filter is $y(t)$ whose covariance σ_y^2 should be determined. Stability implies $\alpha > 0$ and $\beta > 0$

Applying Eq. (B.8)

$$\begin{aligned}
&\sigma_y^2 = \frac{1}{2\pi}\int_{-\infty}^{\infty} \sigma_w^2 \frac{2\beta}{\omega^2+\beta^2}\frac{\alpha^2}{\omega^2+\alpha^2}\,\mathrm{d}\omega \;\Leftrightarrow\\
&\sigma_y^2 = \frac{\sigma_w^2 2\alpha^2}{(\beta^2-\alpha^2)}\frac{1}{2\pi}\left(\int_{-\infty}^{\infty}\frac{\beta}{(\omega^2+\alpha^2)}\,\mathrm{d}\omega - \int_{-\infty}^{\infty}\frac{\beta}{(\omega^2+\beta^2)}\,\mathrm{d}\omega\right) \;\Leftrightarrow\\
&\sigma_y^2 = \frac{2}{\pi}\alpha^2\frac{\sigma_w^2}{\beta^2-\alpha^2}\left(\frac{1}{2}\pi\frac{\beta}{\alpha}-\frac{1}{2}\pi\right) \;\Leftrightarrow\\
&\sigma_y^2 = \sigma_w^2\frac{(\beta-\alpha)\,\alpha}{\beta^2-\alpha^2} \;\Rightarrow\; \sigma_y^2 = \frac{\alpha}{\alpha+\beta}\,\sigma_w^2.\;\square
\end{aligned} \tag{B.11}$$

Continuous and discrete-time white noise processes. A scalar continuous-time white noise process $x(t)$ is a weakly stationary continuous-time random process with autocovariance function

$$C_{xx}(\tau) = \mathrm{E}\left((x(t+\tau)-\mu)(x(t)-\mu)\right) = s\delta(\tau), \tag{B.12}$$

where s is a real constant called intensity and $\delta(\tau)$ is the Dirac impulse. For a vector white noise process, (B.12) takes the form

$$\boldsymbol{C}_{xx}(\tau) = \mathrm{E}\left((\boldsymbol{x}(t+\tau)-\boldsymbol{\mu})(\boldsymbol{x}(t)-\boldsymbol{\mu})^{\mathrm{T}}\right) = \boldsymbol{S}\delta(\tau), \tag{B.13}$$

where the intensity $\boldsymbol{S}$ is a semi-positive definite matrix.

For a white noise process, the spectral density function is constant:

$$S_{xx}(\omega) = s \quad \text{or} \quad \boldsymbol{S}_{xx}(\omega) = \boldsymbol{S}.$$

A white noise process is clearly an abstraction, as it corresponds to a process with infinite variance. It can be seen as the limiting case when β tends to infinity in the given example, which means that no correlation exists between successive time instants. The term "white noise" comes from the analogy with the spectral properties of white light; all frequencies are present with the same intensity in the signal.

A white noise sequence is defined in a similar way as its continuous-time counterpart. It is a weakly stationary discrete-time process with autocovariance given by

$$C_{xx}^d(\tau) = \begin{cases} \sigma^2 & \tau = 0 \\ 0 & \tau = \pm 1, \pm 2, \ldots \end{cases} \tag{B.14}$$

in the scalar case. The associated spectral density writes $S_{xx}^d(\omega) = \sigma^2$.

A band-limited white noise process is a random process whose spectral density is constant over a finite range of frequencies, and zero outside this range. A noise source that has a constant spectral density down to zero frequency, and a bandwidth

of B [rad/s] is defined by

$$S_{vv}(\omega) = \begin{cases} \alpha & |\omega| \leq B \\ 0 & |\omega| > B. \end{cases} \tag{B.15}$$

Such a stochastic process can be approximated as the output of a linear dynamical system with a white noise input and with cut-off frequency $B/2\pi$ [s^{-1}]. This is the subject of the next section.

Filtered weakly stationary process and filtered white noise process. Consider a linear time-invariant system described by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{w}(t) \tag{B.16}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{v}(t), \tag{B.17}$$

where $\boldsymbol{w}(t)$ and $\boldsymbol{v}(t)$ are mutually uncorrelated weakly stationary random processes with mean and power spectral density $\boldsymbol{m}_w$, $\boldsymbol{S}_w(\omega)$ and $\boldsymbol{m}_v$, $\boldsymbol{S}_v(\omega)$ respectively.

Our aim is first to compute the power spectral density of $\boldsymbol{x}(t)$ and $\boldsymbol{y}(t)$. To this end, let us introduce the following transfer functions: $\boldsymbol{H}_{yw}(s) = \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B}$ and $\boldsymbol{H}_{xw}(s) = (s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B}$. When system (B.16), (B.17) is asymptotically stable, $\boldsymbol{x}(t)$ and $\boldsymbol{y}(t)$ are stationary stochastic processes and their spectral density is, respectively,

$$\begin{aligned} \boldsymbol{S}_x(\omega) &= \boldsymbol{H}_{xw}(\mathrm{j}\omega)\boldsymbol{S}_w(\omega)\boldsymbol{H}_{xw}(-\mathrm{j}\omega)^{\mathrm{T}} \\ \boldsymbol{S}_y(\omega) &= \boldsymbol{H}_{yw}(\mathrm{j}\omega)\boldsymbol{S}_w(\omega)\boldsymbol{H}_{yw}(-\mathrm{j}\omega)^{\mathrm{T}} + \boldsymbol{S}_v(\omega). \end{aligned} \tag{B.18}$$

The mean of $\boldsymbol{y}(t)$, $\boldsymbol{m}_y$ is computed from $\boldsymbol{m}_y = \boldsymbol{H}_{yw}(0)\boldsymbol{m}_w + \boldsymbol{m}_v$.

When $\boldsymbol{w}(t)$ and $\boldsymbol{v}(t)$ are zero-mean white noise processes with intensities $\boldsymbol{S}_w$ and $\boldsymbol{S}_v$, one can compute the mean and variance of $\boldsymbol{x}(t)$ ($\boldsymbol{m}_x(t)$ and $\boldsymbol{Q}_x(t)$) directly from the state-space model. Let $\boldsymbol{m}_0$ and $\boldsymbol{Q}_0$ denote the mean and variance of the initial state, then $\boldsymbol{m}_x(t)$ and $\boldsymbol{Q}_x(t)$ are governed by the following differential equations:

$$\frac{\mathrm{d}\boldsymbol{m}_x(t)}{\mathrm{d}t} = \boldsymbol{A}\boldsymbol{m}_x(t), \quad \boldsymbol{m}_x(0) = \boldsymbol{m}_0 \tag{B.19}$$

$$\frac{\mathrm{d}\boldsymbol{Q}_x(t)}{\mathrm{d}t} = \boldsymbol{A}\boldsymbol{Q}_x(t) + \boldsymbol{Q}_x(t)\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{B}\boldsymbol{S}_w\boldsymbol{B}^{\mathrm{T}}, \quad \boldsymbol{Q}_x(0) = \boldsymbol{Q}_0 \tag{B.20}$$

The variance of $\boldsymbol{y}(t)$ is deduced from (B.17), namely

$$\boldsymbol{Q}_y(t) = \boldsymbol{C}\boldsymbol{Q}_x(t)\boldsymbol{C}^{\mathrm{T}} + \boldsymbol{S}_v. \tag{B.21}$$

If moreover the linear time-invariant system (B.16), (B.17) is asymptotically stable, $\boldsymbol{x}(t)$ tends to a weakly stationary stochastic process with zero mean and variance $\bar{\boldsymbol{Q}}_x$ given as the solution of the following algebraic Lyapunov equation

$$\boldsymbol{A}\bar{\boldsymbol{Q}}_x + \bar{\boldsymbol{Q}}_x\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{B}\boldsymbol{S}_w\boldsymbol{B}^{\mathrm{T}} = \boldsymbol{O}. \tag{B.22}$$

Example Covariance calculation - continuous-time case
A stationary random signal $w(t)$ has the autocovariance function

$$R_{ww}(\tau) = \sigma_w^2 e^{-\beta|\tau|}.$$

We wish to calculate the variance σ_w^2.

The signal $w(t)$ is generated by a dynamical (low pass) filter,

$$\dot{w}(t) = -\beta w(t) + \sqrt{2\beta}\sigma_w v(t),$$

where $v(t)$ is a white noise with intensity 1. The variance of $w(t)$ is obtained from the Lyapunov equation,

$$\begin{aligned} 0 &= -\beta Q - Q\beta + \sqrt{2\beta}\sigma_w \sigma_w \sqrt{2\beta} \\ &\Rightarrow -2\beta Q + 2\beta\sigma_w^2 = 0 \\ &\Rightarrow Q = \sigma_w^2 \end{aligned}$$

which is the same result as was obtained by the frequency domain calculation (B.9). □

Example Covariance calculation - low-pass filter
A stationary random signal $w(t)$ with the autocorrelation function

$$R_{ww}(\tau) = \sigma_w^2 e^{-\beta|\tau|}$$

is passed through a stable high-pass filter

$$H(s) = \frac{\alpha}{s+\alpha}$$

with the equivalent state-space representation

$$\begin{aligned} \dot{x}(t) &= -\alpha x(t) + \alpha w(t) \\ y(t) &= x(t). \end{aligned}$$

Hence, the filter acting on $v(t)$ is

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} w(t) \\ x(t) \end{pmatrix} &= \begin{pmatrix} -\beta & 0 \\ \alpha & -\alpha \end{pmatrix}\begin{pmatrix} w(t) \\ x(t) \end{pmatrix} + \begin{pmatrix} \sigma_w\sqrt{2\beta} \\ 0 \end{pmatrix} v(t) \\ y(t) &= \begin{pmatrix} 0 & 1 \end{pmatrix}\begin{pmatrix} w(t) \\ x(t) \end{pmatrix} \end{aligned}$$

The covariance is symmetric,

$$\boldsymbol{Q} = \begin{pmatrix} \sigma_w^2 & \sigma_{wx}^2 \\ \sigma_{xw}^2 & \sigma_x^2 \end{pmatrix} = \begin{pmatrix} a & c \\ c & b \end{pmatrix}$$

and with $S_{\mathrm{v}} = 1$, the Lyapunov equation gives

$$\begin{pmatrix} -\beta & 0 \\ \alpha & -\alpha \end{pmatrix} \begin{pmatrix} a & c \\ c & b \end{pmatrix} + \begin{pmatrix} a & c \\ c & b \end{pmatrix} \begin{pmatrix} -\beta & \alpha \\ 0 & -\alpha \end{pmatrix} + \\ + \begin{pmatrix} \sigma_w \sqrt{2\beta} \\ 0 \end{pmatrix} \begin{pmatrix} \sqrt{\beta}\sigma_w \sqrt{2} & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

equivalent to the three conditions

$$\begin{aligned} -2a\beta + 2\sigma_w^2 \beta &= 0 \\ -2b\alpha + 2c\alpha &= 0 \\ a\alpha - c(\alpha + \beta) &= 0 \end{aligned}$$

which gives

$$\begin{aligned} -2a\beta + 2\sigma_w^2 \beta = 0 &\Rightarrow a = \sigma_w^2 \\ a\alpha - c(\alpha + \beta) = 0 &\Rightarrow c = a \frac{\alpha}{\alpha + \beta} \\ -2b\alpha + 2c\alpha = 0 &\Rightarrow b = c. \end{aligned}$$

Hence

$$Q = \begin{pmatrix} 1 & \frac{\alpha}{\alpha+\beta} \\ \frac{\alpha}{\alpha+\beta} & \frac{\alpha}{\alpha+\beta} \end{pmatrix} \sigma_w^2$$

and

$$\sigma_y^2 = CQC^{\mathrm{T}} = \frac{\alpha}{\alpha + \beta} \sigma_w^2 \tag{B.23}$$

which is the desired result. □

Similar results can be obtained for the analysis of a discrete-time system subject to a white noise input sequence. More specifically, consider a discrete-time random process defined as the output of the following discrete-time linear time-invariant system

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{w}(k) \tag{B.24}$$

$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{v}(k), \tag{B.25}$$

where $\boldsymbol{w}(k)$ and $\boldsymbol{v}(k)$ are mutually uncorrelated white noise sequences with variance $\boldsymbol{Q}_w$ and $\boldsymbol{Q}_v$.

Let the mean and variance of the initial state of system (B.24), (B.25) be $\boldsymbol{m}_0$ and $\boldsymbol{Q}_0$ respectively. The mean value function of the random sequence $\boldsymbol{x}(k)$ is then given by

$$\boldsymbol{m}_x^d(k+1) = \boldsymbol{A}\boldsymbol{m}_x^d(k), \quad \boldsymbol{m}_x^d(0) = \boldsymbol{m}_0$$

and its variance is the solution of the following discrete Lyapunov equation:

$$\boldsymbol{Q}_x^d(k+1) = \boldsymbol{A}\,\boldsymbol{Q}_x^d(k)\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{B}\,\boldsymbol{Q}_w\boldsymbol{B}^{\mathrm{T}}, \quad \boldsymbol{Q}_x^d(0) = \boldsymbol{Q}_0$$

If the linear time-invariant system is asymptotically stable, the random sequence $\boldsymbol{x}(k)$ tends to a stationary process with zero mean and with variance $\bar{\boldsymbol{Q}}_x^d$ given as the solution of the following discrete algebraic Lyapunov equation $\bar{\boldsymbol{Q}}_x^d = \boldsymbol{A}\,\bar{\boldsymbol{Q}}_x^d\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{Q}_w$. The variance of the corresponding output is computed from

$$\bar{\boldsymbol{Q}}_y^d = \boldsymbol{C}\,\bar{\boldsymbol{Q}}_x^d\boldsymbol{C}^{\mathrm{T}} + \boldsymbol{Q}_v.$$

Generation of coloured noise. In the previous section, one was given a system with white noise inputs, and one had to compute the spectral density of its output. Quite often, the reverse problem is encountered; one has to generate a coloured noise with a given rational spectral density. For the sake of simplicity, only the case of a scalar stochastic process is considered. Such a coloured noise can be obtained by entering a white noise process into an appropriate stable filter as indicated in Fig. B.1.

The computation of the filter is based on relation (B.18) with $S_v(\omega) = 0$ and $S_w(\omega) = 1$,

$$S_y(\omega) = H_{yw}(\mathrm{j}\omega)H_{yw}(-\mathrm{j}\omega). \tag{B.26}$$

Introducing $s = j\omega$, the right hand side of (B.26) can be written:

$$F_{yw}(s) = H_{yw}(s)H_{yw}(-s) \tag{B.27}$$

When z_i (p_i) is a zero (pole) of $H_{yw}(s)$, $-z_i$ ($-p_i$) is a zero (pole) of $H_{yw}(-s)$. Besides, every complex pole appears with its complex conjugate in $F_{yw}(s)$. The poles and zeros of $F_{yw}(s)$ are thus symmetric with respect to the imaginary axis. Letting z_i^-, $i = 1, \ldots, m$ and p_i^-, $i = 1, \ldots, n$ denote the zeros and poles of $F_{yw}(s)$ with negative real part, one can obtain the filter transfer function $H(s)$ as

$$H(s) = k\frac{\prod_{i=1}^{m}(s - z_i^-)}{\prod_{i=1}^{n}(s - p_i^-)},$$

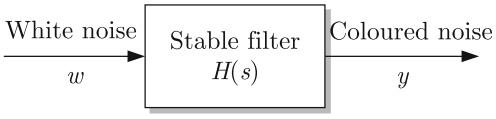


Fig. B.1 Coloured noise generated by a filtered white noise

where k is a gain chosen so that $S_y(0)$ has the desired value.

Sampling a continuous-time system–discrete-time noise covariance. Consider a continuous process described by the linear differential equation

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_{\mathrm{c}}\boldsymbol{x}(t) + \boldsymbol{B}_{\mathrm{c}}\boldsymbol{v}(t), \tag{B.28}$$

where $\boldsymbol{v}(t)$ is a zero mean white noise process with intensity $\boldsymbol{S}_{vv}$ and autocorrelation $E\{\boldsymbol{v}(t)\boldsymbol{v}(t-\tau)^{\mathrm{T}}\} = \boldsymbol{S}_{vv}\delta(\tau)$. For a scalar process, $E\{v(t)v(t-\tau)^T\} = \sigma_{\mathrm{v}}^2\delta(\tau)$.

Integration of (B.28) over a sampling period T_s, from time t_k to t_{k+1}, gives the state at time t_{k+1},

$$\boldsymbol{x}(t_{k+1}) = \mathrm{e}^{\boldsymbol{A}_{\mathrm{c}}T_{\mathrm{s}}}\boldsymbol{x}(t_k) + \int_{t_k}^{(t_{k+1})} e^{\boldsymbol{A}_{\mathrm{c}}(t_{k+1}-\tau)}\boldsymbol{B}_{\mathrm{c}}w(\tau)\mathrm{d}\tau. \tag{B.29}$$

We would like to obtain a discrete-time representation of Eq. (B.28), for constant T_{s}, in the form

$$\boldsymbol{x}(k+1) = \boldsymbol{A}_{\mathrm{d}}\boldsymbol{x}(k) + \boldsymbol{w}(k). \tag{B.30}$$

The sampled noise term $\boldsymbol{w}(k)$ is hence determined from the continuous $\boldsymbol{v}(t)$ as,

$$\boldsymbol{w}(k) = \int_{t_k}^{t_{k+1}} \mathrm{e}^{\boldsymbol{A}_{\mathrm{c}}(t_{k+1}-\tau)}\boldsymbol{B}_{\mathrm{c}}v(\tau)\mathrm{d}\tau. \tag{B.31}$$

The covariance $\boldsymbol{Q}_{ww}$ of the discrete-time noise $\boldsymbol{w}(k)$ is then,

$$\begin{aligned}
&\boldsymbol{Q}_{ww} = \mathrm{E}\{\boldsymbol{w}\boldsymbol{w}^{\mathrm{T}}\} = \\
&\mathrm{E}\left\{\left(\int_{t_k}^{t_{k+1}} \mathrm{e}^{\boldsymbol{A}_{\mathrm{c}}(t_{k+1}-\tau_1)}\boldsymbol{B}_{\mathrm{c}}\boldsymbol{v}(\tau_1)\mathrm{d}\tau_1\right)\left(\int_{t_k}^{t_{k+1}} \mathrm{e}^{\boldsymbol{A}_{\mathrm{c}}(t_{k+1}-\tau_2)}\boldsymbol{B}_{\mathrm{c}}\boldsymbol{v}(\tau_2)\mathrm{d}\tau_2\right)^{\mathrm{T}}\right\} \\
&= \int_{t_k}^{t_{k+1}}\int_{t_k}^{t_{k+1}} \mathrm{e}^{\boldsymbol{A}_{\mathrm{c}}(t_{k+1}-\tau_1)}\boldsymbol{B}_{\mathrm{c}}\mathrm{E}\{\boldsymbol{v}(\tau_1)\boldsymbol{v}(\tau_2)^{\mathrm{T}}\}\boldsymbol{B}_{\mathrm{c}}{}^{\mathrm{T}}\mathrm{e}^{\boldsymbol{A}_{\mathrm{c}}(t_{k+1}-\tau_2)^{\mathrm{T}}}\mathrm{d}\tau_1\mathrm{d}\tau_2
\end{aligned} \tag{B.32}$$

With the approximation of the matrix exponential

$$\mathrm{e}^{\boldsymbol{A}\tau} = \boldsymbol{I} + \boldsymbol{A}\tau + \frac{1}{2!}\boldsymbol{A}^2\tau^2 + \frac{1}{3!}\boldsymbol{A}^3\tau^3 \ldots \tag{B.33}$$

a first-order approximation to Eq. (B.32) includes only one term in the series expansion. We then get the approximation

$$\boldsymbol{Q}_{ww} \simeq \boldsymbol{B}\boldsymbol{S}\boldsymbol{B}^{\mathrm{T}}T_{\mathrm{s}} \tag{B.34}$$

The approximate result in Eq. (B.34) is widely used. If more exact results are needed, numerical discretisation is preferred over analytical solution of Eq. (B.32).

Appendix C
Nomenclature

The symbols are used according to the following conventions. Scalars are represented by italics like i, j, a, s, vectors by bold lower case letters like $\boldsymbol{x}$, $\boldsymbol{u}$ and matrices by bold upper case letters like $\boldsymbol{A}$, $\boldsymbol{K}$. Sets are denoted by calligraphic letters like $\mathcal{Y}$.

Abbreviation	Meaning	Introduced on p.
ARR	Analytical redundancy relation	173
CUSUM	CUmulative SUM	281
FMEA	Failure modes and effects analysis	83
GLR	Generalised likelihood ratio	281
LFT	Linear fractional transformation	455
LQ	Linear quadratic (regulator)	358
LTI	Linear time-invariant	265
MSO	Minimal structurally overdetermined	171
UM	Use-mode	80

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8

Appendix D
Terminology

The terminology used in diagnosis and fault-tolerant control literature has only during the recent years approached a coherency in the published material. The Safeprocess Technical Committee of IFAC, the International Federation of Automatic Control, has compiled a list of suggested definitions [155], which is generally in accordance with the terminology used throughout this book.

An exception is the use of the word estimation where the Safeprocess terminology is identification. The word identification is widely used as a synonym for system identification or system parameter identification. Estimation is commonly used when the magnitude of a signal is reconstructed, which is what is done in this book.

Active fault-tolerant control system	A fault-tolerant system where faults are explicitly detected and accommodated. Opposite of a passive fault-tolerant system.
Admissibility	Admissibility refers to a control law, a diagnosis algorithm, a communication scheme, etc. It is the property that a specified objective is achieved while a specified set of constraints are satisfied.
Analytical redundancy	Use of two or more, but not necessarily identical ways to determine a variable where one way uses a mathematical process model in analytical form.
Availability	Likelihood that a system or equipment will operate satisfactorily and effectively at any given point in time.
Constraint	The limitation imposed by nature (physical laws) or man. It permits the variables to take only certain values in the variable space.

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8

Coordination In distributed systems, local estimation or diagnosis results may be only partial or even contradictory. A coordination level is needed to obtain global consistent results from the local results.

Decision logic The functionality that determines which remedial action(s) to execute in case of a reported fault and which alarm(s) shall be generated.

Decomposable specification In a distributed system, an overall system specification is decomposable if it is equivalent to a set of specifications that apply to each subsystem. Decomposable specifications are easier to handle, since each subsystem is to be (fault-tolerant) controlled with respect to its own specification. In most cases, specifications are not decomposable because of the coupling between the subsystems.

Discrepancy An abnormal behaviour of a physical value or inconsistency between more physical values and the relationship between them.

Distributed system A distributed system is composed of a set of interacting and communicating subsystems that cooperate in order to achieve a global objective. In distributed control each subsystem computes a subset of the control signals, in distributed estimation, each subsystem estimates a subset of the states, in distributed diagnosis each subsystem elaborates a partial diagnosis decision, etc. In all cases, the data available to a subsystem depend on the information pattern that is implemented.

Error Deviation between a measured or computed value (of an output variable) and the true, specified or theoretically correct value.

Fail-safe The ability to sustain a failure and retain the capability to make a safe close-down. A system where the occurrence of a single fault can be determined but not isolated and where the fault cannot be accommodated to continue operation.

Fail-operational	The ability to sustain any single point failure.
Failure	Permanent interruption of a systems ability to perform a required function under specified operating conditions.
Failure effect	The consequence of a failure mode on the operation, function, or status of an item.
Failure mode	Particular way in which a failure can occur.
Fault	Unpermitted deviation of at least one characteristic property or parameter of a system from its acceptable/usual/standard condition. A fault is the occurrence of a failure mode.
Fault accommodation	The action of changing the control law in response to fault, without switching off any system component. In fault accommodation, faulty components are stillkept in operation thanks to an adapted control law.
Fault detection	Determination of faults present in a system and time of detection.
Fault detector	An algorithm that performs fault detection and isolation.
Fault diagnosis	Determination of kind, size, location, and time of occurrence of a fault. Fault diagnosis includes fault detection, isolation and estimation.
Fault estimation	Determination of a model of the faulty system.
Fault identification	Determination of the sihze and time-varying behaviour of a fault. Follows fault isolation. Used as a synonym for fault estimation.
Fault isolation	Determination of the location of a fault. Follows fault detection.
Fault modelling	Determination of a mathematical model to describe a specific fault effect.
Fault propagation analysis	Analysis to determine how certain fault effects propagate through the considered system.
Fault recovery	The result of a successful fault accommodation or system reconfiguration.
Fault-tolerant system	A system where a fault is recovered with or without performance degradation, but

	a single fault does not develop into a failure on subsystem or system level.
Hardware redundancy	Use of more than one independent instrument to accomplish a given function.
Incipient fault	A fault where the effect develops slowly e.g. clogging of a valve. In opposite to an abrupt fault.
Information pattern	In a distributed system, the data available to each subsystem may be produced by the subsystem itself, or received from other subsystems through the communication network. The information pattern defines which data are available to each subsystem through the communication network (and therefore it influences the communication cost).
Objective	The control specification, the aim of the control system.
Objective reconfiguration	The action of changing the objective of the control system. Objective reconfiguration is mandatory when unrecoverable faults for the current objective occur.
Passive fault-tolerant control system	A fault-tolerant system where faults are not explicitly detected and accommodated, but the controller is designed to be insensitive to a certain restricted set of faults. Contrary to an active fault-tolerant system.
Qualitative model	A system model describing the behaviour with relations among system variables and parameters in heuristic terms such as causalities or if-then rules.
Quantitative model	A system model describing the behaviour with relations among system variables and parameters in analytical terms such as differential or difference equations.
Reconfigurability	The possibility to recover a fault by using the reconfiguration strategy: switching off the faulty components, and changing the control law so as to achieve the specified objective by using only the healthy components.
Reconfiguration effort	A measure of the changes that are introduced in the information pattern and in

	the control system in order to recover a fault. In distributed systems changes in the information pattern impact the communication network load, while changes in the local controls impact the number of subsystems that have to be reconfigured.
Recoverability	Possibility ot accommodate the fault or to reconfigure the system if fault occurs.
Reliability	Probability of a system to perform a required function under normal conditions and during a given period of time.
Reliability overcost	The cost that has to be paid, in terms of the system performance, for using a reliable control law instead of an optimal control law.
Remedial action	A correcting action (reconfiguration or a change in the operation of a system) that prevents a certain fault to propagate into an undesired end effect.
Residual	Fault information carrying signals, based on deviation between measurements and model-based computations.
Safety system	Electronic system that protects local subsystems from permanent damage or damage to environment when potential dangerous events occur.
Sensor fusion	Integration of information from different sensors taking into account the quality of the measurements provided by each of them.
Severity	A measure on the seriousness of fault effects using verbal characterisation. Severity considers the worst-case damage to equipment, damage to environment, or degradation of a system's operation.
Structural analysis	Analysis of the structural properties of the models, i. e. properties that are independent on the actual values of the parameter.
System reconfiguration	The action of switching off the faulty components and accordingly changing the control law, in response to a fault. In system reconfiguration, faulty components are no longer employed.

Supervision	Monitoring of a physical system and taking appropriate actions to maintain the operation in the case of faults.
Supervisor	A function that performs supervision using results of fault diagnosis, determines remedial actions when needed, and execute corrective actions to handle faults.
Threshold	Limit value of a residual's deviation from zero, so if exceeded, a fault is declared as detected.

Appendix E
Dictionary

This appendix shows the main notions in English and the native languages of the four authors.

English	German	French	Danish
Basic notions in systems and control theory			
Actuator	Aktor, Stellglied	Actionneur	Aktuator
Analysis	Analyse	Analyse	Analyse
Architecture	Aufbau, Architektur	Architecture	Arkitektur
Automaton	Automat	Automate	Automat
Block diagram	Blockschaltbild	Schéma fonction nel	Blokdiagram
Behaviour	Verhalten	Comportement	Opførsel
Closed-loop system	Regelkreis	Système en boucle fermée	Lukketsløjfe system
Continuous-variable system	Wertkontinuierliches System	Système à variables continues	Kontinuert system
Control	Steuerung, Regelung	Commande, régulation	Styring, regulering
Controllability	Steuerbarkeit	Gouvernabilité, commandabilité	Styrbarhed
Controller	Regler	Régulateur, contrôleur	Regulator
Control design	Reglerentwurf	Conception de régulateur	Regulator design
Control input	Stellgröße	Commande	Styresignal (procespåvirkning)
Control loop	Regelkreis	Boucle de régulation	Reguleringssløjfe
Control objective	Regelungsziel	Consigne, objectif à réaliser	Regulerings formål
Control output	Regelgröße	Signal de commande	Styret variabel
Discrete-event system	Ereignisdiskretes System	Système à événements discrets	Diskret-hændelses system
Disturbance	Störung	Perturbation	Forstyrrelse
Event	Ereignis	Événement	Hændelse
Feedback	Rückführung	Boucle de retour	Modkobling
Feedback control	Regelung	Régulation par rétroaction	Lukketsløjfe styring eller—regulering

(continued)

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8

(continued)

English	German	French	Danish
Feedforward control	Steuerung (in der offenen Wirkungskette)	Régulation avec action Anticipative	Fremkobling
Input	Eingang	Entrée	Indgang
Measurement	Messung, Messwert	Signal de mesure	Måling
Model	Modell	Modèle	Model
Modelling	Modellbildung	Modélisation	Modellering
Model uncertainties	Modellunbestimmtheiten	Erreur de modélisation	Model usikkerhed
Objective function	Gütefunktion	Fonction de coût	Kostfunktion
Observability	Beobachtbarkeit	Observabilité	Observerbarhed
Observation	Beobachtung	Observation	Observation
Observer	Beobachter	Observateur	Observer
Open-loop system	Offene Kette	Système en boucle ouverte	Åbensløjfe system
Optimal control	Optimale Steuerung	Régulation optimale, commande optimale	Optimal regulering
Output	Ausgang	Sortie	Udgang
Petri net	Petrinetz	Réseau de Petri	Petri net
Performance	Regelgüte	Performance	Performance
Prediction	Vorhersage, Prädiktion	Prédiction	Prediktion
Qualitative model	Qualitatives Modell	Modèle qualitatif	Qualitativ model
Quantised system	Quantisiertes System	Système quantifié	Kvantiseret system
Requirement	Forderung	Cahier des charges	Krav
Specification	Güteforderung	Spécification	Specifikation
Sensor	Sensor	Capteur	Giver, sensor
Set point	Sollwert	Point de fonctionnement, grandeur de référence	Setpunkt
Signal	Signal	Signal	Signal
Stability	Stabilität	Stabilité	Stabilitet
State	Zustand	État	Tilstand
State space	Zustandsraum	Espace d'état	Tilstandsrum
Structure	Struktur	Structure	Struktur
Structure graph	Strukturgraph	Graphe structurel	Struktur graf
System	System	Système	System
System identification	Systemidentifikation	Identification des systèmes	System identifikation
Notions describing the system subject to faults			
Actuator fault	Fehler im Stellglied	Défaut d'actionneur	Aktuator fejl
Abrupt fault	Plötzlich eintretender Fehler	Défaut instantané	Pludselig fejl
Availability	Verfügbarkeit	Disponibilité	Tilgængelighed
Consistency	Widerspruchsfreiheit, Konsistenz	Cohérence, compatibilité	Konsistens
Consistency-based diagnosis	Konsistenzbasierte Diagnose	Diagnostic par invalidation (réfutation)	Konsistensbaseret diagnose
Decision logic	Entscheidungslogik	Logique de décision	Beslutningslogik
Dependability	Verfügbarkeit	Sûreté de foncionnement	Pålidelighed

(continued)

(continued)

English	German	French	Danish
Deviation	Abweichung	Déviation	Afvigelse
Diagnosis	Diagnose	Diagnostic	Diagnose
Diagnostic algorithm	Diagnosealgorithmus	Algorithme de diagnostic	Diagnose algorithme
Evaluation	Bewertung	Évaluation	Evaluering
Failure	Versagen	Panne	Nedbrud
Fault	Fehler	Défault, défaillance	Fejl
Faultless system	Fehlerfreies System	Système sain	Fejlfrit system
Faulty system	Fehlerbehaftetes System	Système en défaut	Fejlbehæftet system
Fault-tolerant control	Fehlertolerante Steuerung	Commande tolérante aux fautes	Fejltolerant regulering
Fault detection	Fehlerdetektion	Détection de défaillances	Fejldetektion
Fault diagnosis	Fehlerdiagnose	Diagnostic de défaillances	Fejldiganose
Fault effects	Fehlerwirkung	Effets d'une défaillance	Fejleffekt
Fault estimation	Schätzung der Fehlergröße	Estimation de défaillance	Fejlestimation
Fault identification	Fehleridentifikation	Identification de défaillance	Fejlidentifikation
Fault isolation	Fehlerisolation	Localisation des défaillances	Fejlisolation
Fault probability	Fehlerwahrscheinlichkeit	Probabilité d'une défaillance	Fejlsandsynlighed
Fault propagation	Fehlerfortpflanzung	Propagation d'une défaillance	Fejlpropagering
Incipient fault	Fehler, der sich langsam entwickelt	Défaillance naissante	Fejl der udvikles langsomt
Maintenance	Wartung	Entretien, maintenance	Vedligehold
Model-based diagnosis	Modellbasierte Diagnose	Diagnostic fondé sur un modéle	Modelbaseret diagnose
Reconfiguration	Rekonfiguration	Reconfiguration	Rekonfiguration
Redundancy	Redundanz	Redondance	Redundans
Reliability	Zuverlässigkeit	Fiabilité	Pålidelighed
Remedial action	Korrektur	Action corrective	Handling der skal genoprette normal funktion
Repair	Reparatur	Réparation	Reparation
Residual	Residuum	Résidu	Residual
Robustness	Robustheit	Robustesse	Robusthed
Safety	Sicherheit	Sécurité, sûreté	Sikkerhed
Sensor fault	Sensorfehler	Défaut de capteur	Sensor fejl
Sensor fusion	Sensorfusion	Fusion multicapteurs	Sensorfusion
Shut off	Abschaltung	Arrêt	Nedlukning
Symptom	Symptom	Symptôme	Symptom
Supervision	Überwachung	Supervision	Overvågning
Threshold	Schranke	Seuil	Grænseværdi
Tolerance	Toleranz	Tolérance	Tolerance

References

1. A. Ahmad, M. Gani, F. Yang, Decentralized robust Kalman filtering for uncertain stochastic systems over heterogeneous sensor networks. Signal Process. **88**(8), 1919–1928 (2008)
2. J. Aidemark, P. Folkesson, J. Karlsson, A framework for node-level fault tolerance in distributed real time systems, in *Proceedings of the International Conference on Dependable Systems and Networks* (2005)
3. A. Aitouche, A.L. Gehin, N. Flix, G. Dumortier, Generic control/command distributed system. Application to the supervision of moving stage sets in theaters. Eur. J. Control **8**, 64–75 (2002)
4. J.S. Albus, F.G. Proctor, A reference model architecture for intelligent hybrid control systems, in *IFAC 13th World Congress*, San Francisco (1996), pp. 473–488
5. H. Alwi, C. Edwards, C.P. Tan, *Fault Detection and Fault-Tolerant Control Using Sliding Modes* (Springer, Berlin, 2011)
6. J. Armengol, A. Bregon, E. Escobet, R. Gelso, M. Krysander, M. Nyberg, X. Olive, B. Pulido, L. Trave-Massuyes, Minimal structurally overdetermined sets for residual generation: a comparison of alternative approaches, in *IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Barcelona (2009)
7. A.S. Asratian, T.M.J. Denley, R. Häggkvist, *Bipartite Graphs and Their Applications* (Cambridge University Press, Cambridge, 2008)
8. K.J. Aström, P. Albertos, M. Blanke, A. Isidori, R. Sanz, W. Schaufelberger, *Control of Complex Systems* (Springer, London, 2001)
9. K.J. Aström, B. Wittenmark, *Computer Controlled Systems: Theory and Design* (Prentice-Hall, Englewood Cliffs, 1984)
10. A. Avi Ienis, J.-C. Laprie, B. Randell, Dependability and its threats: a taxonomy, in *Building the Information Society*, ed. by R. Jacquart. IFIP Advances in Information and Communication Technology, vol. 156 (Springer, 2004), pp. 91–120
11. M. Basseville, I.V. Nikiforov, *Detection of Abrupt Changes: Theory and Application*. Information and System Science (Prentice Hall, New York, 1993)
12. A. Benveniste, A. Aghasaryan, E. Fabre, R. Boubour, C. Jard, Fault detection and diagnosis in distributed systems: an approach by partially stochastic petri nets. Discret. Event Dyn. Syst.: Theory Appl. **8**, 203–231 (1998)
13. P. Baroni, G. Lamperti, P. Pogliano, M. Zanella, Diagnosis of a class of distributed discrete-event systems. IEEE Trans. **SMC-30**, 731–752 (2000)
14. M. Bayart, A.L. Gehin, M. Staroswiecki, Fault detection and isolation and mode management in smart actuators, in *IFAC SICICA'92*, Malaga (1992)
15. M. Bayart, E. Lemaire, M.A. Péraldi, C. André, External model and syncharts description of an automobile cruise control system. Control Eng. Pract. **7**, 1259–1267 (1999)

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8

16. D. Berdjag, M. Staroswiecki, K. Zhang, M. Abbas-Turki, Reducing the reliability over-cost in reconfiguration-based fault tolerant control under actuator faults. IEEE Trans. Autom. Control **57**(12), 3181–3186 (2012)
17. C. Berge, Two theorems graph theory. Proc. Natl. Acad. Sci. USA **43**, 842–844 (1957)
18. G. Biswas, M.-O. Cordier, J. Lunze, L. Travé-Massuyès, M. Staroswiecki, Diagnosis of complex systems: bridging the methodologies of the FDI and DX communities, Editorial. IEEE Trans. **SMC-34**, 2159–2162 (2004)
19. J. Biteus, M. Nyberg, E. Frisk, An algorithm for computing the diagnoses with minimal cardinality in a distributed system. Eng. Appl. Artif. Intell. **21**(2), 269–276 (2008)
20. J. Biteus, E. Frisk, M. Nyberg, Distributed diagnosis using a condensed representation of diagnoses with application to an automotive vehicle. IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. **41**(6), 1262–1267 (2011)
21. M. Blanke, Ship propulsion losses related to automatic steering and prime mover control. Ph.D. thesis, Technical University of Denmark (1981)
22. M. Blanke, Consistent design of dependable control systems. Control Eng. Pract. **4**, 1305–1312 (1996)
23. M. Blanke, Fault-tolerant sensor fusion with an application to ship navigation, in *IEEE Joint 2005 International Symposium on Intelligent Control and 13th Mediterranean Conference on Control and Automation* (2005), pp. 1385–1390
24. M. Blanke, O. Borch, F. Bagnoli, G. Allasia, Development of an automated technique for failure modes and effect analysis, in *Proceedings of the European Safety and Reliability Conference*, Munich (1999), pp. 839–844
25. M. Blanke, J.S. Andersen, On dynamics of large two stroke diesel engines: new results from identification, in *Proceedings 9th IFAC World Conference*, Budapest (1984)
26. M. Blanke, M.R. Blas, S. Hansen, J.C. Andersen, F. Caponetti, Autonomous robot supervision using fault diagnosis and outdoor semantic mapping, in *Chapter 1 in Fault Diagnosis in Robotic and Industrial Systems*, ed. by G. Rigatos (iConceptPress, USA, 2012), pp. 1–22
27. M. Blanke, S. Hansen, M.R. Blas, Diagnosis for control and decision support in autonomous systems, in *Chapter 1.1 in Control of Complex Systems*, ed. by G. Dimirovski (Springer, 2015) to appear
28. M. Blanke, R. Izadi-Zamanabadi, T.F. Lootsma, Fault monitoring and re-configurable control for a ship propulsion plant. J. Adapt. Control Signal Process. **12**, 671–688 (1998)
29. M. Blanke, T. Frederiksen, J. Kristensen, J. Sandberg Thomsen, Electrical steering system. U.S. Patent 6,693,405 b2
30. M. Blanke, R. Izadi-Zamanabadi, S.A. Bøgh, C.P. Lunau, Fault-tolerant control systems—a holistic view. Control Eng. Pract. **5**, 693–702 (1997)
31. M. Blanke, T. Lorentzen, Satool—a software tool for structural analysis of complex automation systems, in *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes SAFEPROCESS*, Beijing (2006)
32. M. Blanke, M. Staroswiecki, N.E. Wu, Concepts and methods in fault-tolerant control, in *Proceedings of the American Control Conference*, Washington (2001)
33. M. Blanke, M. Staroswiecki, Fault-tolerant control with safe behaviour under multiple actuator or sensor faults—theory and application, in *14th IFAC Safeprocess Symposium*, Beijing (2006)
34. M. Blanke, M. Staroswiecki, Structural design of systems with safe behavior under single and multiple faults, in *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes SAFEPROCESS*, Beijing (2006), pp. 511–516
35. M. Blanke, J.S. Thomsen, Electrical steering of vehicles—fault-tolerant analysis and design. Microelectron. Reliab. **46**, 1415–1420 (2006)
36. M.R. Blas, M. Blanke, Natural environment modeling and fault-diagnosis for automated agricultural vehicle, in *Proceedings 17th IFAC World Congress*, Seoul, Korea (2008), pp. 1590–1595
37. M.R. Blas, M. Blanke, Stereovision with texture learning for fault-tolerant automatic baling. Comput. Electron. Agric. **75**, 159–168 (2011)

38. J. Blesa, V. Puig, J. Saludes, Robust fault detection using polytope-based set-membership consistency test. IET Control Theory Appl. **6**(12), 1767–1777 (2012)
39. M. Staroswiecki, B. Ciubotaru, C. Christophe, Fault tolerant control of the boeing 747 short-period mode using the admissible model matching technique, in *IFAC Symposium Safeprocess*, Beijing (2006), pp. 871–876
40. M. Staroswiecki, B.D. Ciubotaru, N.D. Christov, Modified pseudo-inverse method with generalized linear quadratic regulator for fault tolerant model matching with prescribed stability degree, in *CDC, ECC, 2011*, Orlando (2011)
41. R.K. Boel, J.H. van Schuppen, Decentralized failure diagnosis for discrete-event sysetems with costly communication between diagnosers, in *Workshop on Discrete Event System*, Zaragoza (2002), pp. 175–181
42. S.A. Bøgh, Fault tolerant control systems—a development method and real-life case study. Ph.D. thesis, Department of Control Engineering, Aalborg University, Denmark (1997)
43. S.A. Bøgh, R. Izadi-Zamanabadi, M. Blanke, Onboard supervisor for the & ørsted satellite attitude control system, in *Artificial Intelligence and Knowledge Based Systems for Space, 5th Workshop*, Noordwijk (1995), pp. 137–152
44. B. Buchberger, Gröbner bases: an algorithmic method in polynomial ideal theory. *Multidimensional Systems Theory* (Reidel, Dordrecht, 1985), pp. 184–232
45. C. Bonivento, A. Paoli, L. Marconi, Fault-tolerant control for the ship propulsion system. Control Eng. Pract. **11**, 483–492 (2003)
46. O. Boumaman, G. Dauphin-Tanguy, Bond graph model of a steam generator process and its environment, in *10-th European Simulation Multiconference*, Budapest (1996), pp. 238–242
47. B. Ould, G. Bouamama, R. Biswas, R. Loureiro, Merzouki: graphical methods for diagnosis of dynamic systems: review. Annu. Rev. Control **38**, 199–219 (2014)
48. A. Bouras, M. Bayart, M. Staroswiecki, Specification of smart instruments for distributed intelligent control, in *IEEE International Conference on Systems*, Vancouver (1995)
49. A. Bouras, M. Staroswiecki, Building distributed architectures by the interconnection of intelligent instruments, in *IFAC INCOM'98*, Nancy (1998)
50. A. Bouras, M. Staroswiecki, How can intelligent instruments interoperate in an application framework? A mechanism for taking into account operating constraints, in *IFAC SICICA'97*, Annecy (1997)
51. R. Bukharaev, *Theorie der stochastischen Automaten* (B. G. Teubner, Stuttgart, 1995)
52. F.G. Cabral, M.V. Moreira, O. Diene, J.C. Basilio, A petri net diagnoser for discrete event systems modeled by finite state machines. IEEE Trans. Autom. Control **60**, 59–71 (2015)
53. G.C. Calafiore, F. Abrate, Distributed linear estimation over sensor networks. Int. J. Control **82**(5), 868–882 (2009)
54. S.L. Campbell, R. Nikoukhah, *Auxiliary Signal Design for Failure Detection* (Princeton University Press, Princeton, 2004)
55. C. Cao, F. Lin, Z. Lin, Why event observation: observability revisited. Discret. Event Dyn. Syst. Theory Appl. **7**, 127–149 (1997)
56. J. Carlyle, State-calculable stochastic sequential machines, equivalences and events. Switching circuit theory and logic, in *IEEE Conference Record on Switching Circuit Theory and Logic Design* (1965), pp. 865–870
57. T. Carpentier, R. Litwak, J.-Ph. Cassar, Criteria for the evaluation of FDI systems—application to sensors location, in *IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes*, Hull (1997), pp. 1083–1088
58. M.A. Cash, T.G. Habetler, G.B. Kliman, Insulation failure prediction in induction machines using line-neutral voltages. IEEE Trans. Ind. Appl. **54**, 1234–1239 (1998)
59. C. Cassandras, S. Lafortune, *Introduction to Discrete Event Systems* (Kluwer Academic Publishers, Boston, 1999)
60. A. Çela, M. Ben Gaid, X.-G. Li, S.-I. Niculescu, *Optimal Design of Distributed Control and Embedded Systems*. Series: Communications and Control Engineering (Springer, Berlin, 2014), p. XXIV

61. H. Chakib, A. Khoumsi, Multi-decision diagnosis: decentralized architectures cooperating for diagnosing the presence of faults in discrete event systems. Discret. Event Dyn. Syst. **22**, 333–380 (2012)
62. G. Chartrand, O.R. Oellermann, *Applied and Algorithmic Graph Theory*. Pure and Applied Mathematics (McGraw-Hill Inc., New York, 1993)
63. C.T. Chen, *Linear System Theory and Design* (Holt, Rinehart and Winston, New York, 1984)
64. J. Chen, R.J. Patton, *Robust Model-Based Fault Diagnosis for Dynamic Systems* (Springer, Berlin, 2012)
65. E.Y. Chow, A.S. Willsky, Analytical redundancy and the design of robust failure detection filters. IEEE Trans. **AC-29**, 603–614 (1984)
66. B.D. Ciubotaru, M. Staroswiecki, Anytime algorithm for parametric faults accommodation under handling quality constraints, in *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Barcelona, Spain (2009), pp. 887–892
67. C. Commault, M. Staroswiecki, J.-M. Dion, Component usefulness measures for fault tolerance evaluation, in *IFAC Safeprocess 2012*, Mexico (2012)
68. C. Commault, M. Staroswiecki, J.-M. Dion, Fault tolerance evaluation based on the lattice of system configurations. Int. J. Adapt. Control Signal Process **26**(1), 54–72 (2012)
69. M.-O. Cordier, P. Dague, F. Lévy, M. Dumas, J. Montmain, M. Staroswiecki, L. Travé-Massuyès, AI and automatic control approaches of model-based diagnosis: links and underlying hypothesis, in *IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes*, Budapest (2000), pp. 274–279
70. M.-O. Cordier, Ph. Dague, F. Lévy, J. Montmain, M. Staroswiecki, L. Travé-Massuyès, Conflicts versus analytical redundancy relations: a comparative analysis of the model based diagnosis approach from the artificial intelligence and automatic control perspectives. IEEE Trans. Syst. Man Cybern. Part B: Cybern. **34**(5), 2163–2177 (2004) (Special Issue on Diagnosis of Complex Systems: Bridging the methodologies of the FDI and DX Communities)
71. D. Cox, J. Little, J.D. O'Shea, *Varieties and Algorithms* (Springer, New York 1992)
72. M.J. Daigle, I. Roychoudhury, G. Biswas, X.D. Koutsoukos, A. Patterson-Hine, S. Poll, A comprehensive diagnosis methodology for complex hybrid systems: a case study on spacecraft power distribution systems IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. **40**, 917–931 (2010)
73. B.A. Davey, H.A. Priestley, *Introduction to Lattices and Order* (Cambridge University Press, Cambridge, 2002)
74. M.R. Davoodi, K. Khorasani, H.A. Talebi, H.R. Momeni, Distributed fault detection and isolation filter design for a network of heterogeneous multiagent systems. IEEE Trans. Control Syst. Technol. **22**(3), 1061–1069 (2014)
75. R. Debouk, S. Lafortune, D. Teneketzis, Coordinated decentralized protocols for failure diagnosis of discrete event systems. Discret. Event Dyn. Syst. **10**, 33–79 (2000)
76. P. Declerck, M. Staroswiecki, Characterisation of the canonical components of a structural graph for fault detection in large scale industrial plants, in *European Control Conference*, Grenoble (1991), pp. 298–303
77. J. De Kleer, A. Mackworth, R. Reiter, Characterizing diagnoses and systems. Artif. Intell. **56**(2–3), 197–222 (1992)
78. S.X. Ding, *Model-Based Fault Diagnosis Techniques—Design Schemes, Algorithms and Tools*, 2nd edn. (Springer, London, 2013)
79. S.X. Ding, *Data-Driven Design of Fault Diagnosis and Fault-Tolerant Control Systems* (Springer, Heidelberg, 2014)
80. X. Ding, P.M. Frank, Frequency domain approach and threshold selector for robust model-based fault detection and isolation, in *IFAC Symposium Safeprocess*, vol. 1. Baden-Baden (1991), pp. 307–312
81. D. Ding, Z. Wang, H. Dong, H. Shu, Distributed H_∞ state estimation with stochastic parameters and nonlinearities through sensor networks: the finite-horizon case. Automatica **48**, 1575–1585 (2012)
82. S. Diop, Elimination in control theory. Math. Control Signals Syst. **4**, 17–32 (1991)

83. F. Dörfler, F. Pasqualetti, F. Bullo, Continuous-time distributed observers with discrete communication. IEEE J. Sel. Top. Signal Process. **7**(2), 296–304 (2013)
84. S. Drüppel, J. Lunze, M. Fritz, Modeling of asynchronous discrete-event systems as networks of input-output automata, in *17th IFAC Congress*, Seoul (2008), pp. 544–549
85. G.J.J. Ducard, *Fault-tolerant Flight Control and Guidance Systems* (Springer, Berlin, 2009)
86. A.L. Dulmage, N.S. Mendelsohn, Covering of bi-partite graphs. Can. J. Math. **10**, 517–534 (1958)
87. A.L. Dulmage, N.S. Mendelsohn, A structure theory of bi-partite graphs of finite exterior dimension. Trans. R. Soc. Can. Sect. III **53**, 1–13 (1959)
88. D. Düştegör, E. Frisk, V. Cocquempot, M. Krysander, M. Staroswiecki, Structural analysis of fault isolability in the DAMADICS benchmark. Control Eng. Pract. **14**(6), 597–608 (2006)
89. J. Edmonds, Paths, trees and flowers. Can. J. Math. **17**, 449–467 (1965)
90. C. Edwards, S.K. Spurgeon, A sliding mode observer based FDI scheme for the ship benchmark. Eur. J. Control **6**(4), 341–356 (2000)
91. C. Edwards, T. Lombaerts, H. Smaili, *Fault Tolerant Flight Control: A Benchmark Challenge* (Springer, Berlin, 2010)
92. A. Emami-Naeini, M.M. Akhter, S.M. Rock, Effect of model uncertainty on failure detection—the threshold selector. IEEE Trans. **AC-33**, 1106–1115 (1988)
93. D. Eriksson, E. Frisk, M. Krysander, A method for quantitative fault diagnosability analysis of stochastic linear descriptor model. Automatica **49**, 1591–1600 (2013)
94. N. Ertugrul, W. Soong, G. Dostal, D. Saxon, Fault tolerant motor drive system with redundancy for critical applications, in *Proceedings of the IEEE 33rd Annual Power Electronics Specialists Conference* (2002), pp. 1457–1462
95. E. Fabre, Diagnosis and automata, in *Control of Discrete-Event Systems*, ed. by C. Seatzu, M. Silva, J.H. van Schuppen (Springer, Heidelberg, 2013)
96. S. Fang, M. Blanke, Fault monitoring and fault recovery control for position-moored vessels. Int. J. Appl. Comput. Sci. Control AMCS **21**(3), 467–478 (2011)
97. S. Fang, B.J. Leira, M. Blanke, Position mooring control based on a structural reliability criterion. Struct. Saf. **41**, 97–106 (2013)
98. S. Fang, M. Blanke, B.J. Leira, Mooring system diagnosis and structural reliability based control for position-moored vessels. Control Eng. Pract. **36**, 12–26 (2015)
99. M.E. Feki, A. Jardin, W. Marquis-Favre, L. Krähenbühl, D. Thomasset, Structural analysis by bond graph approach: duality between causal and bicausal procedures. J. Syst. Control Eng. **226**, 82–100 (2012)
100. R. Ferrari, T. Parisini, M. Polycarpou, Distributed fault diagnosis with overlapping decompositions: an adaptive approximation approach. IEEE Trans. Autom. Control **54**(4), 794–799 (2009)
101. C. Fetzer, F. Cristian, Fail-awareness: an approach to construct fail-safe systems. Real-Time Syst. **24**(2), 203–238 (2003)
102. B. Fong, N. Ansari, A. Fong, Prognostics and health management for wireless telemedicine networks. IEEE Wirel. Commun. **19**(5), art. no. 6339476, 83–89 (2012)
103. P.M. Frank, Analytical and qualitative model-based fault diagnosis—a survey and some new results. Eur. J. Control **2**, 6–28 (1996)
104. M. Fouladirad, L. Freitag, I. Nikiforov, Optimal fault detection with nuisance parameters and a general covariance matrix. Int. J. Adapt. Control Signal Process. **22**(5), 431–439 (2008)
105. P.M. Frank, X. Ding, Frequency domain approach to optimally robust residual generation and evaluation using model-based fault diagnosis. Automatica **30**, 789–804 (1994)
106. M.L. Fravolini, V. Brunori, G. Campa, M.R. Napolitano, M. La Cava, Structured analysis approach for the generation of structured residuals for aircraft FDI. IEEE Trans. Aerosp. Electron. Syst. **45**(4), 1466–1482 (2009)
107. E. Frisk, M. Krysander, M. Nyberg, J. Åslund, A toolbox for design of diagnosis systems, in *Proceedings of the IFAC Safeprocess'2006* (2006)
108. E. Frisk, M. Krysander, J. Åslund, Sensor placement for fault isolation in linear differential-algebraic systems. Automatica **45**(2), 364–371 (2009)

109. E. Frisk, A. Brgon, J. Åslund, M. Krysander, B. Pulido, G. Biswas, Diagnosability analysis considering causal interpretations for differential constraints. IEEE Trans. Syst. Man Cybern. **42**, 1216–1229 (2012)
110. L.R. Ford, D.R. Fulkerson, Maximal flow through a network. Can. J. Math. **8**, 399–404 (1956)
111. L.R. Ford, D.R. Fulkerson, A simple algorithm for finding maximal network flows and an application to the hitchcock problem. Can. J. Math. **9**, 210–218 (1957)
112. C.W. Frei, Fault-rolerant methods in anesthesia. Ph.D. thesis, EHT Zürich (2000)
113. C.W. Frei, F.J. Kraus, M. Blanke, Recoverability viewed as a system property, in *Proceedings of the European Control Conference*, Karlsruhe (1999)
114. C. Fritsch, J. Lunze, M. Schwaiger, V. Krebs, Remote diagnosis of discrete-event systems with on-board and off-board components, in *IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS)*, Beijing (2006)
115. R. Galeazzi, M. Blanke, N.K. Poulsen, Early detection of parametric roll resonance on container ships. IEEE Trans. Control Syst. Technol. **21**(2), 489–503 (2013)
116. M. Gálvez-Carrillo, M. Kinnaert, Sensor fault detection and isolation in doubly-fed induction generators accounting for parameter variations. Renew. Energy **36**, 1447–1457 (2011)
117. Z. Gao, P.J. Antsaklis, Stability of the pseudo-inverse method for reconfigurable control systems. Int. J. Control **53**, 717–729 (1991)
118. P. Garg, S. Essakiappan, H.S. Krishnamoorthy, P.N. Enjeti, A fault-tolerant three-phase adjustable speed drive topology with active common-mode voltage suppression. IEEE Trans. Power Electron. **30**(5), art. no. 6922554, 2828–2839 (2014)
119. J.P. Gauthier, H. Hammouri, S. Othman, A simple observer for nonlinear systems applications to bioreactors. IEEE Trans. **AC-37**, 875–880 (1992)
120. A.L. Gehin, M. Bayart, M. Staroswiecki, Faulty resources management in automation systems, in *IEEE SMC'97*, Orlando (1997)
121. A.L. Gehin, M. Staroswiecki, A formal approach to reconfigurability analysis. Application to the three tank benchmark, in *Proceedings of the European Control Conference*, Karlsruhe (1999)
122. A.-L. Gehin, M. Staroswiecki, From control to supervision. Annu. Rev. Control **25**, 1–11 (2001)
123. S. Genc, S. Lafortune, Distributed diagnosis of discrete-event systems using Petri nets, in *ICATPN* (2003), pp. 316–336
124. J. Gertler, *Fault Detection and Diagnosis in Engineering Systems* (Marcel Dekker, New York, 1998)
125. J. Gertler, Analytical redundancy methods in failure detection and isolation. Control Theory Adv. Technol. **1**(9), 259–285 (1993)
126. J. Gertler, D. Singer, A new structural framework for parity space equation based failure detection and isolation. Automatica **26**, 381–388 (1990)
127. A. Gheorghe, A. Zolghadri, J. Cieslak, P. Goupil, R. Dayre, H.L. Berre, Model-based approaches for fast and robust fault detection in an aircraft control surface servo loop: from theory to flight tests. IEEE Control Syst. Mag. **33**(3), 20–30 (2013)
128. A. Gill, *Introduction to the Theory of Finite-State Machines* (McGraw-Hill, New York, 1962)
129. S.T. Glad, Non-linear state space and input-output descriptions using differential polynomials. *New Trends in Nonlinear Control Theory*. Control and Information Science, vol. 122 (Springer, Berlin, 1989), pp. 182–189
130. K. Glover, L.M. Silverman, Characterisation of structural controllability. IEEE Trans. **AC-21**(4), 534–537 (1976)
131. M. Gondran, M. Minoux, Graphes et algorithmes. *Collection Direction des Etudes et Recherches EDF*, 3rd edn. (Eyrolles, Paris, 1995)
132. R. Gould, *Graph Theory* (Dover Publications Inc., Mineola, 2012)
133. S.F. Graebe, A.L.B. Ahlén, Dynamic transfer among alternative controllers and its relation to anti-windup controller design. IEEE Trans. Control Syst. Technol. **4**, 92–99 (1996)
134. M. Green, D.J.N. Limebeer, *Linear Robust Control* (Prentice-Hall, Englewood Cliffs, 1995)

135. L. Grunske, K. Winter, N. Yatapanage, S. Zafar, P.A. Lindsay, Experience with fault injection experiments for FMEA. Softw. Pract. Exp. **41**(11), 1233–1258 (2011)
136. L. Guo, J. Gao, J. Yang, J. Kang, Criticality evaluation of petrochemical equipment based on fuzzy comprehensive evaluation and a BP neural network. J. Loss Prev. Process Ind. **22**(4), 469–476 (2009)
137. V. Gupta, B. Hassibi, R.M. Murray, A sub-optimal algorithm to synthesize control laws for a network of dynamic agents. Int. J. Control **78**(16), 1302–1313 (2005)
138. F. Gustafsson, *Adaptive Filtering and Change Detection*, 2nd edn. (Wiley, New York, 2000) (2001)
139. W. Hamscher, L. Console, J. de Kleer, *Readings in Model-Based Diagnosis* (Morgan Kaufmann, San Mateo, 1992)
140. T. Hanus, M. Kinnaert, J.L. Henrotte, Conditioning technique, a general anti-windup and bumpless transfer method. Automatica **3**, 729–739 (1987)
141. R. Hanus, Y. Peng, Conditioning technique for controllers with time delays. IEEE Trans. Autom. Control **37**, 689–692 (1992)
142. F. Harary, A graph theoretic approach to matrix inversion by partitioning. Numer. Math. **4**, 128–135 (1962)
143. D. Harel, Statecharts, a visual formalism for complex systems. Sci. Comput. Program. **89**, 231–274 (1987)
144. X. He, Z. Wang, D.H. Zhou, Robust fault detection for networked systems with communication delay and data missing. Automatica **45**, 2634–2639 (2009)
145. G. Heredia, A. Ollero, M. Bejar, R. Mahtani, Sensor and actuator fault detection in small autonomous helicopters. Mechatronics **18**, 90–99 (2008)
146. J.P. Hespanha, P. Naghshtabrizi, Y. Xu, Survey of recent results in networked control systems. Proc. IEEE **95**(1), 138–162 (2007)
147. S.A. Herrin, Maintainability applications using the matrix FMEA technique. IEEE Trans. **R-30**, 212–217 (1981)
148. D.M. Himmelblau, *Fault Detection and Diagnosis in Chemical and Petrochemical Processes* (Elsevier, Amsterdam, 1978)
149. G. Hoblos, M. Staroswiecki, A. Aitouche, Sensor network design for fault tolerant estimation. Int. J. Adapt. Control Signal Process. **18**, 55–72 (2004)
150. J.E. Hopcroft, R.M. Karp, An algorithm for maximum matchings in bipartite graphs. SIAM J. Comput. **2**, 225–231 (1973)
151. R.F. Huang, C.Y. Stangel, Restructurable control using proportional-integral implicit model following. J. Guid. Control Dyn. **13**, 303–309 (1990)
152. A. Ingimundarson, J.M. Bravo, V. Puig, T. Alamo, P. Guerra, Robust fault detection using zonotope-based set-membership consistency test. Int. J. Adapt. Control Signal Process. **23**(4), 311–330 (2009)
153. R. Isermann, Process fault detection based on modelling and estimation methods: a survey. Automatica **20**(4), 387–404 (1984)
154. R. Isermann, Fault diagnosis of machines via parameter estimation and knowledge processing. Automatica **29**(4), 815–836 (1993)
155. R. Isermann, P. Ballé, Trends in the application of model-based fault detection and diagnosis of technical processes. Control Eng. Pract. **5**(5), 709–719 (1997)
156. R. Isermann, *Fault-Diagnosis Systems* (Springer, Berlin, 2006)
157. R. Isermann, *Fault-Diagnosis Applications* (Springer, Berlin, 2011)
158. Y. Iwasaki, H.A. Simon, Causality in device behaviour. Artif. Intell. **29**, 3–32 (1986)
159. R. Izadi-Zamanabadi, P. Amann, M. Blanke, V. Cocquempot, G.L. Gissinger, E.C. Kerrigan, T.F. Lootsma, J.M. Perronne, G. Schreier, *Ship Propulsion Control and Reconfiguration*. in [8], pp. 285–315
160. R. Izadi-Zamanabadi, M. Blanke, S. Katebi, Cheap diagnosis using structural modelling and fuzzy-logic based detection. Control Eng. Pract. **11**(4), 415–421 (2003)
161. J. Isaksson, An on-line threshold selector for for failure detection, in *Proceedings of the International Conference on Fault Diagnosis: TOOLDIAG*, Toulouse (1993), pp. 628–634

162. R. Izadi-Zamanabadi, M. Blanke, Ship propulsion system as a benchmark for fault-tolerant control. Technical report, Control Engineering Department, Aalborg University, Denmark (1998)
163. R. Izadi-Zamanabadi, M. Blanke, A ship propulsion system as a benchmark for fault-tolerant control. Control Eng. Pract. **7**, 227–239 (1999)
164. R. Izadi-Zamanabadi, Fault-tolerant supervisory control—system analysis and logic design. Ph.D. thesis, Department of Control Engineering, Aalborg University, Denmark (1999)
165. P. Jalote, *Fault Tolerance in Distributed Systems* (Prentice-Hall, Englewood Cliffs, 1994)
166. A.H. Jazwinski, *Stochastic Processes and Filtering Theory* (Academic Press, New York, 1970)
167. N.R. Jennings, J.A. Pople, E.H. Mamdani, Designing a reusable coordination module for cooperative industrial control application, in *IEE Proceedings on Control Theory and Applications* (1996), pp. 91–102
168. S. Jiang, Z. Huang, V. Chandra, R. Kumar, A polynomial algorithm for testing diagnosability of discrete-event systems. IEEE Trans. **AC-46**(8), 1318–1321 (2001)
169. B. Jiang, H. Yang, M. Staroswiecki, Supervisory fault tolerant control for a class of uncertain nonlinear systems. Automatica **10**, 2319–2324 (2009)
170. B. Jiang, H. Yang, M. Staroswiecki, Fault recoverability analysis of switched systems. Int. J. Syst. Sci. **43**(3), 535–542 (2012)
171. B. Jiang, H. Yang, M. Staroswiecki, Observer based fault tolerant control for a class of switched nonlinear systems. IEE Control Theory Appl. **1**(5), 1523–1532 (2007)
172. B. Jiang, K. Zhang, M. Staroswiecki, Static output feedback based fault accommodation design for continuous-time dynamic systems. Int. J. Control **84**(2), 412–423 (2011)
173. D. Jung, L. Eriksson, E. Frisk, M. Krysander, Development of misfire detection algorithm using quantitative FDI performance analysis. Control Eng. Pract. **34**, 49–60 (2015)
174. K.S. Kallesøe, Fault detection and isolation in centrifugal pumps. Ph.D. thesis, Department of Control Engineering and Grundfos A/S (2005)
175. D. Karnopp, R.C. Rosenberg, *Systems Dynamics. A Unified Approach* (Wiley Intersciences, New York, 1975)
176. S.M. Kay, *Fundamentals of Statistical Signal Processing: Detection Theory* (Prentice-Hall PTR, Upper Saddle River, 1998)
177. M. Kinnaert, Y. Peng, Residual generation for sensor and actuator fault detection and isolation, a frequency domain approach. Int. J. Control **61**, 1423–1435 (1995)
178. M. Kinnaert, D. Vrancic, E. Denolin, D. Juricic, J. Petrovcic, Model-based fault detection and isolation for agas-liquid separation unit. Control Eng. Pract. **8**, 1273–1283 (2000)
179. C. Keroglou, C.N. Hadjicostis, Detectability in stochastic discrete event systems, in *Workshop on Discrete Event Systems*, Cachan (2014), pp. 27–32
180. T. Knüppel, M. Blanke, J. Østergaard, Fault diagnosis for electrical distribution systems using structural analysis. Int. J. Nonlinear Robust Control **24**, 1446–1465 (2014)
181. M. Kokar, On consistent symbolic representations of general dynamic systems. IEEE Trans. **AC-40**, 1231–1242 (1995)
182. J. Korbicz, J. Koscielny, Z. Kowalczuk, W. Cholewa, *Fault Diagnosis* (Springer, Berlin, 2004)
183. S. Kowalewski, S. Engell, J. Preißig, O. Stursberg, Verification of logic controllers for continuous plants using timed condition/event-system models. Automatica **35**, 505–518 (1999)
184. M. Krysander, Design and analysis of diagnosis systems using structural methods. Ph.D. thesis, Linköping University (June 2006)
185. M. Krysander, E. Frisk, Leakage detection in a fuel evaporative system. Control Eng. Pract. **17**(11), 1273–1279 (2009)
186. M. Krysander, E. Frisk, Sensor placement for fault diagnosis. IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. **38**(6), 1398–1410 (2008)
187. M. Krysander, J. Åslund, M. Nyberg, An efficient algorithm for finding minimal overconstrained subsystems for model-based diagnosis. IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. **38**, 197–206 (2008)
188. V. Kucera, *Analysis and Design of Discrete Linear Control Systems* (Prentice-Hall, London, 1991)

189. H.W. Kuhn, The Hungarian method for the assignment problem. Nav. Res. Logist. Q. **2**, 83–97 (1956)
190. G. Lamperti, M. Zanella, *Diagnosis of Active Systems* (Kluwer Academic Publishers, Dordrecht, 2003)
191. M. Laursen, M. Blanke, D. Düştegör, Fault diagnosis in a water for injection system using enhanced structural isolation. Int. J. Appl. Math. Comput. Sci. **18**(4), 593–603 (2008)
192. J.M. Legg, Computerized approach for matrix-form FMEA. IEEE Trans. **R-27**, 254–257 (1978)
193. A. Leitold, K.M. Hangos, Structural solvability analysis of dynamic process models. Comput. Chem. Eng. **25**, 1633–1646 (2001)
194. K. Lemmer, Diagnose diskret modellierter systeme mit petrinetzen. Ph.D. thesis, Technische Universität Braunschweig (1995)
195. C.T. Lin, Structural controllability. IEEE Trans. **AC-19**, 201–208 (1974)
196. C.T. Lin, System structure and minimal structure controllability. IEEE Trans. **AC-22**, 855–862 (1977)
197. F. Lin, Diagnosability of discrete event systems and its applications. Discret. Event Dyn. Syst. Theory Appl. **4**, 197–212 (1994)
198. F. Lin, W. Wonham, On observability of discrete-event systems. Inf. Sci. **44**, 173–198 (1988)
199. M. Lind, Modeling goals and functions of complex industrial plants. Appl. Artif. Intell. **8**, 259–283 (1994)
200. F. Liu, D. Qiu, H. Xing, Z. Fan, Decentralized diagnosis of stochastic discrete event systems. IEEE Trans. Autom. Control **53**, 535–546 (2008)
201. L. Ljung, *System Identification: Theory for the User* (Prentice-Hall, Englewood Cliffs, 1999)
202. A. Locatelli, N. Schiavoni, Fault hiding and reliable regulation in control systems subject to polynomial exogenous signals. Eur. J. Control **4**, 389–400 (2010)
203. T.F. Lootsma, Observer-based fault detection and isolation for nonlinear systems. Ph.D. thesis, Department of Control Engineering, Aalborg University, Denmark (2001)
204. T. Lorentzen, M. Blanke, Industrial use of structural analysis—a rapid prototyping tool in the public domain, in *Proceedings of the ACD Workshop*, Karlsruhe (2004), pp. 166–171
205. X.C. Lou, A.S. Willsky, G.C. Verghese, Optimally robust redundancy relations for failure detection in uncertain systems. Automatica **22**, 333–344 (1986)
206. L. Lovász, M.D. Plummer, *Matching Theory* (AMS Chelsea Publishing, Providence, 2009)
207. C. Luh, B. Zeigler, Abstracting event-based control models for high autonomy systems. IEEE Trans. **SMC-23**, 42–54 (1993)
208. C.P. Lunau, A reflective architecture for process control applications, in *ECOP'97 Object Oriented Programming*, ed. by M. Aksit, S. Matsuoka (Springer, Berlin, 1997), pp. 170–189
209. J. Lunze, *Automatisierungstechnik*, 3. Aufl. (Oldenbourg-Verlag, München 2012)
210. J. Lunze, Complexity reduction in state observation of stochastic automata, in *Workshop on Discrete Event Systems*, Reims (2004), pp. 349–354
211. J. Lunze, *Ereignisdiskrete Systeme*, 2. Aufl. (Oldenbourg-Verlag, München 2012)
212. J. Lunze, *Künstliche Intelligent für Ingenieure* (Oldenbourg-Verlag, München, 2011)
213. J. Lunze, Control reconfiguration after actuator failures: the generalised virtual actuator, in *IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS)*, Beijing, 2006
214. J. Lunze, Relations between networks of standard automata and networks of I/O automata Göteborg, in *Workshop on Discrete-Event Systems* (2008), pp. 425–430
215. J. Lunze, Determination of distinguishing input sequences for the diagnosis of discrete-event systems, in *Workshop on Dependable Control of Discrete Systems*, Bari (2008)
216. J. Lunze, Fault diagnosis of discretely controlled continuous systems by means of discrete-event models. Discret. Event Dyn. Syst. Theory Appl. **18**(2), 181–210 (2008)
217. J. Lunze, J. Richter, Reconfigurable fault-tolerant control: a tutorial introduction. Eur. J. Control **5**, 359–386 (2008)
218. J. Lunze, J. Schröder, State observation and diagnosis of discrete-event systems described by stochastic automata. Discret. Event Dyn. Syst. Theory Appl. **11**, 319–369 (2001)

219. J. Lunze, J. Schröder, Sensor and actuator fault diagnosis of systems with discrete inputs and outputs. IEEE Trans. **SMC-34**, 1096–1107 (2004)
220. J. Lunze, T. Steffen, Control reconfiguration by means of a virtual actuator, in *IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS)*, Washington (2003)
221. J. Lunze, T. Steffen, Control reconfiguration demonstrated at a two-degrees-of-freedom helicopter model, in *European Control Conference*, Cambridge (2003)
222. J. Lunze, T. Steffen, Control reconfiguration after actuator failures using disturbance decoupling methods. IEEE Trans. **AC-51**(9), 1590–1601 (2006)
223. J.M. Maciejowski, *Predictive Control with Constraints* (Prentice-Hall, Harlow, 2002)
224. C. Maffezzoni, L. Ferrarini, E. Carpenzano, Object oriented models for advanced automation engineering, in *9th Symposium on Information Control in Manufacturing* (1998), pp. 21–31
225. J.F. Magni, P. Mouyon, On residual generation by observer and parity space approaches. IEEE Trans. **AC-39**, 441–447 (1994)
226. M. Mahmoud, J. Jiang, Y. Zhang, *Active Fault Tolerant Control Systems, Stochastic Analysis and Synthesis* (Springer, London, 2003)
227. M.S. Mahmoud, Y. Xia, *Analysis and Synthesis of Fault-Tolerant Control Systems* (Wiley, New York, 2014)
228. R.S. Mangoubi, *Robust Estimation and Failure Detection* (Springer, Berlin, 1998)
229. N. Meskin, K. Khorasani, Actuator fault detection and isolation for a network of unmanned vehicles. IEEE Trans. Autom. Control **54**(4), 835–840 (2009)
230. M. Meyer, J.-M. Le Lann, B. Koehret, M. Enjalbert, Optimal selection of sensor location on a complex plant using a graph oriented approach. Comput. Chem. Eng. **18**, 535–540 (1994)
231. B.C. Moore, Principal component analysis in linear systems: controllability, observability and model reduction. IEEE Trans. **AC-26**, 17–32 (1981)
232. K. Murota, *Systems Analysis by Graphs and Matroïds. Structural Solvability and Controllability* (Springer, Berlin, 1987)
233. J.M. Nahman, *Dependability of Engineering Systems* (Springer, Berlin, 2002)
234. Y. Nakamura, K. Kogiso, K. Sugimoto, Design of a gain switching observer for networked control systems under random delay, in *International Symposium on Mathematical Theory of Networks and Systems*, Kyoto (2006), pp. 213–220
235. J. Neidig, J. Lunze, Decentralised diagnosis of automata networks, in *Proceedings of the 16th IFAC World Congress*, paper Th-A05-TO/4, Prague (2005)
236. J. Neidig, J. Lunze, Unidirectional coordinated diagnosis of automata networks, in *Proceedings of the 17th International Symposium MTNS*, Kyoto (2006)
237. J. Neidig, J. Lunze, Coordinated diagnosis of nondeterministic automata networks, in *Proceedings of the 6th IFAC Safeprocess*, Beijing (2006), pp. 175–180
238. D.T. Nguyen, M. Blanke, Fault-tolerant positioning control for offshore vessels with thruster and mooring actuation. Technical Report, Centre for Autonomous Marine Systems and Operations (AMOS), Norwegian University of Science and Technology (Trondheim, 2014), p. 27
239. R. Nikoukhah, S.L. Campbell, A multi-model approach to failure detection in uncertain sampled-data systems. Eur. J. Control **11**, 1–11 (2005)
240. H. Niemann, A setup for active fault diagnosis. IEEE Trans. Autom. Control **51**(9), 1572–1578 (2006)
241. H. Niemann, J. Stoustrup, An architecture for fault tolerant controllers. Int. J. Control **78**, 1091–1110 (2005)
242. H. Niemann, N.K. Poulsen, Active fault diagnosis in closed-loop systems, in *IFAC World Congress*, Prague (2005)
243. H. Niemann, Dual Youla parameterization. IEE Proc. Control Theory Appl. **150**, 493–497 (2003)
244. I. Nikiforov, A simple recursive algorithm for diagnosis of abrupt changes in signals and systems, in *American Control Conference* (1998), pp. 1938–1942

245. I. Nikiforov, A simple recursive algorithm for diagnosis of abrupt changes in random signals. IEEE Trans. Inf. Theory **46**, 2740–2746 (2000)
246. R. Nikoukhah, Innovation generation in the presence of unknown inputs: application to robust failure detection. Automatica **30**, 1851–1868 (1994)
247. H.H. Niemann, *Robust Control, Fault Diagnosis and Fault-Tolerant Control—A Standard Setup Approach*. Lecture Notes. Automation at Ørsted·DTU (2002)
248. H.H. Niemann, J. Stoustrup, Gain scheduling using the Youla parameterization, in *IEEE Conference on Decision and Control*, Phoenix (1999), pp. 2306–2311
249. H.H. Niemann, J. Stoustrup, Design of fault detectors using $\mathcal{H}_\infty$ optimisation, in *Proceedings of the 39th IEEE Conference on Decision and Control* (2000), pp. 4237–4238
250. H.H. Niemann, J. Stoustrup, Reliable control using the primary and dual Youla parameterization, in *41st IEEE Conference Decision and Control*, Las Vegas (2002), pp. 4353–4358
251. H. Niemann, J. Stoustrup, An architecture for fault tolerant controllers. Int. J. Control **78**, 1091–1110 (2005)
252. Y. Nke, J. Lunze, A fault modeling approach for input, output automata, in *18th IFAC World Congress*, Milano (2011), pp. 8657–8662
253. Y. Nke, J. Lunze, Control reconfiguration based on unfolding of input/output automata, in *8th Symnposium on Fault Detection, Supervision and Safety of Technical Processes*, Mexico (2012), pp. 866–873
254. H. Noura, D. Theilliol, J.-C. Ponsart, A. Chamseddine, *Fault-Tolerant Control Systems. Design and Practical Applications* (Springer, Berlin, 2009)
255. M. Nyberg, Model based fault diagnosis: methods, theory and automotive engine applications. Ph.D. thesis, Linköping University, Department of Electrical Engineering (1999)
256. M. Nyberg, Criterions for detectability and strong detectability of faults in linear systems. Int. J. Control **7**, 490–501 (2002)
257. H. Ohlsson, F. Gustafsson, L. Ljung, S. Boyd, Smoothed state estimate under abrupt changes using sum-of-norms regularization. Automatica **48**, 595–605 (2012)
258. B. Ould Bouamama, A.L. Gehin, M. Staroswiecki, Alarm filtering by component modelling and bond graph approach, in *4th IFAC Workshop on On-line Fault Detection and Supervision in the Chemical Process Industries (CHEMFAS-4)*, Seoul (2001)
259. A.S. Özveren, A. Willsky, Observability of discrete event dynamic systems. IEEE Trans. **AC-35**, 797–806 (1990)
260. P. Panagi, M.M. Polycarpou, Distributed fault accommodation for a class of interconnected nonlinear systems with partial communication. IEEE Trans. Autom. Control **56**(12), 2962–2967 (2011)
261. A. Papoulis, *Probability, Random Variables and Stochastic Processes* (McGraw-Hill, New York, 1965)
262. F. Pasqualetti, A. Bicchi, F. Bullo, Computation in unreliable networks: a system theoretic approach. IEEE Trans. Autom. Control **57**(1), 90–104 (2012)
263. K. Patan, M. Witczak, J. Korbicz, Towards robustness in neural network based fault diagnosis. Int. J. Appl. Math. Comput. Sci. **18**(4), 443–454 (2008)
264. R.J. Patton, Fault-tolerant control: the 1997 situation, in *IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes*, Hull (1997), pp. 1033–1055
265. R.J. Patton, P.M. Frank, R.N. Clark (eds.), *Fault Diagnosis in Dynamic Systems Theory and Application* (Prentice Hall, New York, 1989)
266. R.J. Patton, P.M. Frank, R. Clark (eds.), *Issues of Fault Diagnosis for Dynamical Systems* (Springer, London, 1999)
267. R.J. Patton, C. Kambhampati, A. Casavola, P. Zhang, S. Ding, D. Sauter, A generic strategy for fault-tolerance in control systems distributed over a network. Eur. J. Control **13**, 280–296 (2007)
268. A. Paoli, S. Lafortune, Safe diagnosability for fault-tolerant supervision of discrete-event systems. Automatica **41**, 1335–1347 (2005)
269. A. Paoli, S. Lafortune, On the diagnosability of a class of hierarchical state machines, in *IFAC Symposium SAFEPROCESS*, Beijing (2006), pp. 1357–1362

270. A. Paoli, S. Lafortune, Diagnosability analysis of a class of hierarchical state machines. Discret. Event Dyn. Syst. **18**, 385–413 (2008)
271. Y. Papadopoulos, M. Walker, D. Parker, E. Rüde, R. Hamann, A. Uhlig, U. Grätz, R. Lien, Engineering failure analysis and design optimisation with HiP-HOPS. Eng. Fail. Anal. **18**(2), 590–608 (2011)
272. R.J. Patton, C. Kambhampati, A. Casavola, P. Zhang, S.X. Ding, D. Sauter, Generic strategy for fault-tolerance in control systems distributed over a network. Eur. J. Control **13**, 280–296 (2007)
273. L. Pau, *Failure Diagnosis and Performance Monitoring* (Marcel Dekker, New York, 1981)
274. X. Paynter, *Analysis and Design of Engineering Systems* (MIT Press, Cambridge, 1961)
275. Y. Peng, A. Youssouf, Ph. Arte, M. Kinnaert, A complete procedure for residual generation and evaluation with application to heat exchanger. IEEE Trans. **CST-5**, 542–555 (1997)
276. Y. Peng, D. Vrancic, R. Hanus, Anti-windup, bumpless, and conditioned transfer techniques for PID controllers. IEEE Control Syst. Mag. **10**, 48–57 (1996)
277. L. Portinale, Behavioural petri nets: a model for diagnostic knowledge representation and reasoning. IEEE Trans. **SMC-27**, 184–195 (1997)
278. K.R. Popper, *Conjectures, Refutations* (Routledge and Kegan Paul, London, 1963)
279. M.-I. Brudny, *Karl Popper: Un Philosophe Heureux* (Grasset, Paris, 2002)
280. N.K. Poulsen, H. Niemann, Active fault diagnosis based on stochastic tests. Int. J. Appl. Math. Comput. Sci. **18**(4), 487–496 (2008)
281. W. Qiu, R. Kumar, Decentralized failure diagnosis of discrete event systems. IEEE Trans. **SMC-36**, 384–395 (2006)
282. J. Quevedo, V. Puig, M. Serra, Polynomial approach to design a virtual actuator for a fault tolerant control: application to a PEM fuel cell, in *5-th Workshop on Advanced Control and Diagnosis*, Grenoble (2007)
283. W. Qiu, Q. Wen, R. Kumar, Decentralized diagnosis of event-driven systems for safely reacting to failures. IEEE Trans. Autom. Sci. Eng. **6**, 362–366 (2009)
284. P. Ramadge, W. Wonham, Supervisory control of a class of discrete event processes. SIAM J. Control Optim. **25**, 206–230 (1987)
285. H.E. Rauch, Autonomous control reconfiguration. IEEE Control Syst. Mag. **15**, 37–48 (1995)
286. K.J. Reinschke, *Multivariable Control: A Graph Theoretic Approach* (Springer, Berlin, 1988)
287. R.L.A. Ribeiro, C.B. Jacobinna, E.R.C. da Silva, A.M.N. Lima, Fault-tolerant voltagefed PWM inverter AC motor drive systems. IEEE Trans. Ind. Electron. **51**, 439–446 (2004)
288. J.H. Richter, *Reconfigurable Control of Nonlinear Dynamical Systems: A Fault-Hiding Approach* (Springer, Heidelberg, 2011)
289. J. Richter, T. Schlage, J. Lunze, Control reconfiguration of a thermofluid process by means of a virtual actuator. IET Control Theory Appl. **1**(6), 1606–1620 (2007)
290. J.H. Richter, J. Lunze, T. Schlage, Control reconfiguration after actuator failures by Markov parameter matching. Int. J. Control **81**, 1382–1398 (2008)
291. J.H. Richter, J. Lunze, Reconfigurable control of Hammerstein systems after actuator failures: stability, tracking, and performance. Int. J. Control **83**(8), 1612–1630 (2010)
292. J.H. Richter, J. Lunze, T. Steffen, The relation between the virtual actuator and the dual observer. Eur. J. Control **16**(5), 525–531 (2010)
293. J.H. Richter, W.P.M.H. Heemels, N. van de Wouw, J. Lunze, Reconfigurable control of piecewise affinie systems with actuator and sensor faults: stability and tracking. Automatica **47**, 678–691 (2011)
294. J.H. Richter, M.M. Seron, J.A. De Dona, Virtual actuator for Lure systems with Lipschitz-continuous nonlinearity, in *IFAC Congress*, Milan (2011)
295. I. Roychoudhury, G. Biswas, X. Koutsoukos, Designing distributed diagnosers for complex continuous systems. IEEE Trans. Autom. Sci. Eng. **6**(2), 277–290 (2009)
296. A. Saberi, A.A. Stoorvogel, P. Sannuti, Exact, almost and optimal input decoupled (delayed) observers. Int. J. Control **73**, 552–582 (2000)
297. M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, D. Teneketzis, Diagnosability of discrete event systems. IEEE Trans. **AC-40**, 1555–1575 (1995)

298. M. Sampath, R. Sengupta, S. Lafurtune, K. Sinnamohideen, D. Teneketzis, Failure diagnosis using discrete event models. IEEE Trans. **CST-4**(2), 105–124 (1996)
299. D. Sauter, T. Boukhobza, Robustness against unknown networked induced delays of observer based FDI, in *IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Beijing (2006), pp. 331–336
300. F. Schiller, J. Schröder, Combining qualitative model-based diagnosis and observation with fault-tolerant systems. AI Commun. **12**, 79–98 (1999)
301. C. Schizas, F.J. Evans, A graph theoretic approach to multivariables control system design. Automatica **17**, 371–377 (1981)
302. T. Schlage, J. Lunze, Modelling of networked systems for remote diagnosis, in *Conference on Control and Fault-Tolerant Systems*, Nizza (2010), pp. 795–800
303. M. Schmidt, J. Lunze, Active diagnosis of deterministic I/O automata, in *Dependable Control of Discrete Event Systems*, Hull (2013)
304. M. Schmidt, J. Lunze, Active fault diagnosis of discrete event systems subject to safety constraints, in *Conference on Control and Fault-Tolerant Systems (SysTol)*, Nice (2013), pp. 706–713
305. M. Schmidt, J. Lunze, A framework for active fault-tolerant control of deterministic I/O automata, in *Workshop on Discrete Event Systems*, Paris (2014), pp. 446–452
306. J. Schröder, *Modelling, State Observation and Diagnosis of Quantised Systems* (Springer, Berlin, 2003)
307. J.H. van Schuppen, T. Villa (eds.), *Coordination Control of Distributed Systems*. Series: Lecture Notes in Control and Information Sciences, vol. 456 (Springer, Berlin, 2015)
308. J.K. Scott, R. Findeisen, R.D. Braatz, D.M. Raimondo, Input design for guaranteed fault diagnosis using zonotopes. Automatica **50**, 1580–1589 (2014)
309. A. Seidenberg, *An Elimination Theory for Differential Algebra*. University of California Publications in Mathematics, vol. 3 (University of California Press, Oakland, 1956), pp. 31–65
310. M.M. Seron, J.A. De Dona, Fault tolerant control using virtual actuators and invariant-set based fault detection and identification, in *IEEE Conference on Decision and Control*, Shanghai (2009), pp. 7801–7808
311. M.M. Seron, J.A. De Dona, J.H. Richter, Bank of virtual actuators for fault tolerant control, in *IFAC World Congress*, Milano (2011), pp. 5436–5441
312. E. Semsar-Kazerooni, K. Khorasani, Optimal consensus algorithms for cooperative team of agents subject to partial information. Automatica **44**(11), 2766–2777 (2008)
313. R. Seydou, T. Raissi, A. Zolghadri, D. Efimov, Actuator fault diagnosis for flat systems: a constraint satisfaction approachr. Int. J. Appl. Math. Comput. Sci. **23**(1), 171–181 (2013)
314. I. Shames, A.H. Teixeira, H. Sandberg, K.H. Johansson, Distributed fault detection for interconnected systems. Automatica **47**, 2757–2764 (2011)
315. B. Shen, Z. Wang, Y.S. Hung, Distributed H_∞-consensus filtering in sensor networks with multiple missing measurements: the finite-horizon case. Automatica **46**(10), 1682–1688 (2010)
316. S. Simani, C. Fantuzzi, R.R. Patton, *Model-Based Fault Diagnosis in Dynamic Systems Using Identification Techniques* (Springer, Berlin, 2002)
317. S. Skogestad, I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*, 3rd edn. (Wiley, New York, 2005)
318. T. Söderström, P. Stoica, *System Identification* (Prentice-Hall, Upper Saddle River, 1989)
319. V. Srinivasan, M. Jafari, Fault detection/monitoring using timed petri nets. IEEE Trans. **SMC-23**(4), 1155–1162 (1993)
320. H. Starke, *Abstrakte Automaten* (Deutscher Verlag der Wissenschaften, Berlin, 1969)
321. M. Staroswiecki, On reconfigurability with respect to actuator failures, in *IFAC World Congress*, Barcelona (2002)
322. M. Staroswiecki, Fault tolerant control: the pseudo-inverse method revisited, in *16-th IFAC World Congress*, paper Th-E05-TO/2, Prague (2005)
323. M. Staroswiecki, M. Bayart, *Les Actionneurs Intelligents* (Hermès, Paris, 1994)

324. M. Staroswiecki, M. Bayart, Models and languages for the interoperability of smart instruments. Automatica **32**, 859–873 (1996)
325. M. Staroswiecki, J.P. Cassar, P. Declerck, A structural framework for the design of FDI in large scale industrial plants. In [266]
326. M. Staroswiecki, G. Comtet, Varga: analytical redundancy relations for fault detection and isolation in algebraic dynamical systems. Automatica **37**, 687–699 (2001)
327. M. Staroswiecki B. Jiang, V. Cocquempot, Fault accommodation for nonlinear dynamic systems. IEEE Trans. Autom. Control **51**(9), 1578–1583 (2006)
328. M. Staroswiecki, Robust fault tolerant linear quadratic control based on admissible model matching, in *45th IEEE Conference on Decision and Control*, San Diego (CA), USA (2006), pp. 3506–3511
329. M. Staroswiecki, On fault handling in control systems. Int. J. Control Autom. Syst. Spec. Issue FDI FTC **6**(3), 1–10 (2008)
330. M. Staroswiecki, On reconfiguration-based fault tolerance, in *IEEE MED, Plenary Lecture*, Marrakech, Morocco (2010)
331. M. Staroswiecki A.-L. Gehin, Reconfiguration analysis using generic models. IEEE Trans. Syst. Man Cybern. Part A **38**(3), 575–583 (2008)
332. M. Staroswiecki, D. Berdjag, Evaluation and optimal selection of reliable control design specifications, in *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Barcelona, Spain (2009), pp. 870–875
333. M. Staroswiecki, D. Berdjag, A general fault tolerant linear quadratic control strategy under actuator outages. Int. J. Syst. Sci. **41**(8), 971–985 (2010)
334. M. Staroswiecki, F. Cazaurang, Fault recovery by nominal trajectory tracking, in *American Control Conference*, Seattle (2008)
335. M. Staroswiecki K. Zhang, B. Jiang, Reduced-order observer based fault estimation design for multiple-input multiple-output discrete-time systems. Proc. Inst. Mech. Eng. Part I J. Syst. Control Eng. **226**(1), 101–110 (2012)
336. M. Staroswiecki, K. Zhang, B. Jiang, Analysis and design of robust estimation filter for a class of continuous-time nonlinear systems. Int. J. Syst. Sci. **43**(19), 1958–1968 (2012)
337. A. Rosich, E. Frisk, J. Åslund, R. Sarrate, F. Nejjari, Fault, diagnosis based on causal computations. IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. **42**(2), 371–381 (2012)
338. M. Staroswiecki, A. Moradi Amani, Fault tolerant control of distributed systems by information pattern reconfiguration. Int. J. Adapt. Control Signal Process. (2014). doi:10.1002/ACS.2497
339. G.K. Singh, V. Pant, Analysis of a multiphase induction machine under fault condition in a phase-redundant A.C. drive system. Electr. Mach. Power Syst. **28**, 577–590 (2000)
340. T. Steffen, *Control Reconfiguration of Dynamical System: Linear Approaches and Structural Tests* (Springer, Heidelberg, 2005)
341. D.V. Steward, On an approach to techniques for the analysis of the structure of large systems of equations. SIAM Rev. **4**, 321–342 (1962)
342. F. Stoican, S. Olaru, *Set-Theoretic Fault-Tolerant Control in Multisensor Systems* (Wiley, New York, 2013)
343. A.A. Stoorvogel, H.H. Niemann, A. Saberi, P. Sannuti, Optimal fault signal estimation. Int. J. Robust Nonlinear Control **12**, 697–727 (2002)
344. J. Stoustrup, M.J. Grimble, Integrating control and fault diagnosis: a separation result, in *IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, Hull (1997), pp. 323–328
345. J. Stoustrup, M.J. Grimble, H.H. Niemann, Design of integrated systems for control and detection of actuator/sensor faults. Sens. Rev. **17**, 157–168 (1997)
346. J. Stoustrup, H.H. Niemann, Fault tolerant feedback control using the Youla parameterization, in *European Control Conference*, Porto (2001)
347. O. Stursberg, S. Kowalewski, S. Engell, Generating timed discrete models of continuous systems, in *Proceedings of the 2nd MATHMOD Conference*, Vienna (1997), pp. 203–209

348. R. Su, Distributed diagnosis for discrete-event systems. Ph.D. thesis, University of Toronto (2004)
349. R. Su, W.M. Wonham, Global and local consistencies in distributed fault diagnosis for discrete-event systems. IEEE Trans. **AC-50**, 1923–1935 (2005)
350. C. Sundstr, E. Frisk, L. Nielsen, Selecting and utilizing sequential residual generators in FDI applied to hybrid vehicles. IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. **44**(2), 172–185 (2014)
351. P. Supavatanakul, J. Lunze, V. Puig, J. Quevedo, Diagnosis of timed automata: theory and application to the DAMADICS actuator benchmark problem. Control Eng. Pract. **14**, 609–619 (2006)
352. C. Svärd, M. Nyberg, Residual generators for fault diagnosis using computation sequences with mixed causality applied to automotive systems. IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. **40**(6), 1310–1328 (2010)
353. C. Svärd, M. Nyberg, E. Frisk, M. Krysander, Data-driven and adaptive statistical residual evaluation for fault detection with an automotive application. Mech. Syst. Signal Process. **45**(1), 170–192 (2014)
354. C. Svärd, M. Nyberg, E. Frisk, Realizability constrained selection of residual generators for fault diagnosis with an automotive engine application. IEEE Trans. Syst. Man Cybern. Syst. **43**(6), 1354–1369 (2013)
355. S.M. Tabatabaeipour, Active fault detection and isolation of discrete-time linear time-varying systems: a set-membership approach. Int. J. Syst. Sci. **46**(11), 1917–1933 (2014)
356. R.M. Tallam, T.G. Habetler, R.G. Harley, Transient model for induction machines with stator winding turn faults. IEEE Trans. Ind. Appl. **38**, 632–637 (2003)
357. R.M. Tallam, T.G. Habetler, R.G. Harley, Stator winding turn-fault detection for closed-loop induction motor drives. IEEE Trans. Ind. Appl. **39**, 720–724 (2003)
358. A. Tantawy, X. Koutsoukos, G. Biswas, Aircraft power generators: hybrid modeling and simulation for fault detection. IEEE Trans. Aerosp. Electron. Syst. **48**, 552–571 (2012)
359. T.T. Tay, I.M.Y. Mareels, J.B. Moore, *High Performance Control* (Birkhäuser, Basel, 1997)
360. A.H. Teixeira, I. Shames, H. Sandberg, K.H. Johansson, Distributed fault detection and isolation resilient to network model uncertainties. IEEE Trans. Cybern. **44**(11), 2024–2037 (2014)
361. J. Thoma, B. Ould Bouamama, *Modelling and Simulation in Thermal and Chemical Engineering: A Bond Graph Approach* (Springer, Berlin, 2000)
362. J.U. Thoma, *Modern Oilhydraulic Engineering* (Trade and Technical Press Ltd, Morden, 1971)
363. L. Thomas, L. Lambolais, R. Lesiour, C. André, M. Bayart, C. Choukair, Architectural techniques for the description and validation of distributed real-time systems, in *2nd IEEE International Symposium on Object oriented Real-Time Distributed Computing (ISORC'99)* (1999), pp. 323–331
364. J.S. Thomsen, A fault tolerant electronic steering system for a fork lift truck. Internal report, Aalborg University and Danfoss A/S (2000)
365. J.S. Thomsen, M. Blanke, Fault-tolerant electrical steering for warehouse trucks, in *IEEE IECON'06: 32nd Annual Conference of the IEEE Industrial Electronics Society*, November 2006 (submitted)
366. D. Thorsley, D. Teneketzis, Diagnosability of stochastic automata. IEEE Trans. Autom. Control **50**, 476–492 (2005)
367. C. Thybo, Fault-Tolerant Control of Inverter Controlled Induction Motors. Ph.D. thesis, Aalborg University, 2000
368. C. Thybo, M. Blanke, Industrial cost-benefit assessment for fault-tolerant control systems, in *Proceedings of the IEE Conference Control*, Swansea (1998), pp. 1151–1156
369. Y. Tipsuwan, M.-Y. Chow, Control methodologies in networked control systems. Control Eng. Pract. **11**, 1099–1111 (2003)
370. L. Trave-Massuyes, T. Escobet, X. Olive, Diagnosability analysis based on component-supported analytical redundancy relations. IEEE Trans. Syst. Man Cybern. **36**, 1146–1160 (2006)

371. J. Unger, A. Kröner, W. Marquardt, Structural analysis of differential-algebraic equation systems—theory and applications. Comput. Chem. Eng. **19**, 867–882 (1995)
372. M. Ungermann, J. Lunze, D. Schwarzmann, Model-based test signal generation for service diagnosis of automotive systems, in *IFAC Symposium on Advances in Automotive Control*, München (2010), paper MA3.1
373. M. Ungermann, J. Lunze, D. Schwarzmann, Test signal generation for service diagnosis based on local structural properties. Int. J. Appl. Math. Comput. Sci. **22**(1), 55–65 (2012)
374. Varga, A., A numerically reliable approach for the synthesis of periodic fdi filters, in *Proceedings of the IFAC SAFEPROCESS 2012* (2012)
375. R.J. Veillette, J.V. Medani, W.R. Perkins, Design of reliable control systems. IEEE Trans. **AC-37**, 290–304 (1992)
376. N. Viswanadham, J.H. Taylor, E.C. Luce, A frequency-domain approach to failure detection and isolation with application to GE-21 turbine engine control system. Control-Theory Adv. Technol. **3**, 45–72 (1987)
377. E. Walter, L. Pronzato, *Identification of Parametric Models from Experimental Data* (Springer, Berlin, 1997)
378. Y. Wang, H. Ye, S.X. Ding, G. Wanga, D. Zhou, Residual generation and evaluation of networked control systems subject to random packet dropout. Automatica **45**, 2427–2434 (2009)
379. Y.Q. Wang, S.X. Ding, P. Zhang, W. Li, H. Ye, G.Z. Wang, Fault detection of networked control systems with packet dropout *IFAC World Congress*, Seoul (2008), pp. 8884–8889
380. Y. Wang, H. Ye, S.X. Ding, Y. Cheng, P. Zhang, G. Wang, Fault detection of networked control systems with limited communication. Int. J. Control **82**, 1344–1356 (2009)
381. Y. Wang, T.-S. Yoo, S. Lafortune, Diagnosis of discrete event systems using decentralized architectures. Discret. Event Dyn. Syst. **17**(2), 233–263 (2007)
382. W. Winston, *Operations Research: Applications and Algorithms*, 4th edn. (Duxbury Press, Boston, 2003)
383. M. Witczak, *Fault Diagnosis and Fault-Tolerant Control Strategies for Non-Linear Systems, Analytical and Soft Computing Approaches* (Springer, Berlin, 2013)
384. J. Willems, Paradigms and puzzles in the theory of dynamic systems. IEEE Trans. **AC-36**, 259–294 (1991)
385. A. Willersrud, M. Blanke, L. Imsland, A. Pavlov, Fault diagnosis of downhole drilling incidents using adaptive observers and statistical change detection. J. Process Control **30**, 90–103 (2015)
386. A. Willersrud, M. Blanke, L. Imsland, A. Pavlov, Drillstring washout diagnosis using friction estimation and statistical change detection. IEEE Trans. Control Syst. Technol. (2015). doi:10.1109/TCST.2015.2394243
387. A. Willersrud, M. Blanke, L. Imsland, Incident detection and isolation in drilling using analytical redundancy relations. Control Eng. Pract. **41**, 1–12 (2015)
388. W.M. Wonham, A control theory for discrete-event system, in *Advanced Computing Concepts and Techniques in Control Engineering*, ed. by M.J. Denham, A.J. Laub (Springer, Berlin, 1988), pp. 129–169
389. N.E. Wu, Coverage in fault-tolerant control. Automatica **40**, 537–548 (2004)
390. N.E. Wu, T.J. Chen, Reliability prediction for self-repairing flight control systems, in *Proceedings of the 35th IEEE Conference on Desicion and Control*, Kobe (1996)
391. N.E. Wu, G.J. Klir, Optimal redundancy management in reconfigurable control systems based on normalised nonspecificity. Int. J. Syst. Sci. **31**, 797–808 (2000)
392. N.E. Wu, K. Zhou, G. Salomon, Reconfigurability in linear time-invariant systems. Automatica **36**, 1767–1771 (2000)
393. N.E. Wu, Z. Zhou, Detection, estimation and accommodation on loss of control effectiveness. Int. J. Adapt. Control Signal Process. **14**, 175–195 (2000)
394. N.E. Wu, S. Thavamani, Y.M. Zhang, M. Blanke, Sensor fault masking of a ship propulsion system. Control Eng. Pract. **14**, 1337–1345 (2006)

395. A. Zolghadri, D. Henry, J. Cieslak, D. Efimov, P. Goupil, *Fault Diagnosis and Fault-Tolerant Control and Guidance for Aerospace Vehicles: From Theory to Application* (Springer, Berlin, 2013)
396. X. Zhang, T. Parisini, M.M. Polycarpou, Adaptive fault-tolerant control of nonlinear uncertain systems: an information-based diagnostic approach. IEEE Trans. Autom. Control **AC-49**, 1259–1274 (2004)
397. J. Yamé, M. Kinnaert, Parameterization of linear controllers for bumpless switching in multi-controller schemes, in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Providence, Rhode Island (2004)
398. Z. Yang, M. Blanke, M. Verhagen, Robust control mixer method for reconfigurable control design using model matching. Proc. IET Control Theory Appl. **1**, 349–357 (2007)
399. Z. Yang, J. Stoustrup, Design of robust reconfigurable control for parametric and additive faults, in *39th IEEE Conference on Decision and Control*, Sydney (2000), pp. 4132–4137
400. Z. Yang, M. Blanke, The robust control mixer module method for control reconfiguration, in *American Control Conference* (2000)
401. Z. Yang, R. Izadi-Zamanabadi, M. Blanke, *On-line Multiple-model Based Adaptive Control Reconfiguration for a Class of Non-linear Control Systems* (Safeprocess, Budapest, 2000)
402. H. Yang, M. Staroswiecki, B. Jiang, Progressive accommodation of aircraft actuator faults, in *IFAC Symposium Safeprocess*, Beijing (2006)
403. H. Yang, M. Staroswiecki, B. Jiang, Active fault tolerant control based on progressive accommodation. Automatica **43**(12), 2070–2076 (2007)
404. H. Yang, M. Staroswiecki, B. Jiang, J. Liu, Fault tolerant cooperative control for a class of nonlinear multiagent systems. Syst. Control Lett. **60**(4), 271–277 (2011)
405. T.-S. Yoo, S. Lafortune, Polynomial-time verification of diagnosability of partially observed discrete-event systems. IEEE Trans. **AC-47**(9), 1491–1498 (2002)
406. L. Zaccarian, A.R. Teel, A common framework for anti-windup, bumpless transfer and reliable designs. Automatica **38**, 1735–1744 (2002)
407. P. Zhang, S.X. Ding, Fault detection of networked control systems with limitedcommunication, in *IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Beijing (2006), pp. 1135–1140
408. Q. Zhang, M. Basseville, A. Benveniste, Early warning of slight changes in systems. Automatica **30**, 95–113 (1994)
409. Q. Zhang, M. Basseville, A. Benveniste, Fault detection and isolation in nonlineardynamic systems: a combined input-output and local approach. Automatica **34**, 1359–1373 (Benveniste 1998)
410. Y. Zhang, J. Jiang, Bibliographical review on reconfigurable fault-tolerant control systems. Annu. Rev. Control **32**, 229–252 (2008)
411. X. Zhang, Q. Zhang, Distributed fault diagnosis in a class of interconnected nonlinear uncertain systems. Int. J. Control **37**(1), 170–179 (2013)
412. K. Zhou, J.C. Doyle, K. Glover, *Robust and Optimal Control* (Prentice Hall, Upper Saddle River, 1995)
413. T. Zhou, On the controllability and observability of networked dynamic systems. Automatica **52**, 63–75 (2015)
414. A. Zolghadri, Advanced model-based FDIR techniques for aerospace systems: today challenges and opportunities. Prog. Aerosp. Sci. **53**, 18–29 (2012)

Index

M. Blanke et al., *Diagnosis and Fault-Tolerant Control*,
DOI 10.1007/978-3-662-47943-8