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

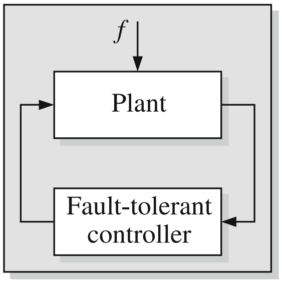


Fig. 1.1 Fault-tolerant system

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 $u : |\mathcal{R} \rightarrow |\mathcal{R}$ and $y : |\mathcal{R} \rightarrow |\mathcal{R}$, 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

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

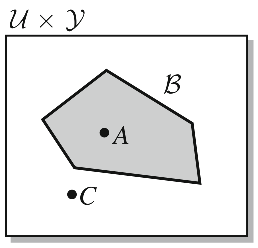


Fig. 1.2 Graphical illustration of the system behaviour

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.

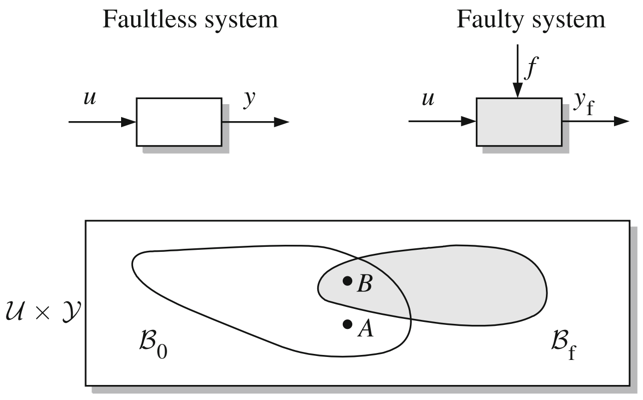


Fig. 1.3 System subject to faults

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 $\mathcal{B}_f$:

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

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

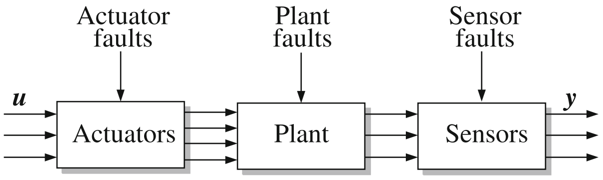


Fig. 1.4 Distinction between actuator faults, plant faults and sensor faults

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.

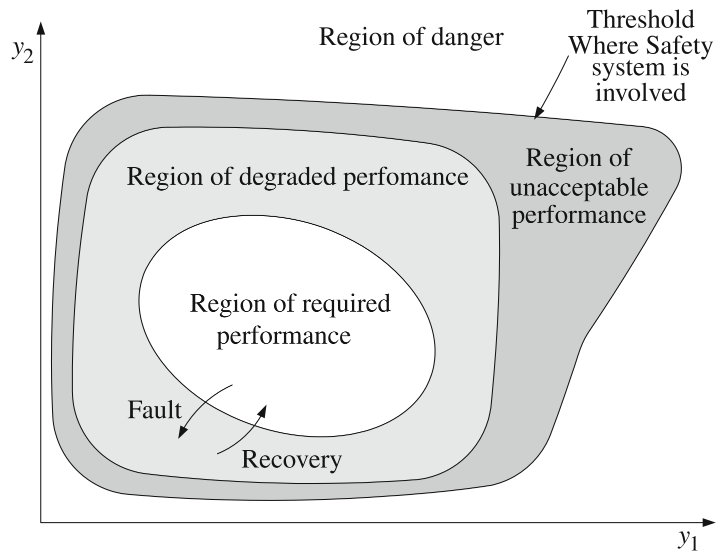

y_2
Region of danger
Threshold Where Safety system is involved
Region of unacceptable performance
Region of degraded perfomance
Region of required performance
Fault
Recovery
y_1

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

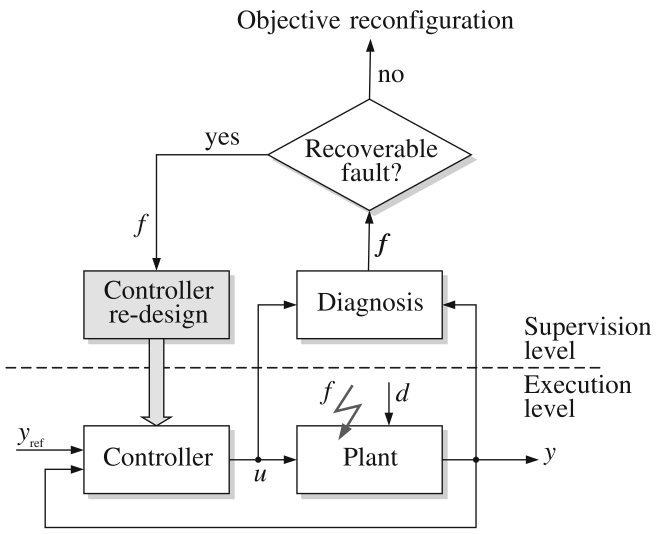


Fig. 1.6 Architecture of fault-tolerant control

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 $\mathcal{F}_{\mathrm{c}}$ 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

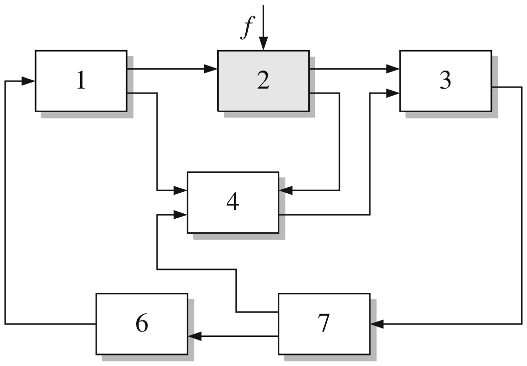


Fig. 1.7 Fault propagation in interconnected systems

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

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

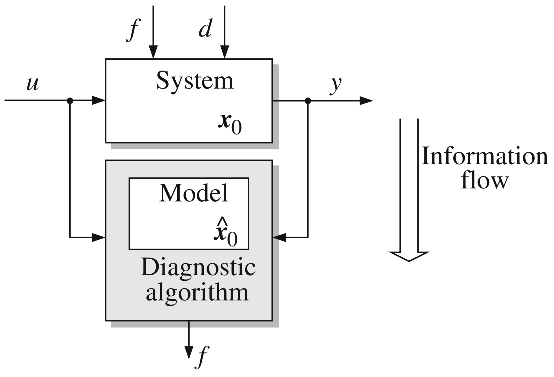
of input and output values measured at discrete-time points $k = 0, 1, \dots, k_\mathrm{e}$ within a given time horizon k_e. It has to solve the following problem:

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

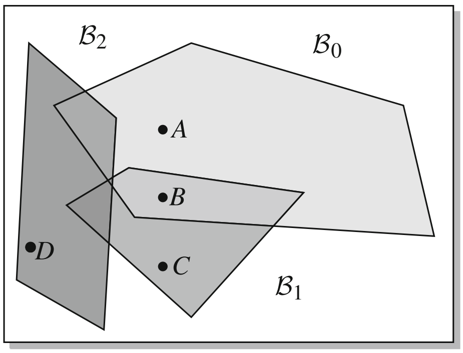


Fig. 1.9 Behaviour of the faultless and the faulty system

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.

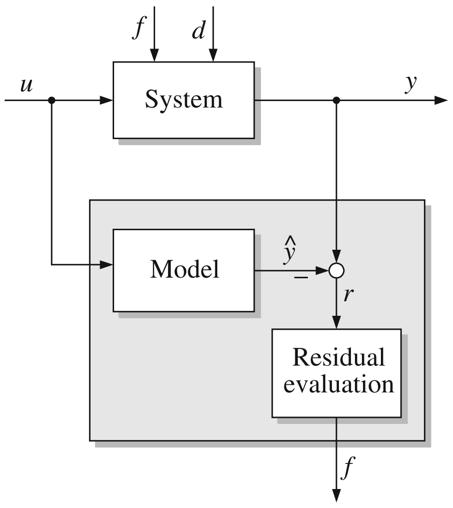


Fig. 1.10 Diagnosis of continuous-variable systems

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 $\mathcal{B}_\mathrm{C}$. The set $\mathcal{B}_\mathrm{C}$ 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 $\mathcal{B}_0 \cap \mathcal{B}_\mathrm{C}$, 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 $\mathcal{B}_\mathrm{spec}$ 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 $\mathcal{B}_\mathrm{spec}$

$$\mathcal{B}_0 \cap \mathcal{B}_\mathrm{C} \subset \mathcal{B}_\mathrm{spec}$$

the closed-loop systems satisfies the performance specifications.

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

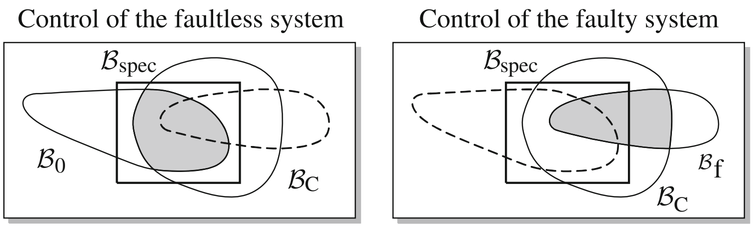

Control of the faultless system
$\mathcal{B}_\mathrm{spec}$
$\mathcal{B}_0$
$\mathcal{B}_\mathrm{C}$
Control of the faulty system
$\mathcal{B}_\mathrm{spec}$
$\mathcal{B}_\mathrm{f}$
$\mathcal{B}_\mathrm{C}$

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 $\mathcal{B}_{\rm f}$ overlaps with the specified behaviour $\mathcal{B}_{\rm spec}$, a controller may be found that restricts this set to a new set $\mathcal{B}_{\rm f} \cap \mathcal{B}_{\rm C}$ which satisfies the relation

$$\mathcal{B}_{\rm f} \cap \mathcal{B}_{\rm C} \subset \mathcal{B}_{\rm spec}$$

(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 $\mathcal{B}_{\rm C}$ 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 $\mathcal{B}_{\rm f}$ does not overlap with $\mathcal{B}_{\rm spec}$. 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 pre-designed 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.

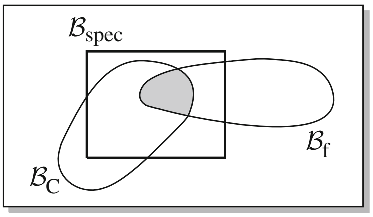


Fig. 1.12 Behavioural representation of fault accommodation

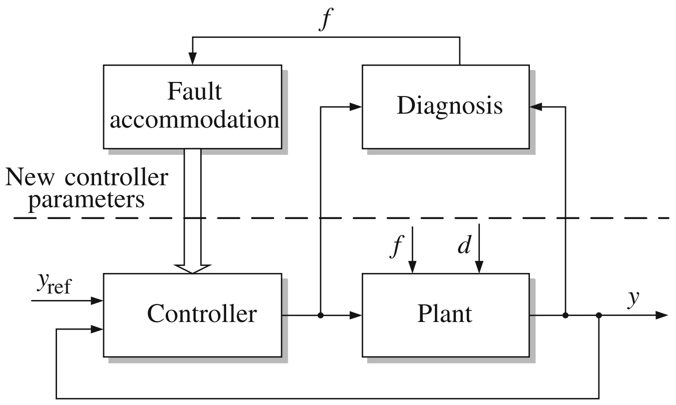


Fig. 1.13 Fault accommodation

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).

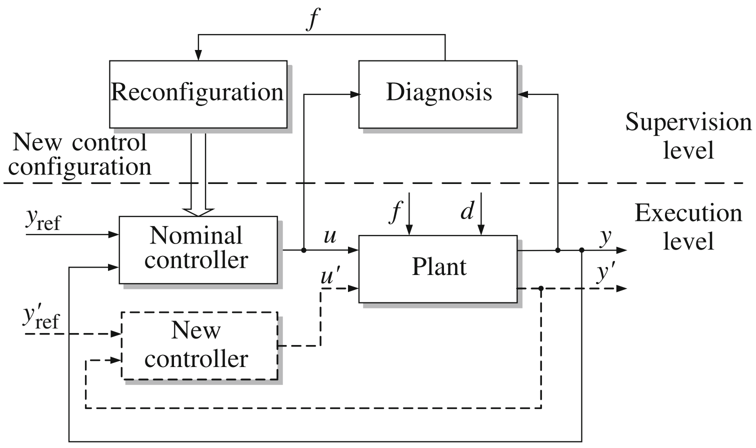


Fig. 1.14 Control reconfiguration

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 $t > t_\mathrm{r}$, 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 $t \in [t_\mathrm{f}, t_\mathrm{r}]$ 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.

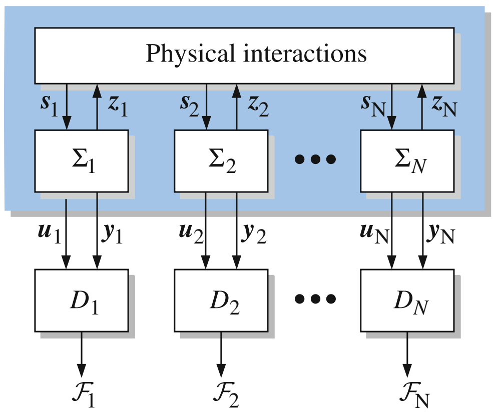


Fig. 1.15 Decentralised diagnosis

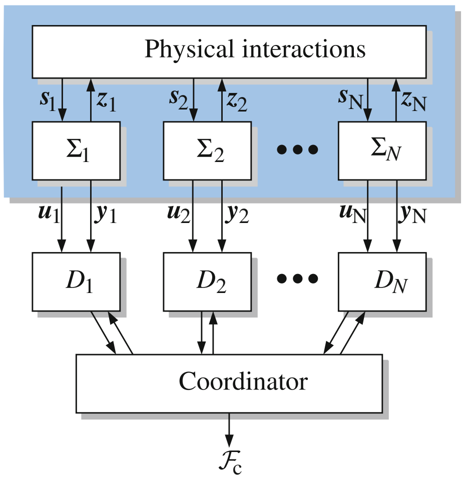


Fig. 1.16 Coordinated diagnosis

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.

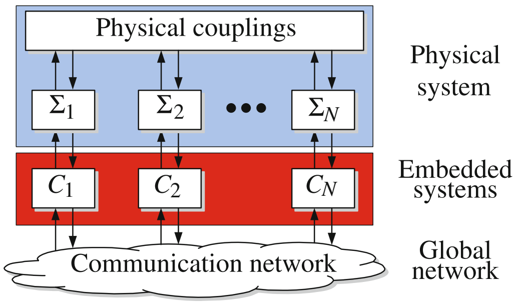


Fig. 1.17 Structure of a cyberphysical system

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

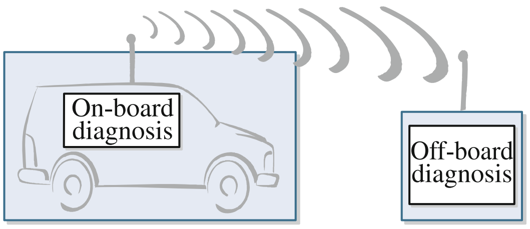


Fig. 1.18 Remote diagnosis

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

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

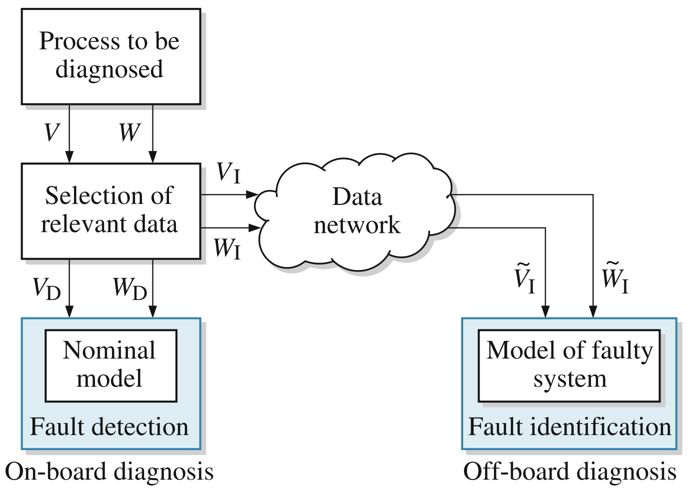


Fig. 1.19 Decomposition of the diagnostic task

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].