Chapter 1

Introduction to diagnosis and fault-tolerant control

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. This can be illustrated by numerous examples. 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 every-day 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 engine has 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 does no longer satisfy 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 causes an ambient temperature increase that eventually stops a reaction or even destroys 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 comes into a certain 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 leads to a degraded system performance or even the loss of the system function.

In large systems, every component has been designed to accomplish a certain function and 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 overall 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. Their aim is 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. The control algorithm adapts to the faulty plant and the overall system satisfies its function again.

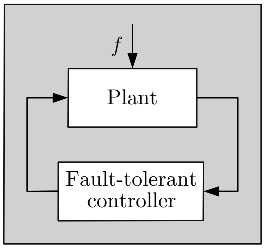

Fig. 1.1. Fault-tolerant system

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 does not only include 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 given performance specifications and, hence, satisfies its function. Fault-tolerant control concerns the situation that the plant is subject to some fault f, which prevents the overall system to satisfy its goal in the future. A fault-tolerant controller has the ability to react on 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 re-design:** 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.

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. In the case of faults, the controller switches to a redundant component. For example, important elements of an aircraft use this principle with a threefold redundancy.

These 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 signal 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 a 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 on-line measurement signals. Then the model is adapted to the faulty situation and the controller is re-designed so that the closed-loop system including the faulty plant satisfies 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 prototypical practical examples. Fault-tolerant control is a new, rapidly developing field. A lot of interesting ideas have already been elaborated, which are presented here.

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 all 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 $y(t)$ 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 $u : \mathbb{R} \rightarrow \mathbb{R}$ and $y : \mathbb{R} \rightarrow \mathbb{R}$, which describe the input or output signals rather than the values of these functions for given points in time.

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 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 $y \neq y_C$.

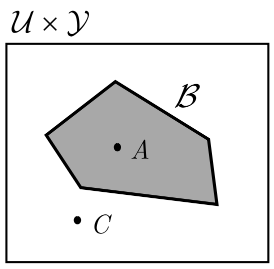

Fig. 1.2. Graphical illustration of the system behaviour

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

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

where k_s is the static gain. The input and the output are elements of the set $\mathbb{R}$ 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 not satisfied, 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

$$U = (u(0),\, u(1),\, u(2), ...,\, u(k_h))$$

of input values that occur at the time instances 0, 1,..., k_h, where k_h denotes the time horizon over which the sequence is considered. Often, k_h is the current time instant, until which the input sequence is stored. Likewise, the output is described by the sequence

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

Consequently, the signal spaces $\mathbb{R}$ used for the static system have to be replaced by $\mathcal{U} = \mathbb{R}^{k_h}$ and $\mathcal{Y} = \mathbb{R}^{k_h}$ 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 is a subset of the cartesian product $\mathcal{U} \times \mathcal{Y} = \mathbb{R}^{k_h} \times \mathbb{R}^{k_h}$

$$\mathcal{B} \subset \mathbb{R}^{k_h} \times \mathbb{R}^{k_h}$$

(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.

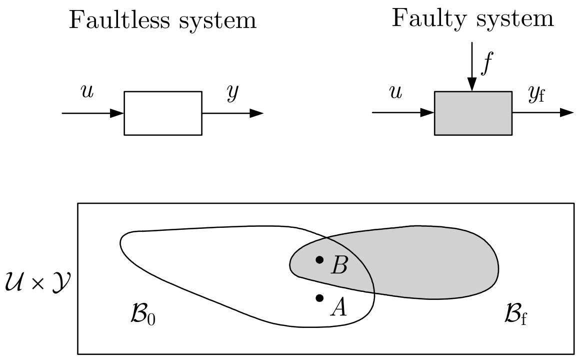

Fig. 1.3. System subject to faults

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 u is applied to the faultless and the faulty system, then both systems answer with the different outputs 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, respectively. 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 misfunction. 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.

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 can also be 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 $\mathcal{B}$:

$$(U, Y) \overset{?}{\in} \mathcal{B}.$$

Faults versus disturbances and model uncertainties. Like faults, disturbances and model uncertainties change the plant behaviour. In order to explain their 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.

The distinction is given by the aim of fault-tolerant control. The faults are those elements which should be detected and whose effects should be removed by remedial actions. Disturbances and model uncertainties are nuisances, which are known to exist but whose effects on the system performance are handled by appropriate measures like filtering or robust design. Control theory has shown that controllers can be designed so as to attenuate disturbances and tolerate model uncertainties up to a certain size. Faults are more severe changes, whose effects on the plant behaviour cannot be suppressed by a fixed controller. Fault-tolerant control aims at changing the control law so as to cancel the effects of the faults or to attenuate them to an acceptable level.

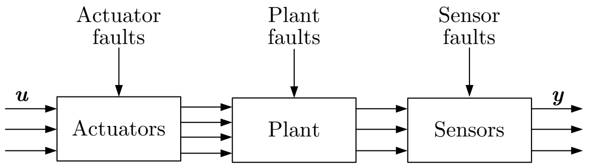

Fig. 1.4. Distinction between actuator faults, plant faults and sensor faults

Classification of faults. The faults are often classified as follows (cf. 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 them. For example, in Section 8.7, observer schemes for sensor and actuator fault diagnosis will be developed and fault-tolerant control will be treated specifically for these cases.

Faults can be distinguished concerning their size and temporal behaviour. Abrupt faults occur, for example, in a break-down 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 a 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 fault from causing a failure at the system level.

1.2.2 Requirements and properties of systems subject to faults

As faults may cause substantial damage on machinery 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 is 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 shut-down, 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.

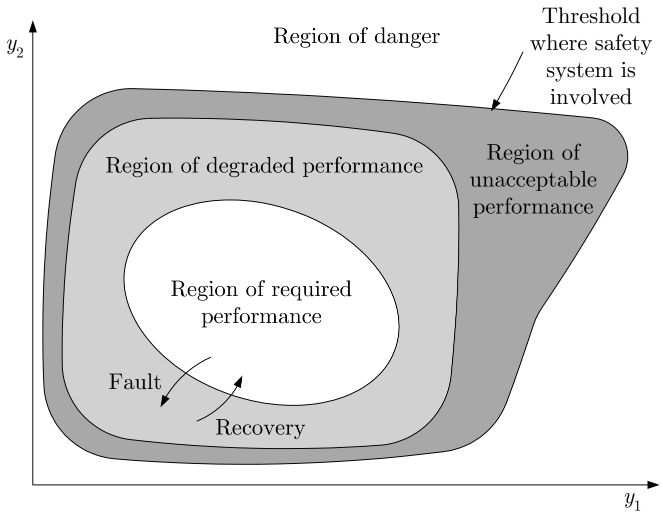
y_2
Region of danger
Threshold where safety system is involved
Region of degraded performance
Region of unacceptable performance
Region of required performance
Fault
Recovery
y_1

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 despite of disturbances and uncertainties of the model used in 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 but 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 re-design" carry out the two steps of fault-tolerant control introduced on page 3.

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

2. The re-design block uses the fault information and adjusts the controller to the faulty situation.

Since the term 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 the figure, all simple arrows represent signals. The connection between the controller re-design 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 re-design of the controller may not only result in new controller parameters, but also in a new control configuration. Then the old and the new controller differ with respect to the input and output signals that they use (cf. Section 1.3.3).

The figure shows that fault-tolerant control extends the usual feedback controller by a supervisor, which includes the diagnostic and the controller re-design blocks. In the faultless case, the nominal controller attenuates the disturbance d and ensures set-point following and other requirements on the closed-loop system. The main

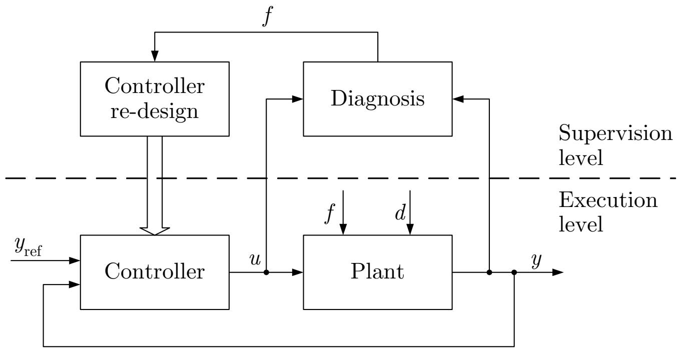

Fig. 1.6. Architecture of fault-tolerant control

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 re-design block adjusts the controller 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 d or model uncertainties bring about uncertainties of the diagnostic results such that instead of the fault f only an approximate fault $\hat{f}$ or a set $\mathcal{F}$ 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, called *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 works suboptimal 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 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 re-design 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 are larger and not restricted to parameter changes and to continous-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.

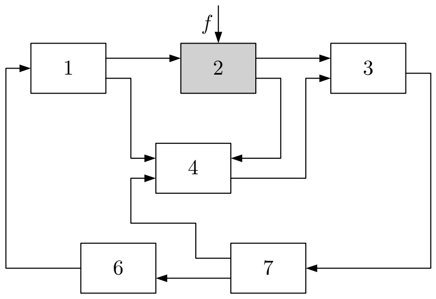

Fig. 1.7. Fault propagation in interconnected systems

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

The propagation of the fault effects through the overall system usually takes time. This time gives the controller of the affected component 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 of the set $\mathcal{F}$ symbolises the faultless case. The diagnostic system obtains the I/O pair $(U,\, Y)$, which consists of the sequences

$$\begin{aligned} U &= (u(0),\, u(1),\, u(2), ...,\, u(k_h)) \\ Y &= (y(0),\, y(1),\, y(2), ...,\, y(k_h)) \end{aligned}$$

of input and output values measured at discrete time points k within a given time horizon k_h. 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.

It should be emphasised that the problem considered here concerns on-line 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 of the information included in a dynamical model and in the time evolution of the signals. Therefore, the term *process diagnosis* is used if these aspects should be emphasised.

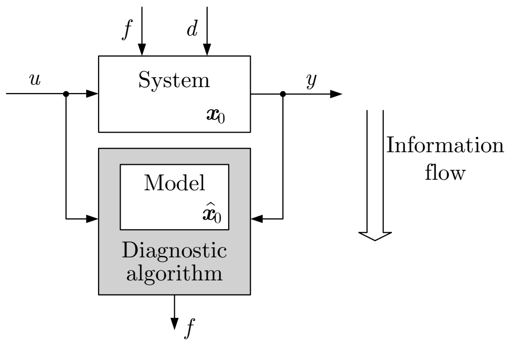

Fig. 1.8. Fault diagnosis

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 sequences. 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 page 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. If the faulty system produces the I/O pair represented by point B, no inconsistency occurs despite of the fault. Hence, the fault is not detected.

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 $(U, Y) \not\in \mathcal{B}$ 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 occur. 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, i.e. there are 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 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

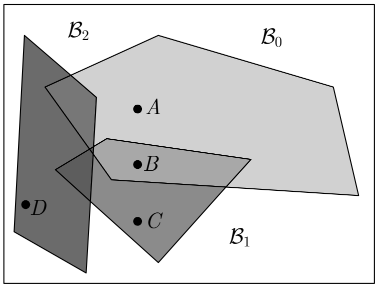

Fig. 1.9. Graphical illustration of the system behaviour

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). The result 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. These are important system properties, which will 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.

- 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 Chapter 8. For continuous-variable systems the usual way of comparison consists in using the difference between the system and the model output in the way explained below.

Diagnosis of continuous-variable systems. Continuous-variable systems, which will be investigated in Chapter 6, 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 residual is evaluated in order to detect, isolate and identify faults.

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

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 $\hat{y}$. This procedure

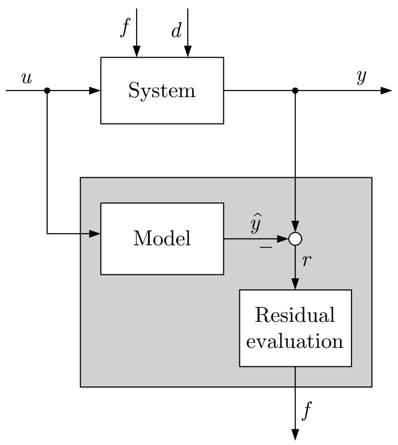

Fig. 1.10. Diagnosis of continuous-variable systems

avoids physical redundancy where more than one sensor is used 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 re-design. 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 in the controller re-design.

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 re-design

Controller re-design 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 despite of the faulty behaviour of the plant.

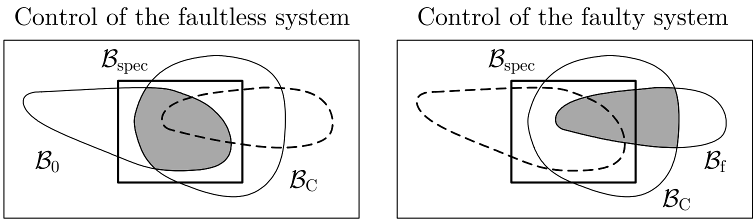

Fig. 1.11. Behaviour of the faultless and faulty closed-loop system

The necessity and aim of the controller re-design 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 the behaviour $\mathcal{B}_{\rm C}$. The set $\mathcal{B}_{\rm C}$ 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 $\mathcal{B}_0 \cap \mathcal{B}_{\rm C}$, 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 $\mathcal{B}_{\rm spec}$ 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 $\mathcal{B}_{\rm spec}$

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

the closed-loop systems satisfies the performance specifications.

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

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

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

(cf. 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

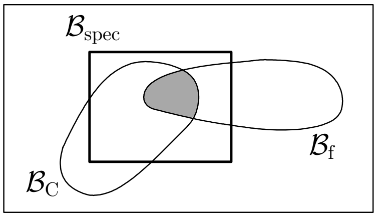

Fig. 1.12. Behavioural representation of fault accommodation

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.

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. 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 re-design method, but a property of the plant subject to faults. 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.

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

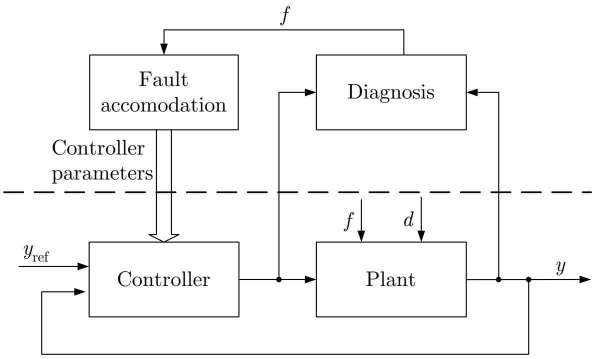

Fig. 1.13. Fault accommodation

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 sequences 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 off-line for a specific fault. The re-design 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 re-design 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.

More general ways of fault accommodation will be explained in Chapter 6.

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 on-line (Fig. 1.14).

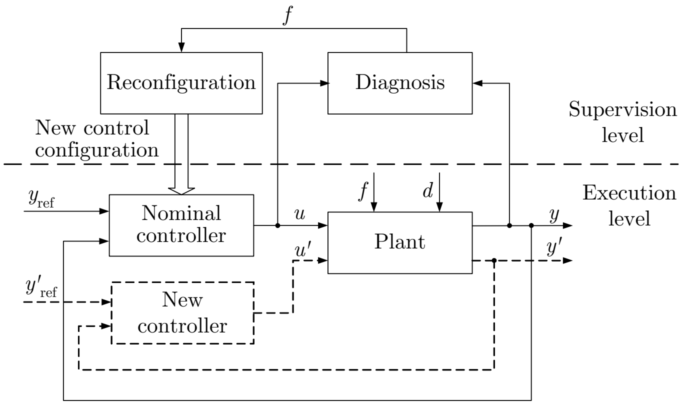

Fig. 1.14. Control reconfiguration

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. Alternative 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. Alternative 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 fail completely, the fault leads to a break-down of the control loop. There is no possibility to adapt the controller by simply changing its parameters to the faulty situation. Instead, alternative 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.

Real-time aspects of fault accommodation and control reconfiguration. Both fault accommodation and control reconfiguration imply the on-line re-design of the controller, which is reminiscent of the well known 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 new restrictions:

- 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 this design problem even if the performance is not optimal.
- Fault accommodation and control reconfiguration have to be done under real-time constraints.

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:

1. Before the fault occurrence, the nominal system is controlled using the nominal control, the control objectives are satisfied.
2. Between fault occurrence and fault recovery, the faulty system is controlled using the nominal control law, and control objectives are in general not satisfied, for example the system may even become unstable.
3. After the fault recovery time, the faulty system is controlled using the accommodated or reconfigured control, the system objectives are satisfied again.

The second point above is critical, and the associated time window should be made as short as possible. Note that this time window occurs due for three reasons:

- Fault detection and isolation delay
- Fault estimation delay
- Delay for the re-design 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 when on-line fault accommodation is used, since the model of the faulty system must be identified for the control algorithm to be accommodated. This delay is avoided if reconfiguration is used, since for reconfiguation it is sufficient to know which component is faulty to switch it off and replace it by another component. Alternative control configurations can be designed off-line, reducing the on-line task to switching among these controllers. Finally, at the recovery time, i.e. once control re-design has been achieved, switching from the nominal to the accommodated or reconfigured control should be done in a bumpless way.

Fault accommodation and control reconfiguration is a recently started subject of fault-tolerant control. There are several promising solutions, which are summarised in this book. In particular, Chapter 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 Chapter 5 shows the redundancies that can be used for reconfiguration. Specific methods for continuous-variable plants are explained in Chapter 7.

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 vs. 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 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 are 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, whose validity 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 re-design 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 re-design tasks are accomplished on a single computer board, which is directly connected to the system to be controlled. All measurement information is 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. In contrast to this, a distributed or a remote systems approach has to be applied, where the fault-tolerant control algorithms and the available information is distributed among different components. These situations, which will be explained in the next sections, have important consequences for the fault-tolerant 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 drop-out of data. These practical circumstances have to be taken into account when elaborating fault-tolerant control algorithms.

1.4.2 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 re-design.

- **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.

- **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 independently from each other.

- **Coordinated diagnosis:** Like in decentralised diagnosis the overall problem is decomposed, but the solutions obtained independently for the subproblems are combined by some coordinator 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 suggest 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 of the overall system.

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 SS_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 these signals are not assumed to be known to the diagnosers. Different approaches have been considered to solve this problem. The sim-

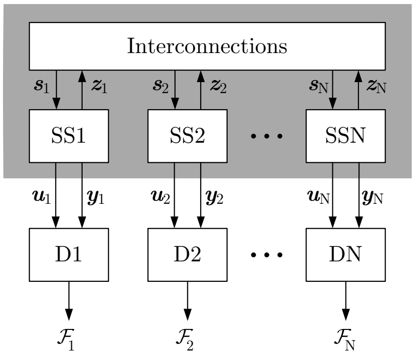

Fig. 1.15. Decentralised diagnosis

plest way is to assume that there is no interaction, which means that the model used by the diagnoser D_i describes the isolated SS_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

$$\mathcal{F}_d = \cup_i \mathcal{F}_i$$

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

$$\mathcal{F}_d \supseteq \mathcal{F}_g.$$

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 from one another, 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 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 $\mathcal{F}_c$ of the overall system. As the coordinator has a model of the interconnections of the subsystems, the overall result $\mathcal{F}_c$ is better that the result of the decentralised diagnosis:

$$\mathcal{F}_c \subseteq \mathcal{F}_d.$$

If the link between the diagnosers and the coordinator is bi-directional, 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

$$\mathcal{F}_c = \mathcal{F}_g.$$

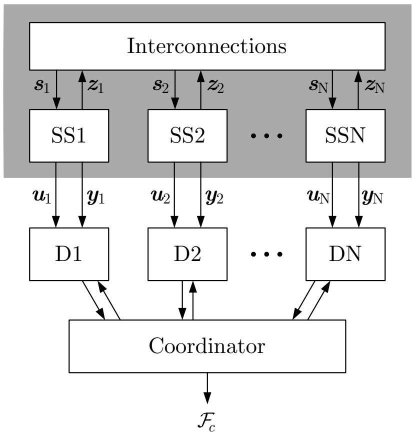

Fig. 1.16. Coordinated diagnosis

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.

1.4.3 Remote diagnosis

Modern data communication networks provide the means for remote supervision and control of technological systems. 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 1.17 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 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 as well as 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.

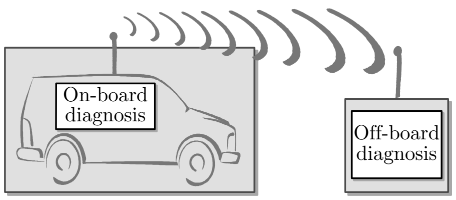

Fig. 1.17. Remote diagnosis

- 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 proposal for the decomposition of the diagnostic task (Fig. 1.18):

- 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 sequences. All available input and output data are represented by the sequences

$$\begin{aligned} V(0...k) &= (v(0), v(1), ..., v(k)) \\ W(0...k) &= (w(0), w(1), ..., w(k)). \end{aligned}$$

Only part of these sequences have to be used for fault detection. That is, some data included in the sequence $V(0...k)$ can be deleted to get the reduced sequence $V_{\rm D}(0...k)$. The same happens with the sequence $W(0...k)$ to get the sequence $W_{\rm D}(0...k)$ used for fault detection.

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

Figure 1.18 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

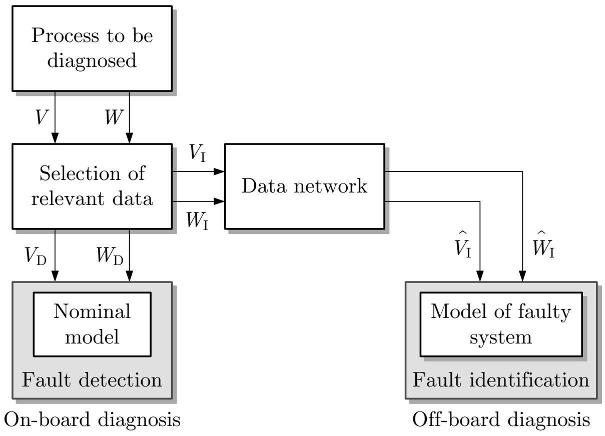

Fig. 1.18. Decomposition of the diagnostic task

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 bi-directional 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 component. 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. 1.18 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 accomodation 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 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, 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 Chapters 4 through 9 deal with another kind of models.

- **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 re-design, 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.

- **Methods for fault detection and isolation.**
  The diagnostic methods explained in this book have been selected for the purpose of fault-tolerant control. Only those methods are described, which do not only detect, but also isolate or identify the fault. The structural analysis for finding analytical redundancy relations for fault detection and isolation in continuous-variable systems explained in Chapter 5 and the diagnostic methods for discrete-event systems described in Chapter 8 provide novel means of fault identification, which have not yet been published in a monograph or textbook.

- **Re-design 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 Chapters 4 and 5.

The book is structured into three parts. The introductory part (Chapters 1 – 3) describes the main problems of fault-tolerant control, illustrates these problems by two examples and give a survey over the alternative models of the plant. The two running examples will be used throughout the book to illustrate the ideas and methods developed.

The main part of the book (Chapters 4 – 9) explains the methods for fault diagnosis and fault-tolerant control. It is structured according to the model types used, beginning with component-based analysis that uses an architectural model of the dynamical system, proceeding with structural analysis based on a structure graph, with diagnosing and controlling continuous-variable systems described by differential or difference equations and transfer functions, with discrete-event systems represented by stochastic automata, and finally considering quantised systems, where the methods for continuous-variable and discrete-event systems have to be combined. The last part (Chapter 10) describes applications.

In more detail, The **main ideas of fault diagnosis and fault-tolerant control** are explained in Chapter 1 in a verbal form before these ideas are developed in a rigorous mathematical form in the later chapters. It also takes a look at the field from the viewpoint of a practising engineer.

Chapter 2 illustrates the problems to be solved by **two simple examples**. The two-tank system and the ship autopilot will be used very often in later chapters for illustration of the methods developed. They are described here in a separate chapter to facilitate cross-references.

Chapter 3 gives a **survey of the models** used. It compares the viewpoints and structures used to represent dynamical systems by different mathematical means for the different purposes of the later chapters.

The treatment of fault diagnosis and fault-tolerant control starts in Chapter 4 with the most abstract model. Based on an architectural model of the plant, failure-modes and effect analysis (FMEA) and **fault propagation analysis** use the fact that the propagation of faults through a system can be analysed if all potential faults and their effects on the system components are known. No complete analytical model is necessary, but it is sufficient to consider the interconnections among the system components. This method is extended to find out, as a preliminary step of fault-tolerant control, where the migration of the fault can be stopped.

Chapter 5 describes the **structural analysis of dynamical systems** which is based on a structure graph that shows which signals are related to each other by the model equations. In the bi-partite graph, the nodes symbolise the model equations on the one hand and the signals on the other hand. By matching techniques, the model equations are associated to unknown signals in order to find out how the unknown variables can be determined by the corresponding model equation. This step discloses the redundancies included in the set of constraints and measurements used for diagnosis and the possibilities to recover from faults. The latter provides the basis for the investigation of the system reconfigurability and leads to tests of the existence of fault-tolerant controllers.

Fault diagnosis and fault-tolerant control of continuous-variable systems are investigated in Chapters 6 and 7. The diagnosis of these systems can be decomposed into residual generation and residual evaluation. The methods for both steps are explained in a deterministic and a stochastic system environment. Chapter 7 shows how the diagnostic result can be used for fault-tolerant control. This chapter introduces several methods for fault accommodation and control reconfiguration.

Chapter 8 is devoted to **fault diagnosis of discrete-event systems** described by non-deterministic or stochastic automata. After the explanation of the model used, the state observation problem is solved. Then the diagnostic task is transformed into an observation task by interpreting the fault as a state of the fault model and by combining the fault model with the plant model to a unique automaton. Methods for investigating the diagnosability of discrete-event systems are also explained. For sensor and actuator diagnosis, specific methods are described which are similar to the dedicated or generalised observer schemes known from continuous systems. These schemes are extended to automatically substitute faulty sensors or actuators and, hence, provide a reconfiguration of the controller.

The combination of ideas and methods for continuous-variable and discrete-event systems is necessary, if **quantised systems** are concerned. In Chapter 9 a purely discrete-event description of a quantised system is built by abstracting the behavioural relation of a stochastic automaton from the hybrid model of the quantised system, which consists of the differential equation describing the continuous-variable part and the inequalities representing the quantisers. The properties of this abstract model are investigated and it is shown that the diagnostic results obtained for the model also hold for the quantised system. Finally, it is shown how the diagnostic algorithm of such systems can be reconfigured to maintain its function after a fault has occurred.

In Chapter 10 a benchmark problem and four **industrial problems** are solved. These problems include the fault-tolerant control of a chemical process, a ship propulsion system, a steam generator and an electrical steering-by-wire system. Experiments show that the methods described in the book can be successfully applied.

In summary, this book covers the whole 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 method for fault accomodation and control reconfiguration. With this scope, it includes much material that was not published in a unified form. This particularly concerns structural methods of fault-tolerant control, diagnosis of discrete-event systems, and modelling and fault-tolerant control of quantised 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 recently 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 recent monographs are restricted to a relatively narrow and advanced research topic, but describe this in much detail like the book [168] on robust estimation and its use for fault detection, [34] on active fault detection by the design of proper auxiliary signals, [121] on diagnosis of a specific class of discrete-event systems called active systems, and [167] on active fault-tolerant control systems with Markovian parameters.

The textbooks [43], [78] 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 book [98] gives a large survey of various methods for fault diagnosis and design of fault-tolerant structures. 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 monograph [116] which also briefly introduces analytical and knowledge-based methods. This book also explains diagnostic methods based on fuzzy set theory and artificial neural networks.

1.6 Bibliographical notes

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 [202]. The interested reader may also consult [203] for an intellectual biography.

Fault detection has been the subject of long research with [94] and [196] as two of the earliest descriptions of the field. [78] and [195] 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 [219] gives a thorough introduction into fault diagnosis by means of identification techniques. [173] describes the main methods for evaluating the dependability of engineering systems in a broader perspective.

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 [86], [139]. In Chapter 9 of the monograph [3] 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 [263]. Several attempts have been made to combine diagnostic methods elaborated in control engineering and Artificial Intelligence [13].

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 off-line for different fault situations. Surveys of these methods are given in [146], [193] and [206].

The systematic treatment of fault-tolerant control by reconfiguring the control loop concerned aerospace examples with [96] and [70] being two of the earliest papers that used model-matching or a pseudo-inverse 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 [106], [151]. Solutions to these problems which have been obtained by alternative methods are described in Chapters 12 and 13 of the monograph [3].

The different structures of centralised, decentralised and coordinated diagnosis have been discussed for discrete-event systems in [152], the main issues of remote diagnosis in [69].