M. Blanke
M. Kinnaert
J. Lunze
M. Staroswiecki

Diagnosis and Fault-Tolerant Control

Second Edition

Springer

Mogens Blanke · Michel Kinnaert · Jan Lunze · Marcel Staroswiecki

Diagnosis and Fault-Tolerant Control

Mogens Blanke · Michel Kinnaert
Jan Lunze · Marcel Staroswiecki

Diagnosis and Fault-Tolerant Control

With contributions by Jochen Schröder

2nd Edition

With 270 Figures, 121 Examples, 5 Application Studies, and 36 Exercises

Springer

Prof. Dr. Mogens Blanke
Technical University of Denmark
Section of Automation at Ørsted · DTU
2800 Lyngby, Denmark
and
Norwegian University
of Science and Technology
7491 Trondheim, Norway
mb@oersted.dtu.dk

Prof. Dr. Michel Kinnaert
Université Libre de Bruxelles
Laboratoire d'Automatique
CP 165
50 Ave. F.D. Roosevelt
1050 Bruxelles, Belgium
Kinnaert@labauto.ulb.ac.be

Prof. Dr.-Ing. Jan Lunze
Ruhr-Universität Bochum
Lehrstuhl für Automatisierungstechnik
und Prozessinformatik
44780 Bochum, Germany
Lunze@atp.rub.de

Prof. Dr. Marcel Staroswiecki
Université Lille I
Ecole Polytechnique Universitaire de Lille
59655 Villeneuve d'Ascq Cedex, France
and
Ecole Normale Supérieure de Cachan
94235 Cachan, France
marcel.staroswiecki@univ-lille1.fr

Library of Congress Control Number: 2006927814

ISBN-10 3-540-35652-5 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-35652-3 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media
springer.com

Printed in Germany

Typesetting: Digital data supplied by data
Final processing: PTP-Berlin Protago-TEX-Production GmbH, Berlin (www.ptp-berlin.com)
Cover-Design: estudio Calamar, Frido Steinen-Broo, Spain
Printed on acid-free paper 62/3141/Yu – 5 4 3 2 1 0

Preface

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

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

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

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

The solution of these problems necessitates new analysis tasks like the test of the reconfigurability of the system or the search for redundant sensors and actuators,

which can replace faulty components. The aim is to close the control loop after a break-down of such components has brought the controller out of operation. With respect to fault accommodation and control reconfiguration, the book presents the current state of the art.

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

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

Structure of the book. The book is organised according to the different models used. As each of these models requires its own mathematical background and the methods based on these models follow different lines of thinking, the book cannot present the methods in all details. The aim is to give the readers a broad view on the field and provide them with bibliographical notes for further reading. A further reason for the different depth with which the chapters tackle the fault-tolerant control problems is given by the current status of research. Whereas for continuous-variable systems, fault diagnostic and fault-tolerant control methods have been developed for long, discrete-event systems became the subject of substantial research with respect to the topic of this book only recently. Hence, this field has not yet reached the same maturity as for continuous systems.

The chapters start with a verbal explanation of the main ideas and illustrate all results by **two running examples** that concern a simple tank system and a ship autopilot. The common use of these examples in all chapters makes a comparison of the alternative approaches very easy. It is the knowledge of the aims, models, ideas and methods used for different problems of fault diagnosis and fault-tolerant control that enables a control engineer to tackle practical problems under the circumstances

given by the particular field of application. To introduce him to this knowledge is the primary aim of this book.

Level of the book. The intended readers of the book are graduate students of control, electrical, mechanical or process engineering with knowledge in control, continuous system theory and filtering. The authors use the text in regular courses at the Université Lille 1, at the Universitè Libre de Bruxelles, and at the universities of Bochum, Lyngby and Trondheim.

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

The book is self-contained with a review on some basics in the appendices. Its understanding requires knowledge about dynamical systems and controller design. Many figures illustrate the problems, methods and results in an intuitive way and make the interpretation of the rigorous mathematical treatment easier.

Common research. The idea of the authors to bring together their different views and principles for fault diagnosis and fault-tolerant control originated in the project "Control of Complex Systems (COSY)", which was funded by the European Science Foundation between 1995 and 1999. Some of the material has been collected for a PhD course at Aalborg University (Denmark) held in April 1999. In the later DAMADICS Training and Research Network funded by the European Union beween 2001 and 2004 several chapters of the book have been presented at the summer schools and workshops. The interest of people from industry in this new subject has been attracted in courses organised by the Carl-Cranz-Gesellschaft Oberpfaffenhofen (Germany), where preliminary versions of this book have been used.

The large scope of the book was made possible by the close cooperation and by common research of the four authors together with their PhD students and colleagues. The introductory part (Chapters 1 through 3) and the application examples (Chapter 10) describe ideas and results of these four groups. The presentation of the methods for dealing with the system architecture (Chapter 4) is common work of the groups of MOGENS BLANKE in Aalborg and Lyngby (Denmark) and MARCEL STAROSWIECKI in Lille (France). The part on structural analysis (Chapter 5) introduces the methods developed in Lille. Diagnostic methods for continuous systems have been elaborated by many groups. The presentation of those ideas that can be used in fault-tolerant control (Chapter 6) resulted from common work and teaching

experiences of MOGENS BLANKE, MICHEL KINNAERT (Brussels, Belgium) und MARCEL STAROSWIECKI. Chapter 7 on fault accommodation and control reconfiguration describes ideas of the groups in Lyngby, Bochum, Brussels and Lille. The methods for dealing with discrete-event systems and quantised systems (Chapters 8 and 9) have been elaborated by the group of JAN LUNZE in Hamburg and Bochum (Germany), where particularly JOCHEN SCHRÖDER has made substantial contributions not only to the research work but also to the presentation of the results in this book.

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

Special thanks are due to our former and current PhD students and research associates, particularly to ROOZBEH IZADI-ZAMANABADI, JAKOB STOUSTRUP and JESPER S. THOMSEN (Aalborg), HENRIK NIEMANN and TORSTEN LORENTZEN (Lyngby), JOCHEN SCHRÖDER (Hamburg), THOMAS STEFFEN (Hamburg/Bochum), JÖRG NEIDIG, JAN RICHTER and THORSTEN SCHLAGE (Bochum), and ANNE-LISE GÉHIN and BELKACEM OULD BOUAMAMA (Lille).

Several industrial applications have been accomplished with the interest and help of, among others, CLAUS THYBO (Danfoss) and MARTIN FRITZ (R. Bosch GmbH, Stuttgart). Further, the support from American Power Conversion Denmark A/S to develop tools for structural analysis and from Sauer-Danfoss A/S to develop the fault-tolerant steering-by-wire system introduced in Chapter 10 are gratefully acknowledged.

We are grateful for the valuable help of Ms. KATRIN LUNZE (Stuttgart) for giving the manuscript a uniform layout and of Ms. ANDREA MARSCHALL (Bochum) for drawing many of the figures.

Second edition. This new edition has been extended by a lot of new material, which has appeared in the three-year period after the first edition was finished. Among them is the description of new diagnostic structures like decentralised vs. cooperative diagnosis, and remote diagnosis. The uniform look at continuous and discrete-event systems diagnosis has been deepened. New results on the diagnosis of continuous systems and the reconfiguration of the control loop after sensor or actuator failures have been included. The application examples are extended by a steering-by-wire system and the air path of a diesel engine, both of which include experimental results. The list of references and the bibliographical remarks at the end of all chapters have been up-dated.

The presentation has been improved according to the authors' experience from using this book in their lectures and courses in industry. Many chapters finish with exercises to be used in lectures or for self-repetition of the material.

Lyngby, Brussels, Bochum and Lille M.B., M.K., J.L., M.S.
May 2006

The book homepage at `www.rub.de/atp` → `Books` provides further information such as the description of the COSY benchmark problems of fault-tolerant control, exercises, and files with the figures of this book for their use as slides in lectures.

Contents

Appendices

The authors

Mogens Blanke is professor of automatic control at the Section of Automation at Ørsted·DTU of the Technical University of Denmark and is adjoin professor at the CESOS center of excellence at the Norwegian University of Science and Technology. His research interests comprise autonomous and fault-tolerant systems, fault diagnosis, systems architecture design to obtain desired safety properties, system modelling, identification and control. His experiences result from the development of fault-tolerant design methods for the Danish Ørsted satellite, from the design of several marine automation systems and from the design of fault-tolerant steering-by-wire architectures for vehicles.

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

Jan Lunze is professor of automatic control and head of the Institute of Automation and Computer Control at the Ruhr-Universität Bochum (Germany). His research interests include fault diagnosis and reconfigurable control of discrete-event and hybrid systems, qualitative modelling of dynamical systems, linear control theory with applications in the automotive and process industries, and applications of symbolic information processing to control systems. He is author of two monographs on robust and decentralised control and of several textbooks on control theory, discrete-event systems and artificial intelligence.

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

Chapter 1

Introduction to diagnosis and fault-tolerant control

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.

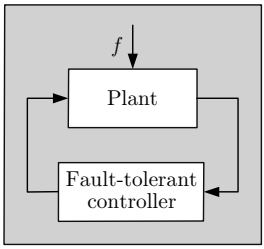


Fig. 1.1. Fault-tolerant system

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 $y \neq y_C$.

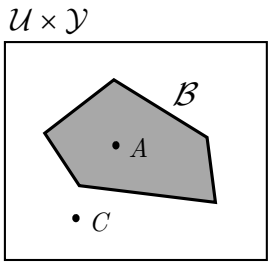


Fig. 1.2. Graphical illustration of the system behaviour

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.

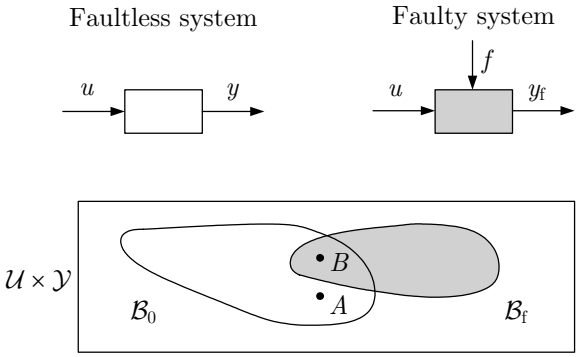


Fig. 1.3. System subject to faults

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 $\mathcal{B}$:

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

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.

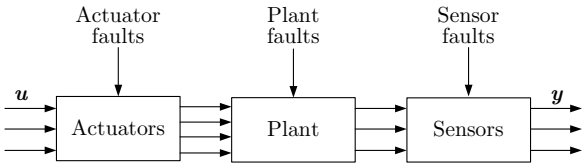


Fig. 1.4. Distinction between actuator faults, plant faults and sensor faults

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 com-

ponent 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.

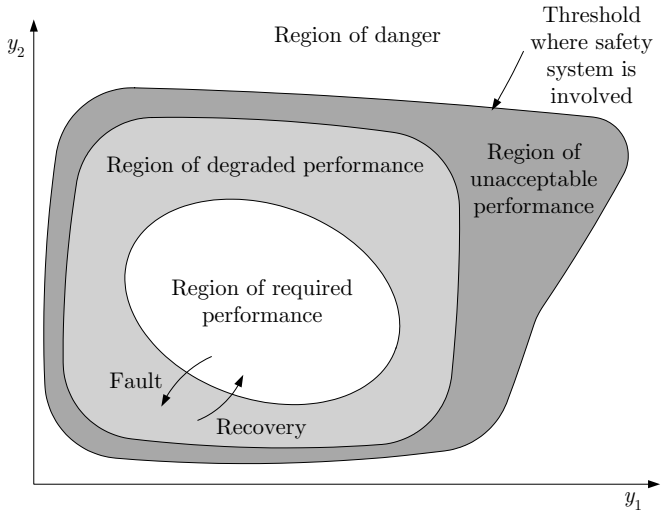


Fig. 1.5. Regions of required and degraded performance

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 bor-

der 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

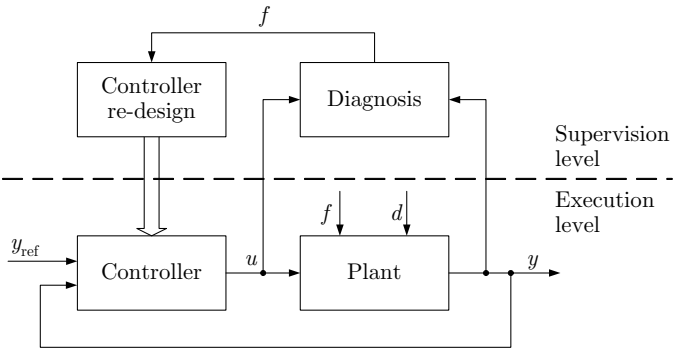


Fig. 1.6. Architecture of fault-tolerant control

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.

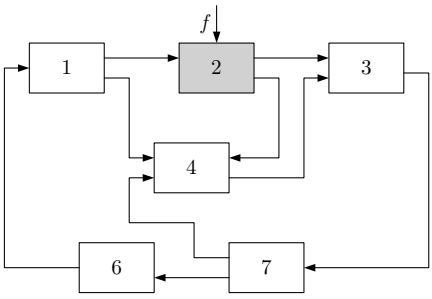


Fig. 1.7. Fault propagation in interconnected systems

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.

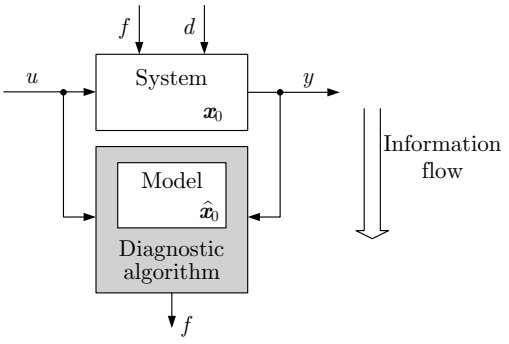


Fig. 1.8. Fault diagnosis

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 $(U,\, Y) \notin \mathcal{B}$ 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 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

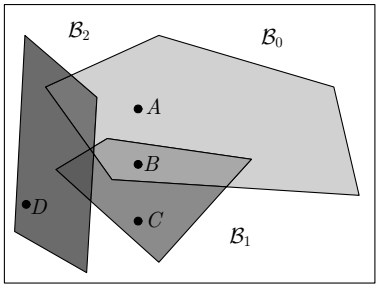


Fig. 1.9. Graphical illustration of the system behaviour

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

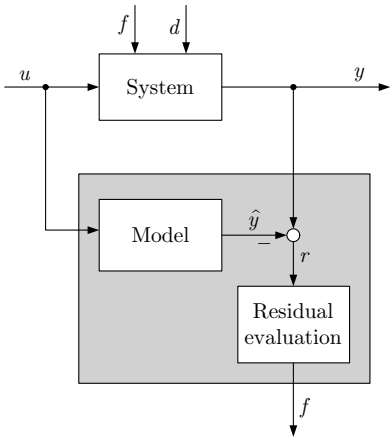


Fig. 1.10. Diagnosis of continuous-variable systems

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.

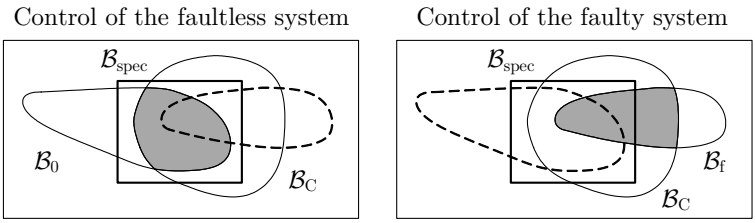


Fig. 1.11. Behaviour of the faultless and faulty closed-loop system

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

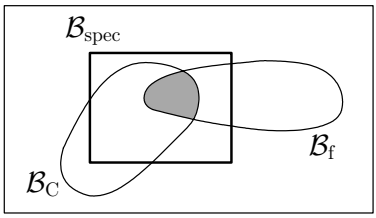


Fig. 1.12. Behavioural representation of fault accommodation

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 $\mathcal{B}_{\mathrm{f}}$ does not overlap with $\mathcal{B}_{\mathrm{spec}}$. 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.

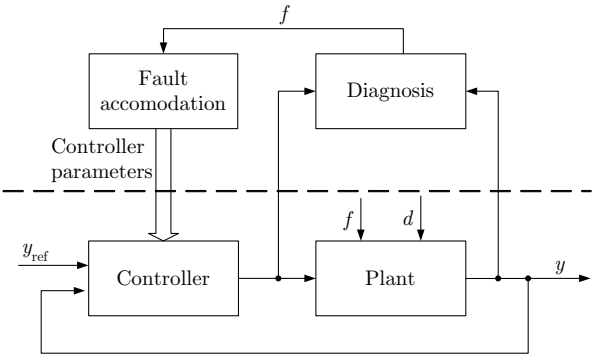


Fig. 1.13. Fault accommodation

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).

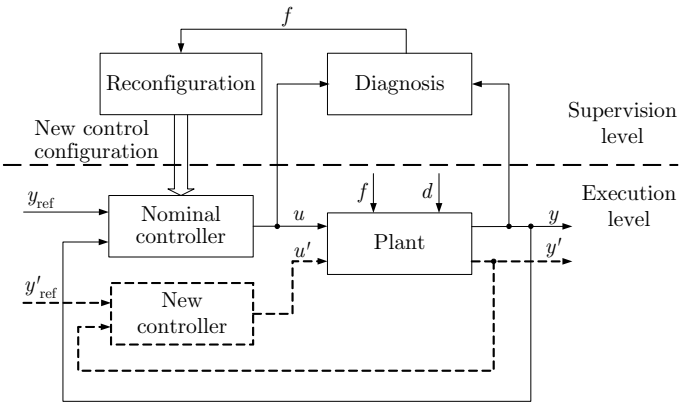


Fig. 1.14. Control reconfiguration

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-

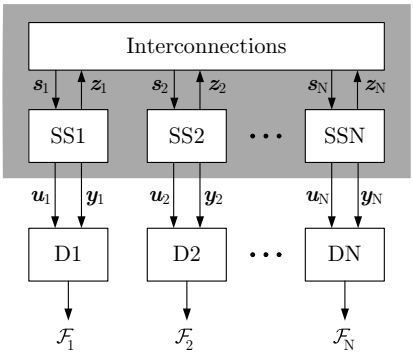


Fig. 1.15. Decentralised diagnosis

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

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

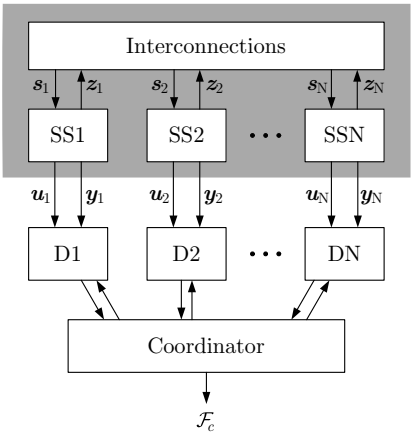


Fig. 1.16. Coordinated diagnosis

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.

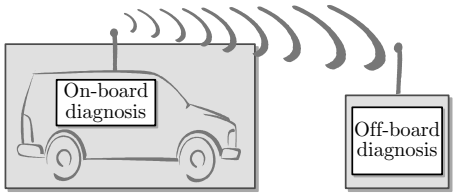


Fig. 1.17. Remote diagnosis

- 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 $V_{\mathrm{D}}(0...k)$. The same happens with the sequence $W(0...k)$ to get the sequence $W_{\mathrm{D}}(0...k)$ used for fault detection.

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

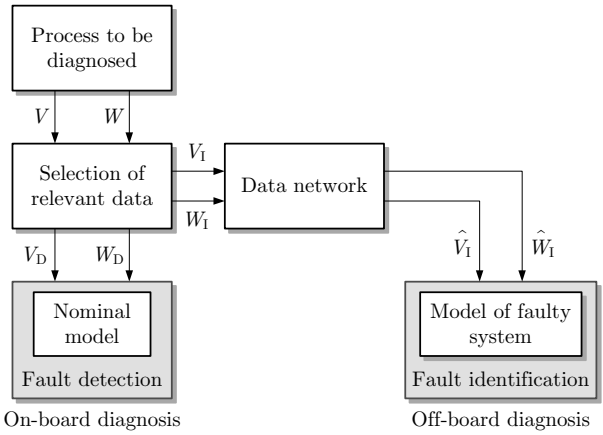


Fig. 1.18. Decomposition of the diagnostic task

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 meth-

ods 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].

Chapter 2

Examples

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

2.1 Two-tank system

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

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

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

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

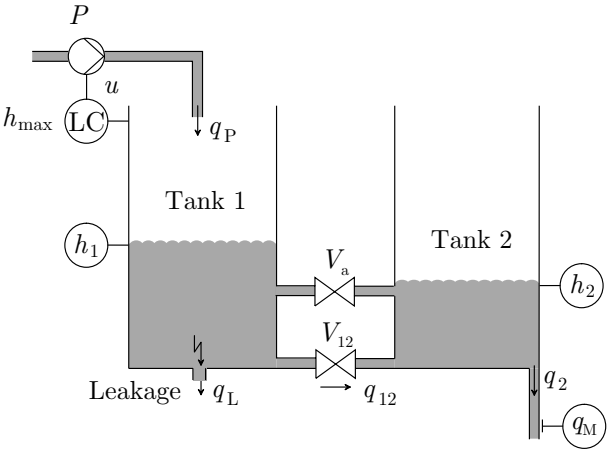


Fig. 2.1. Two-tank system

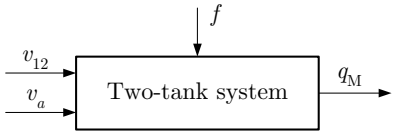


Fig. 2.2. Block diagram of the tank system

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

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

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

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

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

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

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

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

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

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

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

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

Model of the tank system. The two tanks have the liquid levels $h_1(t)$ and $h_2(t)$, which are used as state variables in the model given below. The liquid flows are denoted by q, the ground area of the cylindric tanks by A. The parameters used in the example are summarised in Table 2.1.

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

$$\dot{h}_1(t) = \frac{1}{A}(q_P(t) - q_L(t) - q_{12}(t)) \tag{2.1}$$

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

The measured signal q_M is proportional to the outflow q_2:

$$q_M = c_M \cdot q_2. \tag{2.3}$$

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

Table 2.1 Signals and parameters of the tank system

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

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

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

$$q_P(t) = \begin{cases} u(t) \cdot \bar{q}_P & \text{if } h_1(t) \leq h_{\max} \\ 0 & \text{else ,} \end{cases} \tag{2.6}$$

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

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

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

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

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

2.2 Ship steering and track control

Ship navigation and steering is used as an example to illustrate different methods in both diagnosis and fault-tolerant control. A ship is illustrated in Fig. 2.3. The ship is steered by its rudder, the angle of which is δ. The ships heading angle is denoted ψ, the turn rate ω_3. The ship velocity ahead is v_1, velocity sideways is v_2.

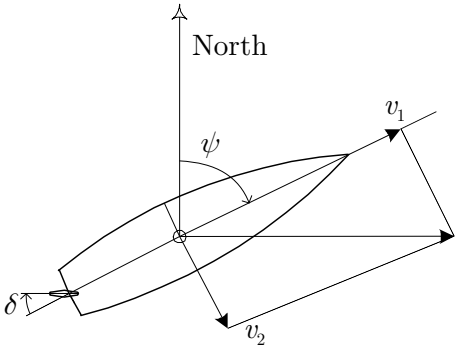


Fig. 2.3. Motion of a ship steered by its rudder. A rudder angle to port side (left) generates a turn to the port side of the ship. When turning to port, there is also a side velocity towards starboard (right).

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

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

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

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

A block diagram of the ship with the above mentioned controllers is shown in Fig. 2.4.

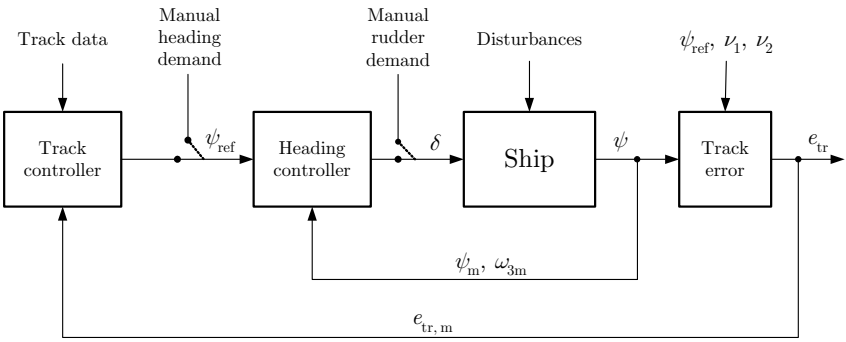


Fig. 2.4. Cascaded architecture of controllers for ship steering. The innermost loop is manual steering with rudder demand as input. The second loop provides automatic heading control, the third implements automatic track control.

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

- Instrumentation with gyro compass and rate gyro as two separate units. The two measurements are independent.
- Measurement of track error by a navigation computer that measures ship's position using a GPS receiver.

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

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

The sensor faults are modelled as additive faults. The rate gyro measures ω_{3m} and the gyro measures the heading angle ψ_m. This is illustrated in the block diagram in Fig. 2.5.

Dynamics of the ship. On a ship, a desired turn rate is obtained by turning the rudder to a certain angle. The input variable is hence rudder angle and the output is turn rate. Waves act as an disturbance to the turn rate, and the combined signal is integrated to give the actual heading of the ship. This dynamics is illustrated in the block diagram 2.5. Turn rate and heading angle are measured variables, the sensors are subject to faults. These are added as fault signals in Fig. 2.5.

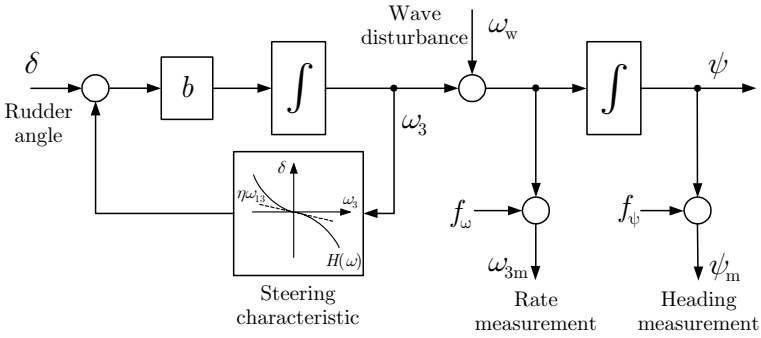


Fig. 2.5. A simple dynamical model of a ship steered by the rudder. Waves act as unknown input and measurement faults are considered on turn rate and heading angle measurements.

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

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

$$\begin{aligned}
\dot{\omega}_3(t) &= b(\delta(t) + H(\omega_3)) \\
\dot{\psi}(t) &= \omega_3(t) + \omega_w(t) \\
\psi_m(t) &= \psi(t) + f_\psi(t) + \nu_\psi(t) \\
\omega_{3m}(t) &= \omega_3(t) + \omega_w(t) + f_\omega(t) + \nu_\omega(t)
\end{aligned} \tag{2.8}$$

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

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

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

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

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

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

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

The variables and parameters in the ship example are listed in Table 2.2

Table 2.2 Signals and parameters of the ship steering example

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

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

$$\delta(t) = L_\omega\,\omega_{3m} + L_\psi\,(\psi_{\rm ref} - \psi_m). \tag{2.9}$$

A block diagram of the autopilot loop is shown in Fig. 2.6.

Track control. Track control means that the ship is commanded to follow a line (great circle) over the sea bottom. The desired track is specified to the controller, and position instruments provide the track error. The control architecture for track control was shown in Fig. 2.4.

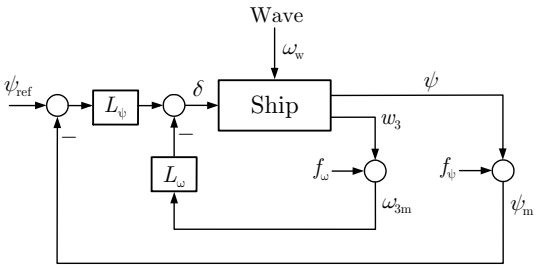


Fig. 2.6. Simple heading controller (autopilot) for the ship example

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

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

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

2.3 Exercises

Exercise 2.1 *Model of ship dynamics*
Using the notation from Section 2.2, the dynamic model of a ship is

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

1. Derive a linear model in state-space form, linearising about the point of operation
$$\bar{\omega}_3 = \omega_o, \quad \bar{\psi} = \psi_o, \quad \bar{\delta} = \delta_o$$
2. Find also the model for the special case
$$\bar{\omega}_3 = 0, \quad \bar{\psi} = 0, \quad \bar{\delta} = 0. \; \square$$

Chapter 3

Models of dynamical systems

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

3.1 Fundamental notions

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

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

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

To illustrate these notions, consider for example a tank. "Storage", which is the *function* classically associated with it, refers to a special operating mode in which the input and the output flows are both equal to zero. In that mode, the mass in the

tank stays constant, which indeed justifies the "storage" denomination. However, many different functions could be assigned to a tank, for example the decoupling (smoothing) of the output flow from some variations of the input flow. This example shows that the notion of function is not univoque, unless the function is understood through the mathematical expression of the constraint that it introduces. In the tank example the tank function would be the "integration" one, since the associated constraint is

$$\frac{dh(t)}{dt} = q_i(t) - q_o(t),$$

where $h(t)$ is the level of the liquid contained in the tank at time t and $q_i(t)$ and $q_o(t)$ are the inflow and outflow at time t.

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

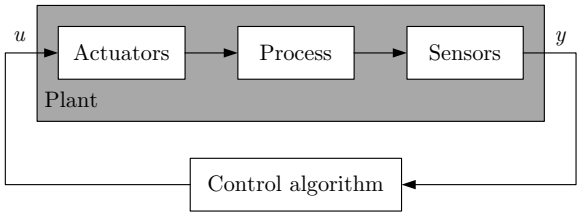


Fig. 3.1. Controlled system

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

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

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

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

$$\begin{aligned} u(t) &= 0 \Longrightarrow q_i(t) = 0 \\ u(t) &= 1 \Longrightarrow q_i(t) = \alpha, \end{aligned}$$

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

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

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

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

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

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

Thus a controlled system is a quadruple

$$<\text{process, actuators, sensors, control devices and algorithms}>.$$

Example 3.1 *Single-tank system*

Consider the following controlled system
$<$(tank, output pipe), input valve, level sensor, level controller$>$.

- Component 1: Tank
  Function: integration
  Constraint: $\frac{dh(t)}{dt} = q_i(t) - q_o(t)$

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

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

- Component 4: Analog level sensor
  Function: provide an image of the actual level in the tank
  Constraint: $y(t) = h(t) + \varepsilon(t), \varepsilon \sim N(0, \sigma)$

- Component 5: On-off control algorithm
  Function: regulate the level in the tank
  Constraint: if $y(t) \leq h_0 - r$ then $u(t) = 1$,
  if $y(t) \geq h_0 + r$ then $u(t) = 0$, where h_0 and r are given constants. □

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

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

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

Example 3.2 *Internal faults of the tank*

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

- Process fault: The tank is leaking.
  Then the description of Component 1 introduced in Example 3.1 is replaced by:
  Component 1: Leaking tank
  Constraint: $\frac{dh(t)}{dt} = q_i(t) - q_o(t) - q_l(t)$ where $q_l(t)$ is some (unknown) leakage flow.

- Actuator fault: The input valve is blocked open.
  Then Component 2 is described by:
  Component 2: Blocked-open input valve
  Constraint: $q_i(t) = \alpha$ whatever the value of $u(t)$.

- Sensor fault: The measurement noise has improper statistical characteristics.
  Then Component 3 is described by:
  Component 3: Level sensor
  Constraint: $y(t) = h(t) + \varepsilon(t)$ with $\varepsilon \sim N(0, \Sigma)$ (instead of $N(0, \sigma)$). □

Example 3.3 *External fault of the tank*

Component 4 defined in Example 3.1 is a control algorithm whose function is to regulate the level in the tank, the objective being to keep $h(t)$ within the interval $(h_0 - r, h_0 + r)$ for any

initial value of the level which belongs to this interval. Note that this objective is expressed in terms of two inequality constraints:

$$\begin{aligned} h(t) &\leq h_0 + r \\ h(t) &\geq h_0 - r. \end{aligned}$$

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

$$\int_{t_1}^{t_2} q_0(t)dt > h(t_1) + \alpha(t_2 - t_1) - h_0 + r.$$

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

3.2 Modelling the system architecture

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

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

Services. A component S is first described by the list of the services that it provides to its users, $S = \{s_i, i \in I_s\}$. A service s_i is a transformation of some *consumed variables* $(cons_i)$ into some *produced variables* $(prod_i)$, that is performed by the

component according to a given procedure $(proc_i)$, either in a systematic way, or only upon some specific *request* $(rqst_i)$.

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

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

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

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

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

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

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

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

be the two measurement equations, where $x(t)$ is the unknown variable to be measured, $y_i(t), i = 1, 2$ are respectively the two transducers output, and $\varepsilon_i, i = 1, 2$

are the measurement noises, with the two Gaussian distributions $N(0, \sigma_i), i = 1, 2$. The *measurement* service of this sensor could obviously be provided under different versions, namely

Version	Procedure
1	$y(t) = \frac{\sigma_2}{\sigma_1+\sigma_2}\, y_1(t) + \frac{\sigma_1}{\sigma_1+\sigma_2}\, y_2(t)$
2	$y(t) = y_1(t)$
3	$y(t) = y_2(t)$

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

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

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

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

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

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

3.3 System behaviour – basic modelling features

Variables. A first question which arises is to select those variables which are of interest to describe the system behaviour. Process components generally introduce power and energy variables, while control systems introduce control and information signals. Therefore, the system variables to be considered are all quantities which are constrained by system components (process, actuators, sensors, control and estimation algorithms). Note that for systems that obey the Markov property, there is a minimal set of variables which summarise the whole past history of the system untill time t (the state variables). The evolution of the state at time t only depends on its value at time t and on the values of the input at time t.

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

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

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

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

Example 3.4 *Single-tank system*

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

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

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

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

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

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

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

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

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

Example 3.5 *On/off-temperature control system*

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

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

when the heater is on, and by the model

$$\frac{d\theta}{dt} = -a\theta$$

when the heater is off (θ is the temperature to be controlled, and a, b are system parameters). The time evolution of such a system is described not only by the temperature (which is a

continuous variable) but also by the heating mode (on/off) which is a qualitative one. Such systems are described by so-called "hybrid models", which will be developed in Section 3.7. □

3.4 Continuous-variable systems

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

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

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

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

where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{u} \in \mathbb{R}^m$, $\boldsymbol{y} \in \mathbb{R}^p$ denote the state vector, the vector of known input signals, and the vector of measured output values. $\boldsymbol{d} \in \mathbb{R}^{n_d}$ stands for the vector of unknown input signals or disturbances acting on the process. The functions $\boldsymbol{g}$ and $\boldsymbol{h}$ are respectively $\mathbb{R}^n$-valued and $\mathbb{R}^p$ -valued and they are assumed to be smooth. A model of the form (3.2), (3.3) can be obtained by using physical laws to describe the considered process.

Example 3.6 *Single-tank model*

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

$$\frac{dh(t)}{dt} = -k\sqrt{h(t)} + \alpha u(t) \tag{3.4}$$

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

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

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

$$\begin{aligned}\frac{d\tilde{\boldsymbol{x}}}{dt} &= \frac{\partial \boldsymbol{g}(\bar{\boldsymbol{x}},\bar{\boldsymbol{u}},\bar{\boldsymbol{d}})}{\partial \boldsymbol{x}}\tilde{\boldsymbol{x}} + \frac{\partial \boldsymbol{g}(\bar{\boldsymbol{x}},\bar{\boldsymbol{u}},\bar{\boldsymbol{d}})}{\partial \boldsymbol{u}}\tilde{\boldsymbol{u}} + \frac{\partial \boldsymbol{g}(\bar{\boldsymbol{x}},\bar{\boldsymbol{u}},\bar{\boldsymbol{d}})}{\partial \boldsymbol{d}}\tilde{\boldsymbol{d}} \\ \tilde{\boldsymbol{y}} &= \frac{\partial \boldsymbol{h}(\bar{\boldsymbol{x}},\bar{\boldsymbol{u}},\bar{\boldsymbol{d}})}{\partial \boldsymbol{x}}\tilde{\boldsymbol{x}} + \frac{\partial \boldsymbol{h}(\bar{\boldsymbol{x}},\bar{\boldsymbol{u}},\bar{\boldsymbol{d}})}{\partial \boldsymbol{u}}\tilde{\boldsymbol{u}} + \frac{\partial \boldsymbol{h}(\bar{\boldsymbol{x}},\bar{\boldsymbol{u}},\bar{\boldsymbol{d}})}{\partial \boldsymbol{d}}\tilde{\boldsymbol{d}}\end{aligned}$$

to obtain a set of linear equations (see Appendix 1)

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

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

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

Example 3.6 (cont.) *Single-tank system*

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

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

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

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

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

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

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

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

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

$\boldsymbol{E}_x$ is defined in a similar way as $\boldsymbol{B}$. Furthermore, the relations $\boldsymbol{C}{=}\boldsymbol{C}_{ct}$, $\boldsymbol{D}{=}\boldsymbol{D}_{ct}$, $\boldsymbol{E}_y = \boldsymbol{E}_{y,ct}$ hold.

Example 3.6 (cont.) *Discrete-time model of the single-tank system*

Letting $\phi = \exp(-\beta T_s)$ and

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

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

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

Stochastic disturbances and measurement noise. For discrete-time stochastic models, measurement noise and stochastic disturbances possibly acting on the state variables are described by stochastic sequences (cf. Appendix 2). A discrete-time state-space model for the system then takes the form

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

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

$$E\left[\begin{pmatrix} \boldsymbol{w}(k) \\ \boldsymbol{v}(k) \end{pmatrix} (\boldsymbol{w}(\ell)' \ \ \boldsymbol{v}(\ell)')\right] = \begin{pmatrix} \boldsymbol{Q}_w & \boldsymbol{Q}_{wv} \\ \boldsymbol{Q}'_{wv} & \boldsymbol{Q}_{\boldsymbol{v}} \end{pmatrix} \delta_{k\ell}.$$

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

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

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

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

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

$$\begin{aligned} \frac{d\boldsymbol{x}(t)}{dt} &= \boldsymbol{A}_{ct}(\boldsymbol{\theta})\boldsymbol{x}(t) + \boldsymbol{B}_{ct}(\boldsymbol{\theta})\boldsymbol{u}(t), \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ \boldsymbol{y}(t) &= \boldsymbol{C}_{ct}(\boldsymbol{\theta})\boldsymbol{x}(t) + \boldsymbol{D}_{ct}(\boldsymbol{\theta})\boldsymbol{u}(t), \end{aligned}$$

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

Example 3.6 (cont.) *Single-tank system*

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

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

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

$$\frac{dh(t)}{dt} = -k\sqrt{h(t)} - k_{leak}(t)\sqrt{h(t)} + \alpha u(t)\ . \tag{3.10}$$

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

$$\frac{dh(t)}{dt} = -\beta h(t) + \alpha u(t) - \sqrt{h_0}k_{leak}(t)\ . \qquad \square$$

3.5 System structure

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

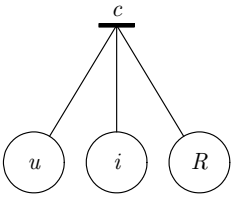
Structural model. The structural model of a system is an abstraction of its behaviour model. For continuous-variable systems, the behaviour is described by a set of algebraic and differential equations. Analysing the structure of these equations resumes to analysing the links which exist between variables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

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

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

The structural model associated with this system says: "There exists one constraint (call it c), which links two variables (u, i) and one parameter (R)". It is represented

by a bi-partite graph $(\mathcal{C}, \mathcal{Z}, \mathcal{A})$ where $\mathcal{C}$ is the set of constraints, $\mathcal{Z}$ the set of variables or parameters, and $\mathcal{A}$ the set of edges between $\mathcal{C}$ and $\mathcal{Z}$ or, equivalently, by this graph's adjacency matrix, as shown on Fig. 3.2, where bars represent constraints and circles represent variables or parameters.



	u	i	R
c	1	1	1

Fig. 3.2. Structure of Ohm's law

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

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

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

$$\begin{aligned} u - Ri &= 0 \\ y_1 - u &= 0 \end{aligned}$$

and, dropping the parameter R, its structure becomes

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

which shows that all the unknown variables in the system can be computed, since u can be computed from y_1 by using the measurement equation m_1 (this is symbolised by the bold **1**), and therefore i can be computed from y_1 and u. Structural observability is indeed one of the properties that structural analysis allows to study. This is of course only a potential property, since constraints m_1 and c might be more complex ones, and the numerical computations might be impossible in some particular cases (change, for example, constraint m_1 into $y_1(1-u)=1$ and suppose that the known value of y_1 is zero!).

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

3.6 Discrete-event systems

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

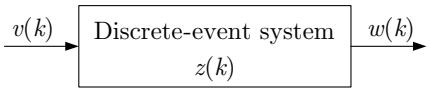


Fig. 3.3. Discrete-event dynamical system

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

$$\begin{aligned} v &\in \mathcal{N}_v = \{1,2,...,M\} \\ z &\in \mathcal{N}_z = \{1,2,...,N\} \\ w &\in \mathcal{N}_w = \{1,2,...,R\} \end{aligned}$$

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

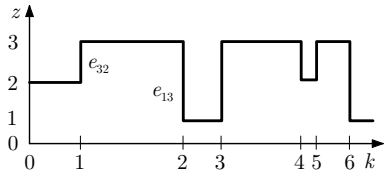


Fig. 3.4. Symbolic signal values and event sequence

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

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

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

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

$$\mathcal{A} = (\mathcal{N}_z,\, \mathcal{N}_v,\, \mathcal{N}_w,\, G,\, H,\, z_0),$$

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

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

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

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

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

Non-deterministic automaton. In the non-deterministic automaton

$$\mathcal{N}(\mathcal{N}_z,\, \mathcal{N}_v,\, \mathcal{N}_w,\, L_n,\, z_0)$$

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

$$L_n : \mathcal{N}_z \times \mathcal{N}_w \times \mathcal{N}_z \times \mathcal{N}_v \;\in\; \{0,1\}$$

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

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

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

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

$$\mathcal{S} = (\mathcal{N}_z,\, \mathcal{N}_v,\, \mathcal{N}_w,\, L,\, \mathrm{Prob}(z(0)))$$

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

$$L : \mathcal{N}_z \times \mathcal{N}_w \times \mathcal{N}_z \times \mathcal{N}_v \;\longrightarrow\; [0,1]$$
$$L(z',\, w,\, z,\, v) = \mathrm{Prob}\,(z_p(1) = z',\, w_p(0) = w \mid z_p(0) = z,\, v_p(0) = v)$$

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

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

$$F(0\,...\,k_h) = (f(0),\, f(1),\, ...,\, f(k_h)).$$

The additional input f extends the stochastic automaton, which becomes

$$\mathcal{S} = (\mathcal{N}_z,\, \mathcal{N}_v,\, \mathcal{N}_f,\, \mathcal{N}_w,\, L,\, \mathrm{Prob}\,(z_0))$$

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

$$L(z',\, w,\, z,\, f,\, v) =$$
$$\mathrm{Prob}\;(z_p(1) = z',\, w_p(0) = w | z_p(0) = z,\, f_p(0) = f,\, v_p(0) = v)\;\;.$$

Example 3.7 *Discrete-event model of the two-tank system*

The question whether a given system should be dealt with as a continous-variable or a discrete-event system depends not only on its properties but also on the task to be solved

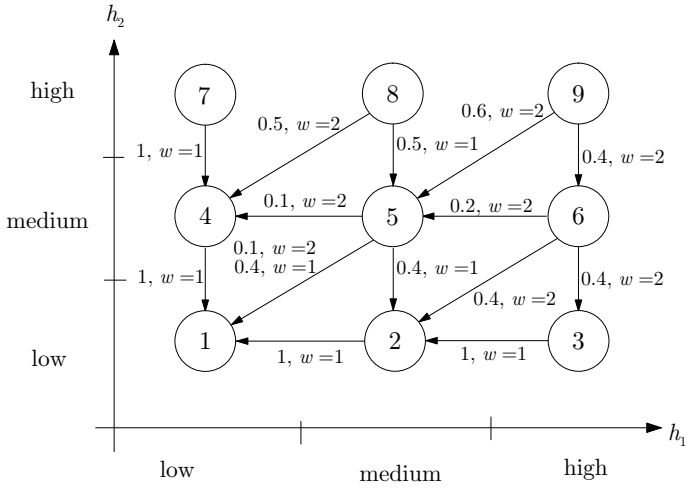


Fig. 3.5. Stochastic automaton describing the tank system for faulty pump ($q_P = 0$)

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

3.7 Hybrid systems

For many technological systems both continuous and discrete phenomena play important roles. The mixture of discrete and continuous signals and discrete and continuous forms of the models used is typical for supervisory control tasks and plays a particular role in diagnosis and fault-tolerant control. As the system possesses both real-valued and discrete-valued signals, combinations of differential equations and automata have to be used for its description (Fig. 3.6).

The main problem in dealing with hybrid systems result from the different range of the signals. The transition between these different ranges are represented by quan-

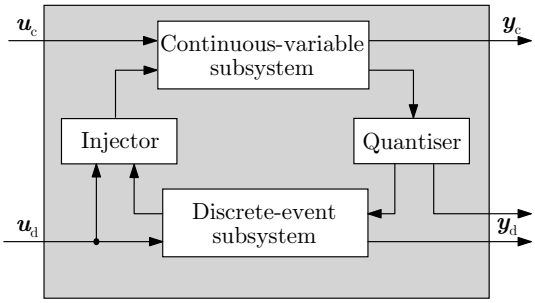


Fig. 3.6. Hybrid dynamical system

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

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

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

Quantised systems, which are considered in Chapter 9 represent an important class of hybrid systems. These systems exhibit principal phenomena that characterise hybrid systems.

3.8 Links between the different models

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

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

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

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

Architecture and behaviour. Components provide services which transform consumed variables into produced variables, according to some given procedure. Variables which are processed by services may be quantitative or qualitative. In any case, the procedures which describe the services of a component are nothing else

than constraints which link the values of the variables associated with this component. The temporal behaviour of these variables is thus defined once the procedures are given. Note that these procedures introduce algebraic and differential constraints for quantitative variables and discrete-event models for qualitative variables.

Example 3.8 *Different models of the tank system*

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

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

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

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

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

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

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

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

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

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

Behaviour and structure. The link between the behaviour model and the structural model is obvious, since the structural model is nothing but an abstraction of the behaviour model. In each operating mode, there is a set of constraints $\mathcal{C}$ which link the values of the system variables $\mathcal{Z} = \mathcal{X} \cup \mathcal{K}$. The structure of these constraints is

directly represented by the set of edges $\mathcal{A}$ in the bi-partite graph $(\mathcal{C}, \mathcal{Z}, \mathcal{A})$ whose nodes are respectively $\mathcal{C}$ and $\mathcal{Z}$.

Example 3.9 *Structure of a valve in different operation modes*

In the valve example, the are two different structure associated with the four different operating modes which appear on the hybrid description of the system behaviour:

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

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

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

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

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

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

3.9 Exercises

Exercise 3.1 *Model of ship dynamics*

Using the notation from Section 2.2, the dynamic model of a ship is

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

1. Derive a linear model in state-space form, linearising about the point of operation
   $$\bar{\omega}_3 = \omega_o, \quad \bar{\psi} = \psi_o, \quad \bar{\delta} = \delta_o$$
2. Find also the model for the special case
   $$\bar{\omega}_3 = 0, \quad \bar{\psi} = 0, \quad \bar{\delta} = 0. \; \square$$

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

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

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

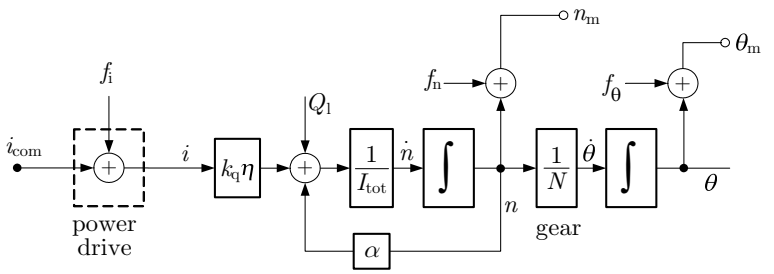


Fig. 3.7. Block diagram of actuator with additive faults - open loop

The faults concern

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

With $\boldsymbol{x} = (n, \theta)'$, $u = i_{\rm com}$, $d = Q_l$, $\boldsymbol{f} = (f_i, f_n, f_\theta)'$, $\boldsymbol{y} = (n_m, \theta_m)'$, the actuator has the following state-space representation

$$\begin{aligned} \frac{d}{dt}\boldsymbol{x} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}u + \boldsymbol{E}_x d + \boldsymbol{F}_x \boldsymbol{f} \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} + \boldsymbol{F}_y \boldsymbol{f}. \end{aligned} \tag{3.15}$$

1. Show that
$$\boldsymbol{A} = \begin{pmatrix} -\frac{\alpha}{I_{tot}} & 0 \\ \frac{1}{N} & 0 \end{pmatrix}$$
and determine the remaining matrices in the state-space model.
2. Show that the system transfer function (Laplace domain) is
$$\begin{aligned} \boldsymbol{x}(s) &= (s\boldsymbol{I} - \boldsymbol{A})^{-1}(\boldsymbol{B}u(s) + \boldsymbol{E}_x d(s) + \boldsymbol{F}_x \boldsymbol{f}(s)) \\ \boldsymbol{y}(s) &= \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}(\boldsymbol{B}u(s) + \boldsymbol{E}_x d(s) + \boldsymbol{F}_x \boldsymbol{f}(s)) + \boldsymbol{F}_y \boldsymbol{f}(s). \end{aligned}$$
3. Using the shorthand notation
$$\boldsymbol{y}(s) = \boldsymbol{H}_{yu}(s)u(s) + \boldsymbol{H}_{yd}(s)d(s) + \boldsymbol{H}_{yf}\boldsymbol{f}(s),$$
determine the three transfer function matrices $\boldsymbol{H}_{yu}, \boldsymbol{H}_{yd}$ and $\boldsymbol{H}_{yf}$. Verify what is apparent from the block diagram,
$$n_m(s) = \frac{1}{sI_{tot} + \alpha}(k_q\eta i_{\rm com}(s) + Q_l(s) + k_q\eta f_i(s)) + f_n(s). \ \square$$

Exercise 3.3 *Discrete-time model of industrial actuator*

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

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

Exercise 3.4 *Industrial actuator with speed control*

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

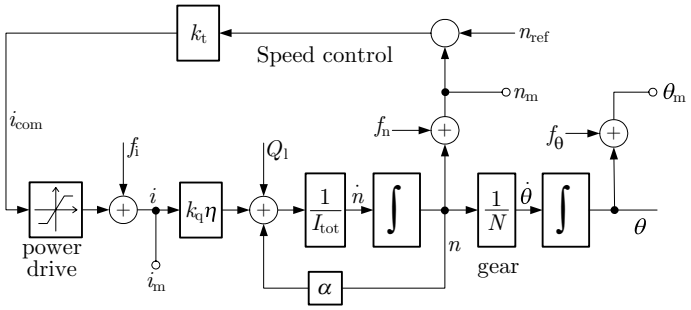


Fig. 3.8. Actuator with angular velocity feedback

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

$$\frac{d}{dt}\begin{pmatrix} n \\ \theta \end{pmatrix} = \boldsymbol{A}_{cl}\begin{pmatrix} n \\ \theta \end{pmatrix} + \boldsymbol{B}_{cl} n_{\mathrm{ref}} + \boldsymbol{E}_{x,cl} Q_l + \boldsymbol{F}_{x,cl}\boldsymbol{f}$$

$$\begin{pmatrix} n_m \\ \theta_m \\ i_m \end{pmatrix} = \boldsymbol{C}_{cl}\begin{pmatrix} n \\ \theta \end{pmatrix} + \boldsymbol{D}_{cl} n_{\mathrm{ref}} + \boldsymbol{E}_{y,cl} Q_l + \boldsymbol{F}_{y,cl}\boldsymbol{f}$$

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

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

3.10 Bibliographical notes

Modular and object-oriented models have been developed to describe architectures of automated systems [2], [110], [165]. Such models are used in the description and the validation of real-time and distributed systems [253], and specific tools, like state charts have been developed to describe their real-time operation [90]. Generic component models are architecture models, first developed for the description of intelligent sensors and actuators [225], [230]. They have been used for the interoperability analysis of distributed architectures [1], [9], [11], [30]. There exists a bridge between SyncCharts and generic component models, which provides a means of analysing both the system architecture and its associated real-time behaviour [8].

Behaviour models based on "first principles" describe the power exchanges and transformations which take place in a process. Bond graphs, first developed in [197], describe power as the product of an effort (e.g. voltage, pressure, force) and a flow (e.g. current, volume flow, velocity), and use a graphical representation to describe the exchanges of power between different components of a process. They provide a unified modelling approach for different engineering disciplines, since power is a concept which is shared by all of them. Bond graph modelling has been further developed in [112] and [252], and recently extended to thermal and chemical engineering in [251]. Bond graph models are used for simulation and control design, as well as for the design of fault detection and isolation algorithms [247].

For an introduction into the wide field of system identification, the reader is referred to the monographs [132], [221] and [261].

Good introductions to discrete-event systems are [39] and [148].

Chapter 4

Analysis based on components and architecture

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

4.1 Introduction

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

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

Example 4.1 *Tank system*

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

UM 0: No operation

UM 1: Filling the tank

UM 2: Processing the batch

UM 3: Emptying the tank via the normal pipe

UM 4: Emptying the tank via the "lost production" pipe

UM 5: Cleaning the tank.

The associated use-mode automaton is illustrated in Fig. 4.1.

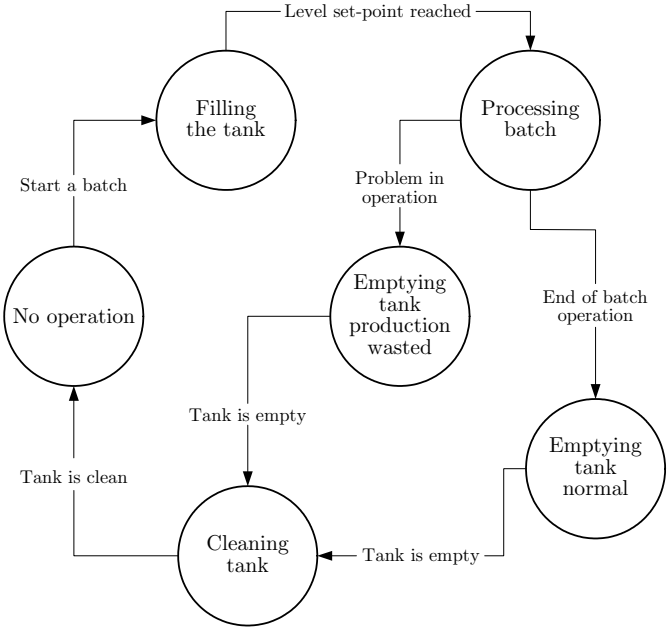


Fig. 4.1. Automaton of a batch process illustrated through use-modes

There are six objectives associated with the different use-modes, namely

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

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

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

4.2 Generic component models

Generic component models describe the system architecture from a formal point of view, so as to perform systematic manipulations for the purpose of fault diagnosis and fault-tolerant-control design. Contrary to box models, which carry no information about the component behaviour in different operating situations (normal, faulty, different modes), generic component models describe components, which offer services according to the current use-mode. The user may be a human operator (who directly accesses the component through some man-machine interface) or another component, which accesses the services either through direct or remote connection (as in distributed systems in which services are requested via a local area network).

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

4.2.1 Services

From the user viewpoint, a system component provides one or several services. For example, a level sensor provides a signal which is a one-to-one correspondence to the level in the tank. However, the signal may be validated or not, it may be filtered or not, the sensor might memorise the minimal (the maximal) level value

encountered on a given time window, it might provide an alarm if the level exceeds a given threshold, etc. All these are examples of services the sensor might provide in the normal operating mode. Other services could be provided in the installation, initialisation, degraded operation, maintenance modes.

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

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

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

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

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

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

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

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

$$S(k) = \{s_i(k), i \in I_s(k)\} \tag{4.1}$$

$$s_i(k) = \{cons_i(k),\ prod_i(k),\ proc_i(k),\ rqst_i(k),\ active_i(k),\ res_i(k)\}, \tag{4.2}$$

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

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

For that reason, definition (4.1) is further extended, by adding some organising structure on the set of services. Normal operating modes are called $use - modes$. They provide the formal model of the structure of the set of services.

Definition 4.1 (Use-mode (UM))
A use-mode is a subset of services of a component. The set of use-modes covers the set of services, i.e. each service belongs at least to one use-mode, and each use-mode contains at least one service.

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

4.2.2 Introduction of the generic component model

The consequence of the use-mode definition is that the component model must now include a (higher) level description, which models the component possible transitions from one use-mode to another one, and the conditions under which these

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

$$\begin{aligned} No-operation : m_1 &= \{set\_mode\_m_2,\ set\_mode\_m_3\} \\ Initialise : m_2 &= \{enter\_set-point,\ display\_set-point, \\ &\quad set\_mode\_m_1,\ set\_mode\_m_3\} \\ In-control : m_3 &= \{read\_set-point,\ calculate\_control\_signal, \\ &\quad set\_mode\_m_1\}. \end{aligned}$$

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

Definition 4.2 (Generic component model)
A system component is defined by the following formal model[1]*:*

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

This definition will be extended later.

4.2.3 Simple components

A simple component is described by the services offered and its use-mode automaton. A component is considered simple when it has no internal means to change

[1] The notion of versions have been introduced in Section 3.2 and will be elaborated in more detail in Section 4.2.4.

the services it provides. Simple components are typically without build-in computational means. Two examples of an actuator and a sensor are treated below.

Actuator for flow control. Flow control is the most widespread actuator function in machinery systems. It is used where a shut-off of a pipe connection is needed, where the flow of a medium is to be controlled, and where control loops manipulate a flow of a liquid in order to change a temperature. Flow control can be open/closed, variable throughput, or redirection of flow from one pipe into two (3-way valves).

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

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

The state is controlled by activation of two relay contacts. They are denoted "open" and "close", respectively. A potentiometer is used to measure the actual rotor position. Figure 4.2 shows the principle in the actuator operation and electrical connections.

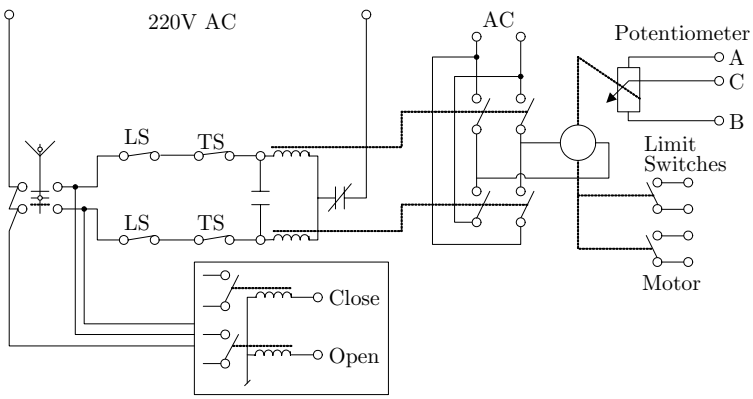


Fig. 4.2. Operation of 3-way valve actuator with relay operated induction motor. Abbreviations: o:open, c:close, s:stop, LS:Limit switch, TS:Torque switch

Limit switches on the rotor provide indication of rotor end positions to permit adequate controller design and fault detection. Torque switches are mounted to provide overload protection.

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

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

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

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

Potentiometer. A potentiometer changes the position of contact between a resistance element and a wiper when the turning angle is changed. The potentiometer can be considered a voltage divider with a division ratio that is a function of the turning angle. Figure 4.3 shows the typical connection diagram.

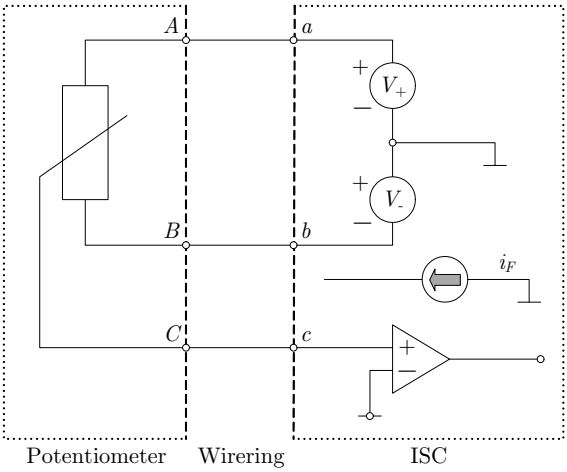


Fig. 4.3. Electrical diagram of potentiometer and computer interface to enable fault detection at the single sensor level

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

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

4.2.4 Complex components

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

Versions of services. Let $s_i(k)$ be a service provided by component k, and suppose that embedded fault tolerance is available. This means that the service $s_i(k)$

would not be interrupted even if the resources it needs were no longer available. This is only possible if it exists within component k under several versions, namely $s_i(k) = \left\{ s_i^j(k),\, j \in J(s_i(k)) \right\}$ where $s_i^j(k)$ is the j^{th} version of service $s_i(k)$.

From this extension, definition (4.1) can now be stated as:

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

where each version $s_i^j(k)$ of service $s_i(k)$ is a 6-tuple like (4.2), which can be used indifferently for the same purpose.

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

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

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

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

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

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

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

Example 4.2 *Component analysis of ship track control*

To illustrate the component analysis it will be applied to the ship steering example introduced in Section 2.2. The subcomponents of the ship steering controller are:

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

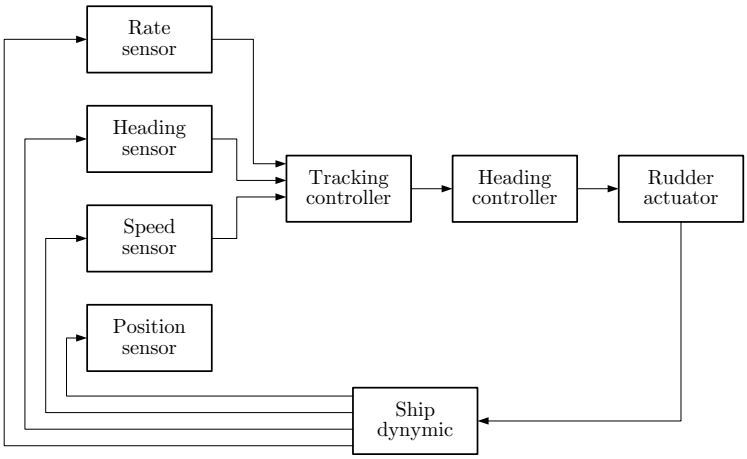


Fig. 4.4. Fault propagation in the ship steering problem

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

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

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

$s(0) = < set\_track;\ set\_heading >$
$s(1) = < set\_track;\ set\_heading >$
$s(2) = < set\_track;\ set\_heading;\ keep\_heading >$
$s(3) = < set\_track;\ keep\_track >.$

The *keep_heading* service calculates the necessary rudder action in order to maintain the heading of the ship based on the measured rate and heading. The service can be defined as:

$$
\begin{aligned}
keep\_heading = \quad & < \psi, \psi_{\mathrm{ref}}, \omega_m >; \\
& < \delta >; \\
& < heading\ controller >; \\
& < \psi_{\mathrm{ref}}\ defined >; \\
& < none >; \\
& < Gyro,\ rate\ sensor > .
\end{aligned}
$$

The *keep_track* service calculates the necessary heading in order to maintain the track of the ship based on the measured rate, heading and position. The service can be defined as:

$$
\begin{aligned}
keep\_track = \quad & < track\_error\ e,\ track,\ ship\ speed\ v_1 >; \\
& < \psi_{\mathrm{ref}} >; \\
& < track\ controller >; \\
& < track,\ reference\ defined >; \\
& < none >; \\
& < Gyro,\ rate\ sensor,\ speed\ sensor > .
\end{aligned}
$$

The *set_track* service prompts the user to input waypoints for the desired track:

$$
\begin{aligned}
set\_track = \quad & < waypoints >; \\
& < track >; \\
& < track\ planner >; \\
& < user\ input >; \\
& < none >; \\
& < electronic\ seamap > .
\end{aligned}
$$

The *set_heading* service prompts the user to set a desired heading:

$$
\begin{aligned}
set\_heading = \quad & < \psi_{in} >; \\
& < \psi_{\mathrm{ref}} >; \\
& < \psi_{\mathrm{ref}} = \psi_{in} >; \\
& < user\ input >; \\
& < none >; \\
& < none > .
\end{aligned}
$$

In the above service definitions some resources are not considered, for example electrical power. □

4.2.5 Building systems from components

Systems are built from the interconnection of different components. Components are interconnected because the services delivered by some of them consume variables which are produced by services of others. The *measurement* service of a sensor, for example, consumes variables produced by the environment of the system, and produces variables which are consumed by the *regulation* service of the regulator, which in turn produces variables which are consumed by the *power modulation* service of the actuator.

In the generic model approach, interconnections are taken into account by considering higher level components which are composed of lower level ones. Therefore, systems are built following a bottom-up approach.

Indeed, system architectures can be described at different hierarchical levels. Sensors, actuators, process components are at the field-level. Higher level components can be built from the aggregation of lower level ones at any hierarchical level. For example, the aggregation of a tank, an input valve, an output pipe, a level sensor and a regulator (with consistent connexions between them) is a high level component, the "single-tank system". Whatever the component level, its generic model can be built defining its use-mode automaton, and the services which are available in each use-mode. Aggregation procedures have to be defined in order to build the generic model of high level components from the generic models of the low level components they are composed of. High level services allow to fulfil the system mission, and the analysis of the overall system fault tolerance can be based on the search of the existence of different versions of high level services.

Aggregation of operation modes. The generic model of a component is first given by its use-mode automaton. In each use-mode, the component is able to perform a set of services (each of them under a variety of versions) in order to achieve some pre-specified objective. Recall that the use-mode automaton which describes a component is a graph $A(M, \tau, m^0)$. Let $A(M(k), \tau(k), m^0(k))$ and $A(M(l), \tau(l), m^0(l))$ be the deterministic automata associated with two components k and l, and let kl be a higher level component, which aggregates these two ones. The automaton $A(M(kl), \tau(kl), m^0(kl))$ associated with the component kl is obviously contained in the asynchroneous product of the two automata $A(M(k), \tau(k), m^0(k))$ and $A(M(l), \tau(l), m^0(l))$:

- $M(kl) \subseteq M(k) \times M(l)$
- $\tau(kl) \subseteq \tau(k) \cup \tau(l)$
- $m^0(kl) = \big(m^0(k), m^0(l)\big)$

Let $(\alpha, \beta) = \mu \in M(k) \times M(l)$. This means that the mode μ of the high level device kl is defined as component k being in mode α and component l being in mode β. Since not every such association is meaningfull, the set of modes $M(kl)$ has to be selected by the designer, by eliminating from $M(k) \times M(l)$ the non-significant or non-allowed associations.

Example 4.3 *Temperature controller*

Consider a temperature controller as a high-level component, obtained by the aggregation of three low-level ones, namely a temperature sensor, a PI regulator, and a heating valve. The use-modes of the low-level components are as follows:

Sensor: {*off*, *calibration*, *automatic*}
Regulator: {*off*, *on* }
Valve:{*off*, *manual*, *automatic*},

where the calibration mode of the sensor and the manual mode of the valve are used for maintenance operations. Then, the asynchroneous product of the three use-mode automata

gives 18 compound modes. Many combinations are inconsistent, e.g. (*off, on, manual*) or (*calibration, on, automatic*), leaving only three consistent modes to describe the "temperature controller" device. The three use-modes of the high level component are therefore {*off, maintenance, automatic*}, and they are defined as follows from the use-modes of the low-level components:

$$\begin{aligned} \textit{off} &= \textit{(off, off, off)} \\ \textit{maintenance} &= \textit{(calibration, off, off)} \vee \textit{(calibration, off, manual)} \\ &\qquad \vee \textit{(off, off, manual)} \\ \textit{automatic} &= \textit{(automatic, on, automatic).} \quad \square \end{aligned}$$

Aggregation of services. Let $S(k)$ and $S(l)$ be the services offered by two low-level components k and l, and let us consider the high level component kl which is their aggregation. Let $(\alpha,\ \beta) = \mu \in M(kl)$ be a consistent use-mode, then any combination of the services $S_\alpha(k)$ (available when component k is in mode α) and $S_\beta(l)$ (available when component l is in mode β) can be used. In other words, any *program* using the services of $S_\alpha(k)$ and $S_\beta(k)$ as *instructions* can be a service available in the mode μ. Again, every combination is not consistent, and only programs with functional interpretations in the application framework are to be considered.

Example 4.3 (cont.) *Temperature controller*

The following program defines a high-level service, available in the automatic mode of the "temperature controller" component:

Regulation service:
Repeat,
  Request the measurement service of the sensor,
  Request the calculation service of the regulator,
  Request the actuation service of the valve,
Until end of the regulation service.

Note that if the measurement service of the sensor is available under three versions, the calculation service under two versions, and the actuation service under two versions, then the regulation service is available under twelve versions. □

Hierarchical levels. System architectures can be described at different hierarchical levels. Sensors, actuators, process components are at the field-level (they exchange data at field-bus level). Higher level components can be built from the aggregation of lower level ones at any hierarchical level. For example, the aggregation of a tank, an input valve, an output pipe, a level sensor and a controller (with consistent connections between them) is a high-level component which can be named the single-tank subsystem. Whatever the component level, its generic model can be built defining its use-mode automaton, and the services which are available in each use-mode. Aggregation procedures have to be defined in order to build the generic model of higher-level components from the generic models of lower level components. High-level services enable fulfilment of the system mission, and the analysis

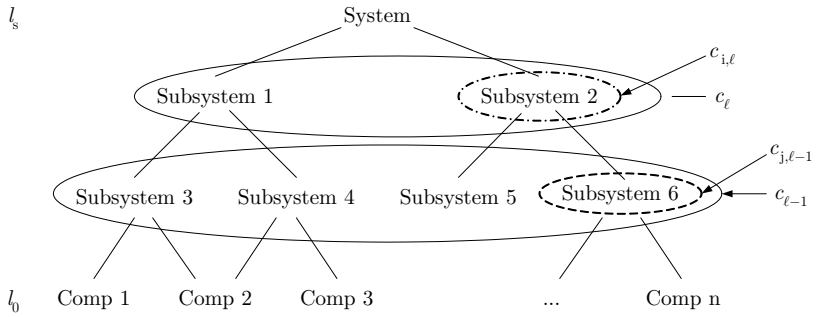


Fig. 4.5. Aggregation of low-level components into high-level ones

of the overall system fault tolerance can be based on the search of the existence of different versions of high-level services.

4.3 Faults in components and their consequences

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

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

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

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

Connection with reliability analysis. For the reasons given above, fault analysis need to incorporate analysis throughout a system. Traditional methods for fault detection and isolation do not cover this problem. They are very able to detect the presence of a fault as a difference between actual and expected behaviour. Isolation of a particular fault requires a hypothesis about the observed effects from this fault. This is obtained by ad hoc engineering and requires deep process knowledge and engineering skills to make a successful design. It is expensive in terms of both key personnel and time.

Analysis of system reliability is mandatory for safety critical systems but is also more and more often used for common industrial systems, driven by the increasing environment and safety awareness in recent years. The state of the art is such that no method can guarantee a complete description of all possible fault modes of a system. Certain forms of risk analysis provide, nevertheless, a very systematic approach to fault modelling once possible component faults have been identified. Faults in common industrial components are subject to constant study, and a methodology based on component fault modelling could use accumulated knowledge for each type of component. The number of principally different components in a certain branch of industry is small enough to make this a manageable exercise.

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

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

The systematic approach shall provide the following information:

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

This is elaborated in the following.

4.4 Fault propagation analysis

Several approaches exist to analyse systems based on the components they comprise. The fields of risk analysis and reliability engineering have developed several approaches to assess the risks associated with component break-down. On commonly accepted standard in everyday industrial use is the failure-modes and effects analysis (FMEA) technique. It is based on description of the failure modes of the individual components, and would thus serve our purpose. The failure mode and effects analysis technique is well established and supported by both databases with breakdown information and mean-time between failure history for many components. It was hence natural to develop a method for analysis of fault propagation based on such available information on component failure modes.

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

An example for a typical failure modes and effects analysis worksheet is illustrated for a pressure gauge in the table. The failure modes and effects analysis worksheet has columns for *item identification*, *failure modes, failure cause, failure effect* and *risk assessment* for the end effects at system/environment level. There are also columns for *risk code* and *actions required*, not shown here.

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

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

Fault propagation matrix. A traditional failure mode and effects analysis starts with selection of the lowest level of analysis. In the present context, this means sensors, valves, motors and similar components. All potential faults and their effects are determined. A fault propagation scheme for each component shows how fault effects out of the component relate to faults at input, output, or parts within the components. This is illustrated in Fig. 4.6.

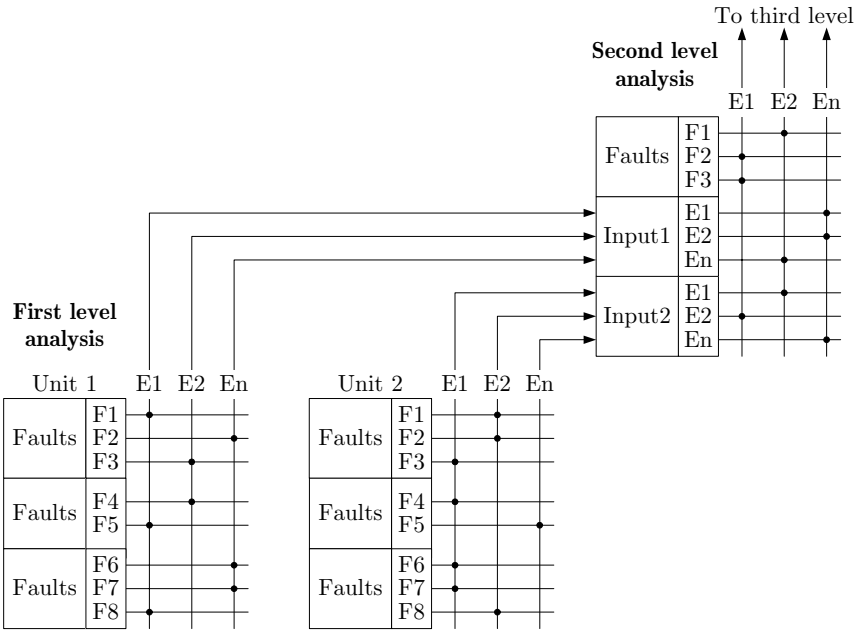


Fig. 4.6. Traditional failure modes and effects analysis scheme illustrated graphically for two component levels

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

Definition 4.3 (Fault propagation matrix)
For a given Boolean mapping $\boldsymbol{M}$

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

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

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

A fault propagation matrix scheme can be expressed as

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

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

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

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

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

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

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

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

Eventually, end effects at the system level are reached.

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

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

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

$$\boldsymbol{f}_c = \left(\boldsymbol{M}^f\right)' \odot \boldsymbol{e}_c$$

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

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

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

Example 4.4 *Propagation with three components*

A system with three components and open loop structure is

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

The fault effect scheme for this example is

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

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

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

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

Merging two levels can then be re-written

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

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

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

Example 4.5 *Autopilot - gyro system diagnosis*

Failure of ships' motion control systems have been the cause of many severe accidents. Some of these were caused by faults in the gyro-system providing the heading and turn rate motion feedback to the autopilot. Early detection of such faults could prevent the control system from unwanted alteration of the ship's heading. This example illustrates fault detection on a ship's gyro compass and an associated turn rate sensor.
Faults are possible in either of the two measurements and the purpose of the fault detection is to isolate the faulty sensor. Subsequent fault accommodation should then switch the faulty sensor out and estimate the missing signal from that of the good sensor. Fault-tolerance against these faults is obtained by implementing the scheme as an autonomous part of the heading control loop.

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

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

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

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

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

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

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

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

The interpretation is clearly that

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

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

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

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

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

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

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

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

Definition 4.4 (Generic component model (extended))
A system component is defined by the model given in Definition 4.4 together with the additional model part:

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

Example 4.6 *Temperature control*

This example illustrates the aggregation of simple components into a complex component that provides a temperature control service. The problem considered is to accommodate some of the faults that would stop the primary service of the temperature control loop in Fig. 4.7. A three way valve controls the mixing of hot water returning from a tank with tempered water from a heat exchanger. The control objective is to keep the cooling water temperature of an exogenous process in the tank at a constant value. The valve is controlled by the temperature control loop, which consists of

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

The control loop is shown in Fig. 4.7.

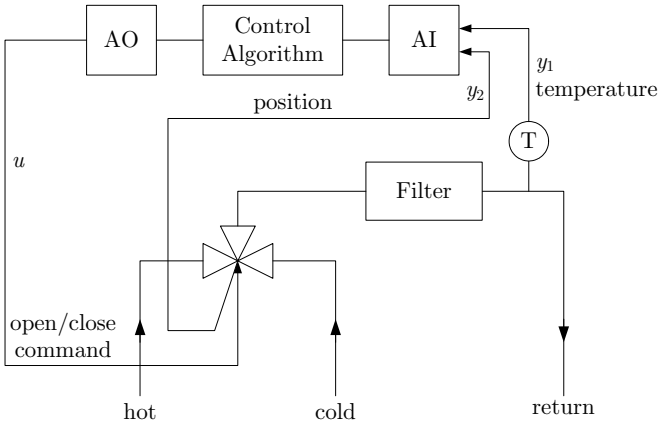


Fig. 4.7. Piping and instrumentation diagram representation of a temperature control loop with 3-way valve

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

Three way valve actuator. The valve is driven by an AC motor which is activated by a double acting relay to either side of rotation. End stop switches are supposed to avoid motor overload and force the motor into mechanical stop should the position control loop fail in some way. The potentiometer gives position feedback. The position loop fails if either potentiometer or end stop switches fail.

Figure 4.8 shows the graphical representation of the FPA scheme for the closed-loop valve position controller. Bold lines in the scheme show how faults propagate. The important observation is that propagation could be stopped at the points marked with stars. This means that fault handling should be applied exactly at these points.

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

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

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

Figure 4.8 showed the FPA scheme in block-diagram form for the valve control part of the loop. The components are: potentiometer, limit switches, motor, 3 way valve, and digital controller.

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

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

- **Actuator fault.** Fault in the valve up-down relay switch or in the ac-motor: The position controller must stop immediately. This will cause a loss of the temperature control service.

- **Actuator fault.** Fault in the valve end-stop switch: The position controller switches to use the position sensor and up-down commands for estimation of position. The service continues until maintenance.

- **Position sensor fault.** The controller should be re-configured. The analytical relation between duration of relay pulses and motor shaft position, a position estimate is readily available. The estimate is used until the fault is repaired.

- **Temperature sensor fault.** The reference to the position controller fails. The controller is reconfigured and a time-history roll-back is made of the reference signal and the mean is used as the new reference until the fault has been repaired.

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

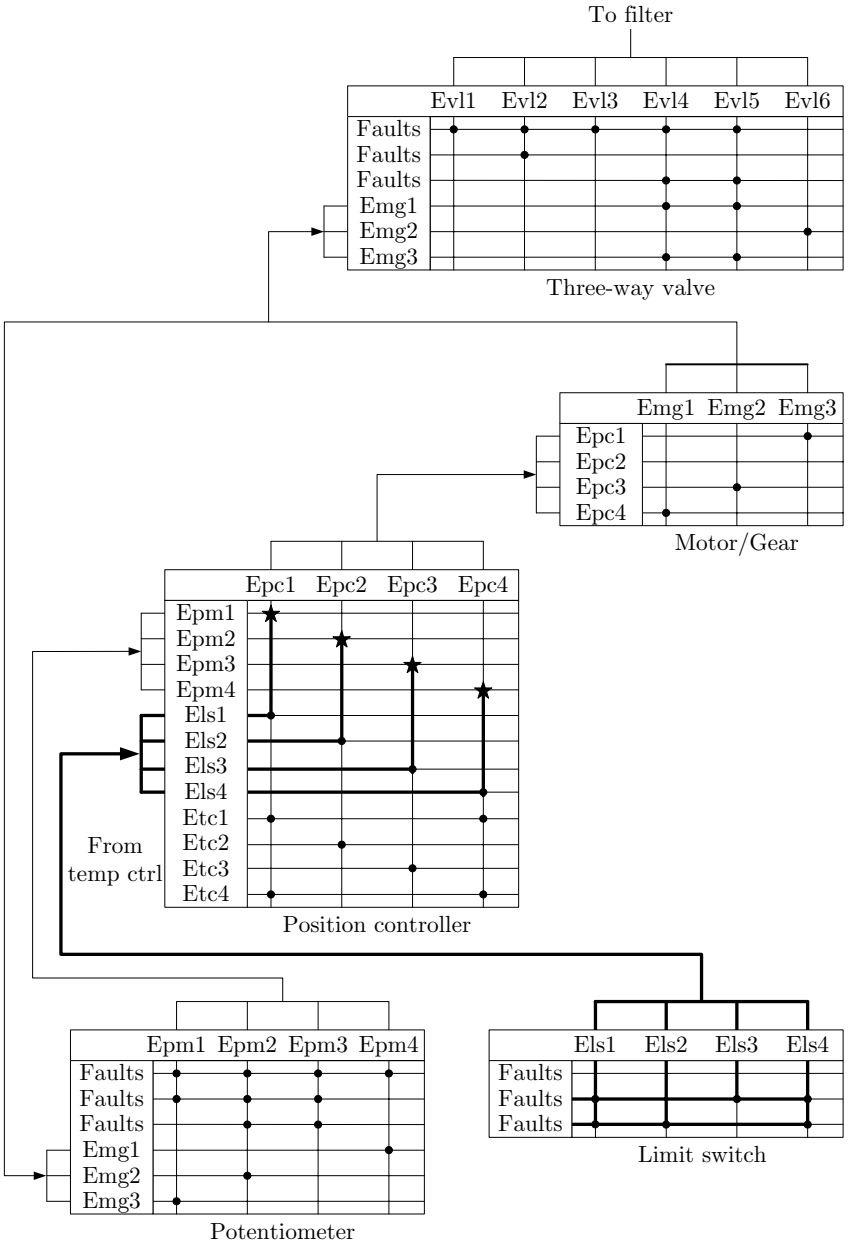


Fig. 4.8. Propagation of fault effects in closed-loop control of 3-way valve. Solid lines show fault propagation, points marked with star show where propagation can be stopped.

4.5 Graph representation of component architecture

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

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

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

Definition 4.5 (Fault propagation analysis graph)
Let a system be comprised of items: input effects ι, components with FPA blocks γ and output effects ζ. Define a set of vertices as $V = \{\iota, \gamma, \zeta\}$ of an FPA graph. Connections between the system items are edges of the graph. The edges constitute the set E. The FPA graph Γ is an ordered pair of disjoint sets (V, E). $V = V(\Gamma)$ is the set of vertices and $E = E(\Gamma)$ the edge set.

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

Definition 4.6 (Orientation)
An edge (i, j) is said to connect a vertice j to i. If an edge is oriented and connects vertice j with i, then $e_{ij} = 1$.

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

Definition 4.7 (Directed adjacency matrix)
The directed adjacency matrix $\boldsymbol{D}$ of Γ, with respect to a given orientation of Γ, is the $n \times n$ matrix (d_{ij}) whose entries are

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

where the number of vertices in the graph is n.

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

Remark 4.2 *Input vertex*
The i^{th} vertex is an input vertex if and only if the i^{th} row in the adjacency matrix comprise zeroes only. □

Remark 4.3 *Output vertex*
The j^{th} vertex is an output vertex if and only if the j^{th} column in the adjacency matrix comprise zeroes only. □

Definition 4.8 (Walk of length k)
A walk of length k in Γ is a finite sequence of vertices in the graph Γ $\{v_0, v_1, ..., v_k\}$ such that v_{t-1} and v_t are adjacent for $1 \leq t \leq k$.

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

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

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

Theorem 4.1 Loops of length k in a fault propagation graph
A graph with a vertex v_i has exactly one walk back to itself of length k if and only if the i^{th} diagonal entry of $\boldsymbol{D}^k$ is 1. Each vertex in the loop has it's diagonal entry equal to 1.
A vertex v_i that participates in n closed loops will have a diagonal entry in $\boldsymbol{D}^k$ equal to n. The number n will include possible multiple rounds in a loop if its length is an integer fraction of k.

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

Example 4.7 *Closed loops in a fault propagation graph*

A fault propagation graph is illustrated in Fig. 4.9.
The directed adjacency matrix is D in Eq. (4.3). Powers of the adjacency matrix are

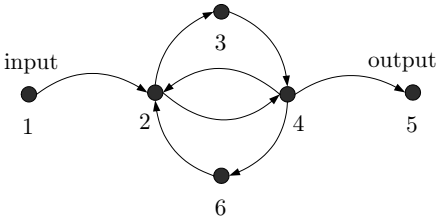


Fig. 4.9. A fault propagation graph example. One vertice is input (1), another is output (5).

$$\boldsymbol{D} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \quad \boldsymbol{D}^2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{4.3}$$

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

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

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

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

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

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

4.6 Fault propagation in closed loops

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

$$\boldsymbol{e}_{ci} \leftarrow (\boldsymbol{M}_i^f \;\; \boldsymbol{I}) \otimes \begin{pmatrix} \boldsymbol{f}_{ci} \\ \boldsymbol{e}_{c1} \end{pmatrix}.$$

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

$$\boldsymbol{e}_{ci}^{+} \leftarrow \begin{cases} \boldsymbol{M}_i^f \otimes (f_{ci}) & \text{if and only if} \quad \boldsymbol{e}_{c1}^{-} = \text{“}0\text{”} \\ \text{“}\mathbf{1}\text{”} & \text{if and only if} \quad \boldsymbol{e}_{c1}^{-} = \text{“}\mathbf{1}\text{”}, \end{cases}$$

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

When a closed logical loop is present, we hence need to cut an appropriate connection within the loop and investigate whether a “true” signal into the broken connection will produce a “true” at the other end of the cut. If this is the case, the loop is a tautology and can be eliminated. If the “true” in produces a “false” at the other end of the cut, the logical loop is unstable and no steady state solution exists. We then need to define the input of the cut as a new input in our failure mode and effects analysis description of the system, and analyse the propagation of an imagined “fault” condition = “true” from this point.

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

Since the existence of closed loops is essential to fault propagation analysis, a graph-theory tool will be presented to determine the existence and location of such loops.

4.6.1 Cutting the closed fault propagation loop

The directed adjacency matrix $\boldsymbol{D}$ and powers of $\boldsymbol{D}$ up to degree k shows both which vertices of the FPA graph are parts of closed loops, if any, and it informs on the paths from input to output. When a logical loop cut has to be made, is should be made such that the path from input to output are not interrupted, while cutting the relevant loop(s). The cut is conducted by cutting an edge, defining a new input and output as needed. The variables associated with the extra (new) input and output vertices is given by the variables associated with the edge that was cut. Logic analysis of the system is carried out using the new input as additional faults. The new output is observed. If the variable (effects) at the output are identical with the input, the

relation is a tautology and can be removed from the analysis. If the result is a logic contradiction, the new input needs to remain in the analysis, but without closing the loop again.

In conclusion, analysis of closed loops can always be done by extending the system with auxiliary faults.

Example 4.8 *Ship track control - fault propagation*

The propagation of faults from the track error sensor to the track controller are investigated in this example.

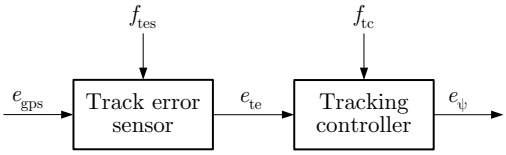


Fig. 4.10. Propagation of faults through the track error sensor and track controller

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

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

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

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

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

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

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

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

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

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

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

The resulting fault propagation is then

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

4.6.2 Assessment of the severity of the fault effects

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

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

4.6.3 Decision about fault handling

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

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

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

4.7 Fault tolerance analysis

Fault tolerance is defined as the possibility of achieving a given (set of) objective(s) in the presence of a given (set of) fault(s). In the generic model, objectives and services are associated with each use-mode. Thus, the system is fault tolerant in the current use-mode as long as services which allow to achieve the current objectives are available in this use-mode.

Therefore, the analysis of fault tolerance rests on three points.

1. Are there services which allow to achieve the current objectives?
2. How are these services to be managed when faults occur?
3. How are the use-modes to be managed when faults occur?

4.7.1 Relation between services and objectives

The generic component model describes the normal behaviour of the system components, at any hierarchical level, since high-level components are built, following a bottom-up approach, from the aggregation of low-level ones.

At the highest hierarchical level, the system itself is modelled as a single component, which aggregates all the elementary components, and whose services correspond to the missions or objectives that it has to achieve. Different aggregation paths can be followed between the field level (associated with elementary components) and the system level. A natural way of building the bottom-up aggregation procedure is to make use of the intuitive decomposition of the system into subsystems whose functions (and services) can be clearly defined.

System pyramidal decomposition. Hierarchical decomposition splits a system into a set of subsystems, which can themselves be further decomposed, each subsystem being associated with a clear functional viewpoint. For example, a chemical process can be decomposed into

- a subsystem which aims at controlling the pH,
- a subsystem which aims at controlling the level,
- a subsystem which aims at controlling the temperature.

The pH control subsystem may be further decomposed into the valve controlling the acid inflow, the valve controlling the base inflow and the stirr motor.

Since some components may belong to several subsystems, it is convenient to use a pyramidal decomposition (Fig. 4.5) whose number of levels is decided by the designer so as to obtain the view of the system which suits him best. Let $l = 1$ be the lowest decomposition level (the level of the field components) and $l = n$ be the highest decomposition level (the level of the system itself).

In a pyramidal decomposition, the following properties hold ($l \geq 2$):

- Each component of level $l-1$ belongs to at least one component of level l.
- Any component of level l includes at least one component of level l-1.

High-level services. Let C_l be the set of components modelled at level l, $(l = 1, ..., n)$, and let $c \in C_l$ and $\gamma \in C_{l-1}$ $(l \geq 2)$. Remember that the services of high-level components are behaviours which use the services of the low-level components they aggregate. Of course, not any combination of low-level services makes sense, and it is necessary, for the subsystems of the pyramidal decomposition to be consistent, that component $\gamma \in C_{l-1}$ is included in component $c \in C_l$ only if at least one service of component γ is used in at least one program which defines a service of component c associated with a known functional interpretation. Indeed, associating elementary services to define a high-level service is always realised to answer a functional requirement of the application. The system pyramidal decomposition is in this case, a real help to define high-level services from low-level ones.

Let $S(\gamma)$ be the services offered by a component $\gamma \in C_{l-1}$ and let $c \in C_l$ be the aggregation of a set Γ of such components. Let

$$\begin{aligned} cons\,(c) &= \cup_{\gamma \in \Gamma} \cup_{s \in S(\gamma)} cons\,(s) \\ prod\,(c) &= \cup_{\gamma \in \Gamma} \cup_{s \in S(\gamma)} prod\,(s). \end{aligned}$$

Note that $cons(c) \cap prod(c)$ may be non-empty, since some components in Γ may consume variables produced by some other ones. Note also that any relation between a subset of variables of $cons(c)$ and a subset of variables of $prod(c)$ can be obtained as the result of a program using the services of $\cup_{\gamma \in \Gamma} S(\gamma)$. If there exists such a relation which makes sense from a functional point of view in the system, then the subsystem $c \in C_l$ can be created at level l and the procedure which establishes such a relation can be defined as a service of the aggregated component c. Note finally that there might exist several subsets of components in Γ and several procedures which establish the same relation between the above mentioned variables. Then, the service exists under several versions. The set of all the versions of a service which can be obtained by aggregation of lower level ones can be found in a rather automated way, for simple kinds of programs composed of sequences and parallel executions.

Once the services of the subsystems have been determined at any level (including the overall system level), the ordering of the versions is left to the designer.

Example 4.9 *High-level regulation service*

Consider the three following low-level components:

- γ_1 is the temperature sensor, whose measurement service is defined by

  $$< \theta,\, \hat{\theta},\, f_1,\, rqst_1,\, enable_1,\, res_1 >$$

  where θ is the actual temperature, $\hat{\theta}$ is its estimate provided by the sensor, f_1 is the procedure which is used to produce the estimate: $\hat{\theta} = f_1(\theta)$. The request, enabling conditions and resources are not of interest here.

- γ_2 is the controller, whose computation service is defined by

  $$< (\hat{\theta},\, \theta^*),\, u,\, f_2,\, rqst_2,\, enable_2,\, res_2 >$$

  where θ^* is the temperature set-point, u is the control signal, and f_2 is the procedure which is used to produce the control signal: $u = f_2(\hat{\theta},\, \theta^*)$.
- γ_3 is the actuator, whose heating service is defined by

  $$< u,\, \pi,\, f_3,\, rqst_3,\, enable_3,\, res_3 >$$

  where u is the control signal, π is the delivered heating power, and f_3 is the procedure which is used to produce the heating power: $\pi = f_3(u)$.

Considering the set $\Gamma = \{\gamma_1,\, \gamma_2,\, \gamma_3\}$ as a candidate for aggregation into one higher-level component c, the above sets are $cons\,(c) = \left\{\theta,\, \hat{\theta},\, \theta^*,\, u\right\}$ and $prod\,(c) = \left\{\hat{\theta},\, u,\, \pi\right\}$, from which it is seen that the simple program sequence "measure, compute, actuate" can provide a relation between θ, θ^* and π, whose functional interpretation is of course that of a "regulation" service.

Moreover, note that if the measurement service is provided e.g. under four versions (as in the following example), and the heating service is provided under two versions, then the regulation service is provided under eight different versions. □

4.7.2 Management of service versions

Consider a service s at the system level. It is a set of pre-ordered versions $s = \left\{s^j,\, j \in J(s)\right\}$. Each version can be used for the same purpose, namely to achieve the system objective(s) in a given use-mode, but the pre-ordering expresses a preference between them.

Two conditions have to be fulfilled for service versions to be enabled at some given time. First, the service must belong to the list of services of the current use-mode. This is a straightforward condition, which insures that only services consistent with the current objectives can be run.

The second condition is related with faults. Suppose some resource from the set res^j is detected faulty by the fault diagnosis procedure at time t. Then, obviously, running the version s^j of the service s would produce incorrect values of the variables $prod$, and this should be forbidden, putting $enable^j = 0$. Note that this might be impossible, since faults might have ever-lasting delivery of some service as a consequence.

Example 4.10 *Unavailable and ever-lasting services*

Let V_open and V_close be two services delivered by an on/off valve, and suppose that the valve is blocked closed. Then, the service V_open becomes unavailable while the service V_close is permanent in time. □

Thus, the consequence of faults is that some services become permanent in time, while some others exist under a number of available versions which depend on the remaining, non-faulty resources. The status of these service obviously depends on the number of available versions, according to the following classification:

- at least one version is available: the service is available,
- no version is available: the service is unavailable.

Note that when more than one version is available, the lowest rank version of the service (which is the most preferred among the available ones) is to be run when the service is requested. Note also that the severity of the failure of a given resource with respect to the service can be evaluated by counting the number of versions which still are available after the failure has occurred. A resource for which this number is zero, or whose failure causes the service to run permanently in time is called a *critical resource*.

Example 4.11 *Management of service versions in an intelligent sensor*

Consider again the *measurement* service of the temperature sensor and suppose it includes two redundant transducers and an observer. Let

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

be the two local measurement equations, and

$$\hat{y}(t) = f(z_1(t), z_2(t))$$

be the observer algorithm, where $z_1(t),\ z_2(t)$ are remote measurements obtained from a local area network communication (LAN) system. The following table gives the different versions of the *measurement* service which are provided by this sensor, along with their ranking.

Class	Procedure	Fault situation
0	$y(t) = \frac{1}{\sigma_1+\sigma_2}(\sigma_2 y_1(t) + \sigma_1 y_2(t))$	no fault
1	$y(t) = y_1(t)$	T_1 ok., T_2 faulty, A/D ok.
1	$y(t) = y_2(t)$	T_2 ok., T_1 faulty, A/D ok.
2	$y(t) = \hat{y}(t)$	T_1 and T_2 or A/D faulty, LAN ok.

Suppose the measurement request is issued by the system clock according to some sampling period, and that the measurement service is consistent with the current use-mode. Then, it will be provided under version 0 if both transducers are operating well, and under version 1 in the presence of a single transducer fault (note that the two versions 1 are mutually exclusive, so that no conflict is possible). If the local Analog to Digital Converter (ADC) fails, version 2 can still be used, and the service will become unavailable only when local measurements and fieldbus communication will be all faulty. Remind that the local ADC was a critical resource when no observer was included in the sensor. Note also that version 2 of the measurement service might be much more unprecise than versions 0 and 1, thus the service would be degraded when this version is used. However, it would still be acceptable, otherwise version 2 should never have been included in the list of versions by the design engineer. □

4.7.3 Management of operation modes

Remember that the set of services is organised into use-modes, whose behaviour is described by a deterministic automaton. Let $A(M, \tau, m^0)$ be the use-mode automaton at the system level

- $M = \{m_i,\, i \in I_m\}$ is the set of the use-modes. Remember that each of these modes m_i is associated with the set of the services $S_i \subseteq S$ by which it is defined,
- $\tau = \{\tau_{ij},\, i,\, j \in I_m\}$ is the set of transitions,
- m^0 is the initial use-mode.

Critical services. In this chapter, use-modes have been further associated with objectives which have to be fulfilled thanks to those services. Let O_i be a set of objectives associated with use-mode m_i. As long as the set of services $S_i \subseteq S$ associated with m_i are available, the objectives O_i can obviously be achieved (otherwise, the component would be inconsistently-designed). Note that this is true, by definition, whatever the version of the service. Versions of high rank provide degraded service, thus achieving the objective in a degraded but still acceptable manner. If not, the versions of that rank should not have been included in the list of possible versions of the service.

Suppose now that some fault has occurred such that some services of S_i become unavailable or run permanently. Then some objectives of O_i might become impossible to achieve. Let *critical services* be services whose unavailability or permanent running implies that at least one objective of the mode to which they belong cannot be achieved. Then, the $A(M,\, \tau,\, m^0)$ automaton model is extended as follows: $M = \{m_i,\, i \in I_m\}$ is the set of the use-modes. Each mode m_i is associated with the set of objectives O_i and the set of the services $S_i \subseteq S$ which is decomposed into $S_i = S_i^c \cup S_i^{nc}$, where S_i^c are the critical and S_i^{nc} are the uncritical ones.

Example 4.12 *Critical service in the single-tank system*

Consider the single-tank system given in the introduction, used in a food industry batch production process, and suppose the current use-mode is UM_2: processing the batch. In that use-mode, the system objective is to regulate the temperature, and the regulation service is thus a critical resource, since UM_2 objective cannot be achieved if this service is lost. □

Staying in a mode. Consider the system operation in a given current use-mode, and suppose faults occur which cause the loss or permanent running of services (remember that as long as at least one service version is available, the service is not lost). When non-critical services are lost or run permanently, the system can obviously remain in the current use-mode, since this use-mode objectives can still be achieved - eventually in a degraded manner - and thus the system is fault tolerant with respect to the current objectives and the current fault situation.

On the contrary, when critical services of the current use-mode are lost or run permanently, the objectives associated with that use-mode can no longer be achieved, and the system is to be given other objectives. This strategy is called an objective reconfiguration strategy. The only way it can be implemented is by firing a transition towards another use-mode, whose objectives will become the current ones (for example change the production recipe, or stop the production and transfer the system to a safe state, in which maintenance can be undertaken).

In general, several other use-modes can be reached from the current one, and the choice of the destination use-mode (i.e. of the new system objectives) is a difficult decision problem, which has to be considered in the system design stage. Unless the system objectives can be ranked according to a total ordering relation, the solution to that problem can in general only be partially automated, thus leaving a very important role to human operators in fault situations.

Transitions between modes. When objective reconfiguration is necessary, the system is commanded to another use-mode whose objectives will become the new ones. The system should, obviously, be able to achieve these new objectives, which means that in the destination use-mode, no critical service is unavailable nor is permanently running as a result of the current fault situation.

This remark is also valid when the transition does not follow from an objective reconfiguration strategy but from the normal operation of the system.

4.8 Exercises

Exercise 4.1 *Fault propagation analysis for industrial actuator*

Consider the component block diagram of the position servo shown in Fig. 4.11. The blocks in the figure are motor, amplifier, controller, potentiometer, gear and reference.

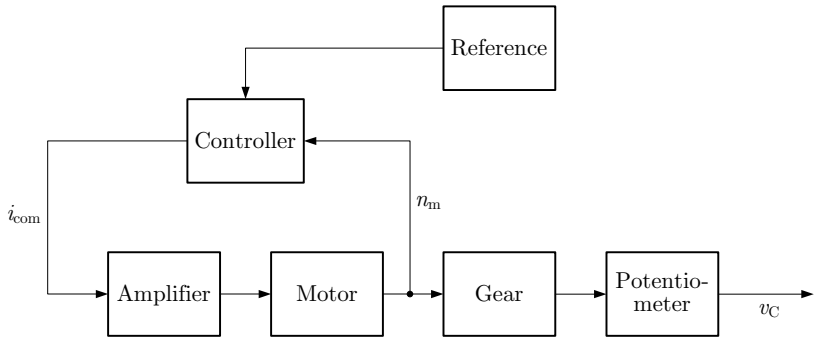


Fig. 4.11. Component diagram for speed loop part of the industrial actuator

1. Construct a fault propagation matrix M_p for the potentiometer (use the FMEA matrix from p.68). Use angle as input and voltage v_C at terminal C as output. Consider only the fault f_{p1}, which indicates the broken wire at C.
2. Using the above figure, define the component architecture in form of the directed adjacency matrix $\boldsymbol{D}$.
3. Determine which node is an input node from the adjacency matrix.

4. Determine the number of closed loops and the length of these.
   The combined fault propagation matrix for motor and amplifier is defined by input and internal faults:

f_{a1}	low gain in amplifier
f_{m1}	brush partial disconnect
e_{a1}	low input command
e_{a2}	high input command
e_{a3}	fluctuating input command

   output:

e_{n1}	low output speed
e_{n2}	high output command
e_{n3}	fluctuating output speed
e_{n4}	output speed not related to input command

   $$\boldsymbol{M}_{am} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

   Without further explanation, assume that the gear has the propagation matrix

   $$\boldsymbol{M}_{g} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

5. Determine the fault propagation from i_{com} to θ_m, using $\boldsymbol{M}_{am}, \boldsymbol{M}_{g}$ and $\boldsymbol{M}_{p}$.
6. Compute the inverse fault propagation matrix, ie. observe the end effects on θ_m and see which faults or input effects could cause the end effects. □

Exercise 4.2 *Component-based analysis of battery charger*
A battery charger component diagram is given as shown in Fig. 4.12. The behaviours of the individual components are not known in detail but are given as

$$\begin{array}{lrcl}
\textit{Power stage mean current:} & I & = & d \cdot \max(V_{\mathrm{sup}} - V_{\mathrm{bat}}, 0) \\
\textit{Current control:} & d & = & f_i(I_{\mathrm{com}} - I_{\mathrm{mes}}) \\
\textit{Voltage control:} & I_{\mathrm{com}} & = & f_v(V_{\mathrm{ref}} - V_{\mathrm{mes}}) \\
\textit{Current sensor:} & I_{\mathrm{mes}} & = & I \\
\textit{Voltage sensor:} & V_{\mathrm{mes}} & = & V_{\mathrm{bat}} \\
\textit{Harness:} & I_{\mathrm{bat}} & = & I \\
\textit{Battery voltage:} & V_{\mathrm{bat}} & = & \frac{\alpha_{\mathrm{bat}}}{C_{\mathrm{bat}}} \int_0^t I_{\mathrm{bat}}(t)\, dt, \quad \alpha_{\mathrm{bat}} \simeq 0.7.
\end{array} \tag{4.4}$$

1. Define one fault for each of the components: PWM converter, controller block (current and voltage control), current sensor, voltage sensor. Assume that the supply voltage and battery cannot fail.

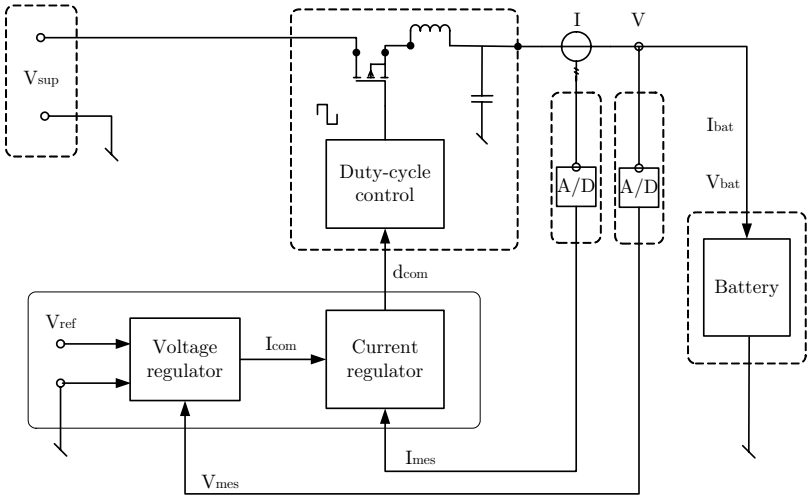


Fig. 4.12. Component diagram of battery charger

2. Determine the fault propagation matrices for these components.
3. Determine the closed logical loops in the battery charger.
4. Cut the loop at the signal d_{com} and determine whether the logical loop has a solution. If not, define d_{com} as an ancillary input.
5. Determine the end effects (on I_{bat} and V_{bat}) for the faults you defined.
6. Express the inverse propagation and list the end effects.
7. Suggest how a fault in the current sensor could be accommodated. □

4.9 Bibliographical notes

Basic bibliographical notes on architecture and generic component models have been given in Chapter 2. Further modelling extensions have been provided, for the description of complex interconnected systems, by the bottom-up procedure which allows to build high-level devices by the interconnection of low-level ones. This procedure has been developed in [31], [32].

Fault detection and hierarchical data validation has been introduced in [7], [40], [228], while alarm filtering applications have been considered in [190].

Failure modes and effect analysis is a classical and widely used tool in industry [93]. Presentation of a matrix formulation suited for computational treatment of FMEA schemes was first presented by [122]. The fault propagation analysis was proposed in [15] and further elaborated in [25]. A prototype computational tool was develped in [17].

Fault tolerance in distributed systems is mainly based on modular models [108]. The generic component model has been used for defining the faulty resources management and the mode management layers [10], [73]. The reconfigurability analysis has been developed more recently [74], [75], [150], [229].

Systematic analysis of fault propagation [15], [25] was shown to be an essential tool for determination of severity of fault effects and for assessment of remedial actions early in the

design phase. A semantics for services based on generic component models was developed in [76], [231] and a graphic analysis was found to be very useful.

Chapter 5

Structural analysis

This chapter uses the structure graph to describe the direct interactions among the signals. This graph is used to analyse the redundancies which can be exploited for fault diagnosis and control reconfiguration. Faults are interpreted as violation of constraints. The analysis shows how component faults, which imply the violation of single constraints, can be found by defining and utilising appropriate redundancy relations.

5.1 Introduction

This chapter investigates the structural properties of dynamical systems by analysing their structural model. The structural model of a system is an abstraction of its behaviour model in the sense that only the structure of the constraints, i.e. the existence of links between variables and parameters is considered and not the constraints themselves. The links are represented by a *bi-partite graph*, which is independent of the nature of the constraints and variables (quantitative, qualitative, equations, rules, etc.) and of the value of the parameters. This indeed represents a qualitative, very low level, easy to obtain, model of the system behaviour.

Structural analysis is concerned with the properties of the system structure model, which resorts to the analysis of its bi-partite graph. As this graph is independent of the value of the system parameters, structural properties are true almost everywhere in the system parameter space.

In spite of their simplicity, structural models can provide many useful information for fault diagnosis and fault-tolerant control design, since structural analysis is able to identify those components of the system which are – or are not – monitorable, to provide design approaches for analytic redundancy based residuals, to suggest alarm filtering strategies, and to identify those components whose failure can be tolerated through reconfiguration.

The main assumption is that each component is described by one or several constraints. Hence, the violation of at least one constraint indicates that this component is faulty. Structural properties lead to analytic redundancy based residuals, which provide sufficient conditions for the existence of some fault within some component and, thus, results in a set of possibly faulty components.

Due to this assumption, merely the model of the nominal system is considered in this chapter and no explicit fault model is necessary.

In this chapter, structural properties of interest are

- the identification of the monitorable part of the system, i.e. the subset of the system components whose faults can be detected and isolated,
- the possibility to design residuals which meet some specific fault diagnosis requirements, namely which are robust (i.e. insensitive to disturbances and uncertainties), and structured (i.e. sensitive to certain faults and insensitive to others),
- the existence of reconfiguration possibilities in order to estimate (respectively to control) some variables of interest in case of sensor, actuator or system component failures.

Answers to these questions are provided by the analysis of the system structural graph and its canonical decomposition. In order to introduce the canonical decomposition, matchings on a bi-partite graph are first presented, and their interpretation is given, introducing the idea of causality which provides the bi-partite graph with an orientation. Then the canonical decomposition of the system structural graph is presented, and structural observability and controllability issues are discussed. The design of fault diagnosis systems is addressed by the determination of robust and structured residuals, which can be designed for those subsystems in which some redundancy is present. Finally, fault tolerance issues consider the possibility to reconfigure the system in case of component failures, which rests on the permanence of the observability and controllabilty properties of the non-failed part of the system.

5.2 Structural model

5.2.1 Structure as a bi-partite graph

This section introduces the structural model of a system as a bi-partite graph which represents the links between a set of variables and a set of constraints. It is an abstraction of the behaviour model, because it merely describes which variables are connected by which constraints, but it does not say how these constraints look like. Hence, the structural model presents the basic features and properties of a system that are independent of its parameters.

The behaviour model of a system is defined by a pair $(\mathcal{C}, \mathcal{Z})$ where $\mathcal{Z} = \{z_1, z_2, ..., z_N\}$ is a set of variables and parameters, and $\mathcal{C} = \{c_1, c_2, ..., c_M\}$ is a set of constraints. According to the granularity of the variables (quantitative,

qualitative, fuzzy) and of the time (continuous, discrete), the constraints may be expressed in several different forms like algebraic and differential equations, difference equations, rules, etc.

Usually, two basic assumptions express the fact that a model defined by some set of constraints is well formed.

Assumption 5.1

(a) All the constraints in $\mathcal{C}$ are compatible.

(b) All the constraints in $\mathcal{C}$ are independent.

Assumption 5.1(a) means that the set of the constraints is associated with a sound model, namely a model whose set of solutions is not empty. In other words, the constraints do not carry any contradiction. This is of course assumed to hold in the sequel.

Assumption 5.1(b) on the independence of the constraints means that the model is minimal in the sense that no constraint defines (at least locally) the same set of solutions as another one, or more generally that there does not exist in $\mathcal{C}$ two different subsets of constraints $\mathcal{C}'$ and $\mathcal{C}''$ such that

$$V(\mathcal{C}') \subseteq V(\mathcal{C}'')$$

holds, where $V(\mathcal{C})$ is the set of solutions associated with the constraint set $\mathcal{C}$. It will be seen that this assumption may hold or not, depending on the redundancy which is present in the system.

Example 5.1 *Dependent constraints*

Consider the two constraints

$$\begin{aligned} c_1 &: \quad z_1 - 1 = 0 \\ c_2 &: \quad (z_1 - 1)(z_2 - 1) = 0. \end{aligned}$$

They are obviously not independent, since one has $V(c_1) \cap V(c_2) = V(c_1)$. In fact, constraint c_1 is enough to describe the set of the system solutions, and one has

$$c_1 \text{ true } \Rightarrow c_2 \text{ true } \square$$

Structure graph. The structure of a system can be represented by a bi-partite graph. A graph is bi-partite if its vertices can be separated into two disjoint sets $\mathcal{C}$ and $\mathcal{Z}$ in such a way that every edge has one endpoint in $\mathcal{C}$ and the other one in $\mathcal{Z}$.

Definition 5.1 (Structural model)
The structural model (or the structure) of the system $(\mathcal{C}, \mathcal{Z})$ is a bi-partite graph $(\mathcal{C}, \mathcal{Z}, \mathcal{E})$ where $\mathcal{E} \subset \mathcal{C} \times \mathcal{Z}$ is the set of edges defined by:

$$(c_i, z_j) \in \mathcal{E} \text{ if the variable } z_j \text{ appears in the constraint } c_i.$$

Note that the bi-partite graph is an undirected graph, which can be interpreted as follows: All the variables and parameters connected with a given constraint vertex

have to satisfy the equation or rule this vertex represents. This graph allows to represent the structure of rather general models including both differential and algebraic constraints.

Example 5.2 *Structure of a system described by a differential-algebraic model*

Let

$$\begin{aligned} \mathcal{Z} &= \boldsymbol{x}_a \cup \boldsymbol{x}_d \cup \boldsymbol{u} \cup \boldsymbol{y} \\ \mathcal{C} &= \boldsymbol{g} \cup \boldsymbol{h} \cup \boldsymbol{m}, \end{aligned}$$

where $\boldsymbol{x}_a$ is the set of variables which appear only algebraically, and $\boldsymbol{x}_d$ are variables whose derivative obeys some differential constraints $\boldsymbol{g}$. A differential-algebraic system model is given by

$$\dot{\boldsymbol{x}}_d = \boldsymbol{g}(\boldsymbol{x}_d, \boldsymbol{x}_a, \boldsymbol{u}) \tag{5.1}$$

$$\boldsymbol{0} = \boldsymbol{m}(\boldsymbol{x}_d, \boldsymbol{x}_a, \boldsymbol{u}) \tag{5.2}$$

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}_d, \boldsymbol{x}_a, \boldsymbol{u}). \tag{5.3}$$

Note that it is possible to define a separate set of variables $\dot{\boldsymbol{x}}_d$ and a separate set of constraints

$$\dot{x}_i(t) - \frac{d}{dt}x_i(t) = 0, \qquad (i = 1, ..., n) \tag{5.4}$$

so that the system is represented by

$$\begin{aligned} \mathcal{Z} &= \boldsymbol{x}_a \cup \boldsymbol{x}_d \cup \dot{\boldsymbol{x}}_d \cup \boldsymbol{u} \cup \boldsymbol{y} \\ \mathcal{C} &= \boldsymbol{g} \cup \boldsymbol{h} \cup \boldsymbol{m} \cup \frac{d}{dt}, \end{aligned}$$

where $\frac{d}{dt}$ stands for the differential constraints (5.4) and all the constraints (5.1), (5.2) and (5.3) are algebraic. The behaviour model of a dynamical system links present and past values of its variables (for discrete time systems) or variables and their time derivatives up to a certain order (for continuous-time systems). Giving two variables the names $\boldsymbol{x}(t)$ and $\dot{\boldsymbol{x}}(t)$ does not guarantee that the second one is the time derivative of the first one. This is only true thanks to the analyst's interpretation, and this fact has to be represented, for automatic treatment, by separate constraints like (5.4). □

In the sequel, bi-partite graphs will be used for the representation of the system structure.

In the following figures, the vertices of $\mathcal{Z}$ will be represented by circles while the vertices of $\mathcal{C}$ will be represented by bars. Note that the edges are not oriented. The incidence matrix of the bi-partite graph is the matrix whose rows and columns represent the set of constraints or variables, respectively. Every edge $(c_i, z_j) \in \mathcal{E}$ is represented by a "1" in the intersection of row c_i and column z_j.

Example 5.3 *Bi-partite graph of a linear system*

The behaviour model of a linear system described by four constraints $\{c_1, c_2, c_3, c_4\}$ with five variables $\{x_1, x_2, \dot{x}_1, \dot{x}_2, u\}$ is given by

$$\begin{aligned} c_1 &: \dot{x}_1 = \frac{dx_1}{dt} \\ c_2 &: \dot{x}_1 = ax_2 \\ c_3 &: \dot{x}_2 = \frac{dx_2}{dt} \\ c_4 &: \dot{x}_2 = bx_1 + cx_2 + du\,. \end{aligned}$$

Its structure graph has the incidence matrix

↗	u	x_1	x_2	$\dot{x}_1$	$\dot{x}_2$
c_1		1		1	
c_2			1	1	
c_3			1		1
c_4	1	1	1		1

leading to the bi-partite graph depicted in Fig. 5.1. □

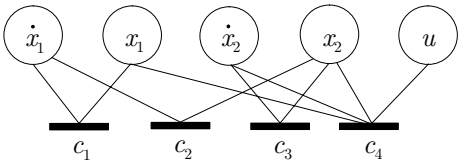


Fig. 5.1. Bi-partite graph of the linear system

Example 5.4 *Tank system*

Consider a tank system where the inflow is controlled via a level sensor and an electric pump and the outflow is realised through an output pipe (Fig.5.2).

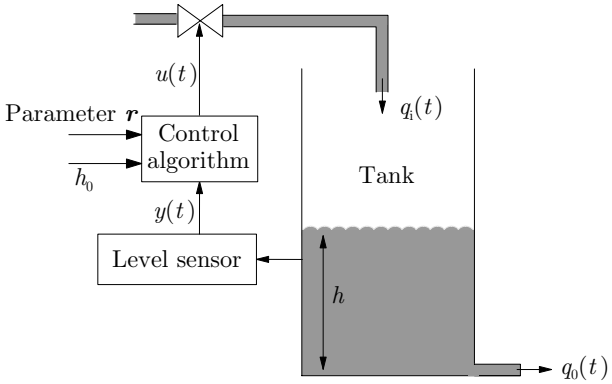


Fig. 5.2. Single-tank system

The system consists of the components {tank, input valve, output pipe, level sensor, level control algorithm}. A continuous-variable continuous-time model is given by the following constraints:

$$
\begin{array}{ll}
\text{Tank } c_1: & \dot{h}(t) = q_i(t) - q_o(t) \\
\text{Input valve } c_2: & q_i(t) = \alpha u(t) \\
\text{Output pipe } c_3: & q_0(t) = k\sqrt{h(t)} \\
\text{Level sensor } c_4: & y(t) = h(t) \\
\text{Control algorithm } c_5: & u(t) = \begin{cases} 1 \text{ if } y(t) \leq h_0 - r \\ 0 \text{ if } y(t) \geq h_0 + r. \end{cases}
\end{array}
\tag{5.5}
$$

u is the control variable, y the sensor output, h_0 the given set-point, and r and k are given parameters. h denotes the liquid level, q_i and q_o the flow into or out of the tank. α is a valve constant. Each component introduces one constraint. The separate constraint

$$c_6 : \dot{h}(t) = \frac{dh(t)}{dt}$$

expresses the fact that $\dot{h}(t)$ is the derivative of the level $h(t)$. Applying the definition to the behaviour model (5.5) of the tank system (without controller) leads to the following incidence matrix.

	Input/Output		Internal variables			
$\nearrow$	$u(t)$	$y(t)$	$h(t)$	$\dot{h}(t)$	$q_i(t)$	$q_o(t)$
c_1				1	1	1
c_2	1				1	
c_3			1			1
c_4		1	1			
c_6			1	1		

The structure graph corresponding to this incidence matrix is shown in Fig. 5.3. Every column of the matrix correspond to a circle-vertex and every row to a bar-vertex.

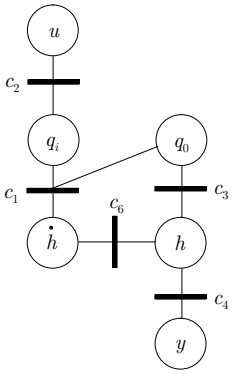


Fig. 5.3. Structure graph of the single-tank system

If the controller is introduced, the graph is extended by a new bar-vertex for c_5 and two new circle-vertices for h_0 and r. Furthermore, if the parameter k appearing in constraint c_3 is considered now as an important variable (rather than a fixed given parameter like the valve constant α) a circle-vertex is introduced for k and linked with c_3.

	Input/Output		Parameters			Internal variables			
$\nearrow$	$u(t)$	$y(t)$	h_0	r	k	$h(t)$	$\dot{h}(t)$	$q_i(t)$	$q_o(t)$
c_1							1	1	1
c_2	1							1	
c_3					1	1			1
c_4		1				1			
c_5	1	1	1	1					
c_6						1	1		

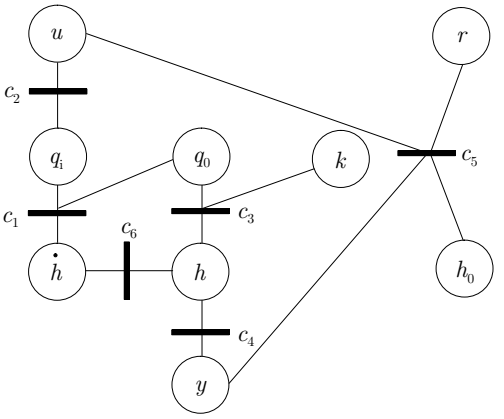


Fig. 5.4. Structure graph of the controlled tank

For simplicity, only the "ones" appear in this matrix and empty boxes are "zero". Figure 5.4 shows the extended graph. □

Remark 5.1 *Bi-partite graph versus digraphs*

For systems described by linear state and measurement equations, a popular structural representation uses the so-called directed graph (digraph), whose set of vertices is the set of the input, output and state variables and whose edges are defined by the following rules:

- an edge exists from vertice x_k (resp. from vertice u_l) to vertice x_i if and only if the state variable x_k (resp. the input variable u_l) really occurs in function f_i (i.e. $\frac{\partial f_i}{\partial x_k}$ - resp. $\frac{\partial f_i}{\partial u_l}$ - is not identically zero),
- an edge exists from vertice x_k to vertice y_j if and only if the state variable x_k really occurs in the function g_j. □

In the digraph representation edges are interpreted as "mutual influences" between variables: an edge from x_k (resp. from u_l) to x_i means that the time evolution of the derivative $\dot{x}_i(t)$ depends on the time evolution of $x_k(t)$ (resp. $u_l(t)$). Similarly, an

edge from x_k to y_j means that the time evolution of the output $y_j(t)$ depends on the time evolution of the state variable $x_k(t)$.

Example 5.5 *Digraph of a linear system*

The digraph which describes the structure of the system

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= ax_2 + bu \\ y &= x_1 \end{aligned} \tag{5.6}$$

is given by Fig. 5.5. □

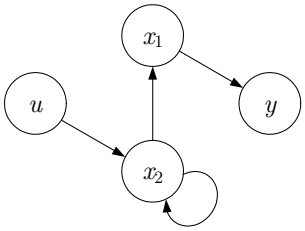


Fig. 5.5. Digraph of the linear system

5.2.2 Subsystems

Instead of considering the whole set of the constraints which describe the behaviour model of a system, it is sometimes convenient to consider only subsets of them. A subsystem is defined by the set of constraints together with the set of variables that occur in these constraints. This subsection introduces the vocabulary connected with subsets of the constraints.

Let $2^{\mathcal{C}}$ (respectively $2^{\mathcal{Z}}$) be the collection of all the subsets of $\mathcal{C}$ (respectively of all the subsets of $\mathcal{Z}$), and let $(\mathcal{C}, \mathcal{Z}, \mathcal{E})$ be the structure of the system $(\mathcal{C}, \mathcal{Z})$. Let Q be a mapping between a set of constraints and the set of variables used in these constraints:

$$\begin{aligned} Q: \quad & 2^C \to 2^{\mathcal{Z}} \\ & \phi \mapsto Q(\phi) = \{z \in \mathcal{Z};\ \exists c \in \phi \quad \text{s.t.} \quad (c,\, z) \in \mathcal{E}\}. \end{aligned} \tag{5.7}$$

Q associates with any subset of constraints ϕ the subset of those variables which intervene in at least one of them. Correspondingly, R associates a set of variables with a set of constraints where these variables appear:

$$\begin{aligned} R: \quad & 2^{\mathcal{Z}} \to 2^C \\ & \xi \mapsto R(\xi) = \{c \in \mathcal{C};\ \exists z \in \xi \quad \text{s.t.} \quad (c,\, z) \in \mathcal{E}\}. \end{aligned} \tag{5.8}$$

Example 5.6 *Q and R mappings for the tank example*

Consider the incidence matrix associated with the single-tank system (without controller).

$\nearrow$	h	$\dot{h}$	q_i	q_o	u	y
c_1		1	1	1		
c_2			1		1	
c_3	1			1		
c_4	1					1
c_5					1	1
c_6	1	1				

Examples for the mappings Q and R are:

$$\begin{aligned} Q(\{c_1,\, c_3\}) &= \left\{h,\, \dot{h},\, q_i,\, q_o\right\} \\ Q(\{c_5\}) &= \{u,\, y\} \\ R(\{q_i,\, q_o\} &= \{c_1,\, c_2,\, c_3\}\,. \quad \square \end{aligned}$$

Definition 5.2 (Subsystem)
A subsystem is a pair $(\phi,\, Q(\phi))$, where $\phi \in 2^{\mathcal{C}}$. The sub-graph that is related with subsystem $(\phi,\, Q(\phi))$ is its structure.

In this definition, a subsystem is any subset of the system constraints ϕ along with the related variables $Q(\phi) \in \mathcal{Z}$. For example, in the system above, the pair

$$(\{c_1,\, c_3\}\,,\, \left\{h,\, \dot{h},\, q_i,\, q_o\right\})$$

is a subsystem, and its structure is described by the subgraph

$\nearrow$	h	$\dot{h}$	q_i	q_o
c_1		1	1	1
c_3	1			1

There are no specific requirements on the choice of the elements in ϕ, and $2^{\mathcal{C}}$ contains all possible subsystems. Of course, only some of them are of interest in applications:

- First, subsystems can be associated with some physical interpretation. Complex systems are often decomposed into subsystems which have a physical or a functional meaning, e.g. the boiler in a steam generator, the instrumentation scheme in a closed-loop control system, etc. These subsystems are associated with subsets of constraints in the system model, so that the fault of one or several subsystem component(s) results in some of these constraints being changed.
- Second, subsystems can be associated with special properties. For example, fault diagnosis is possible only for subsystems which exhibit redundancy properties.

5.2.3 Structural properties

The structural model of a system is an abstraction of its behaviour model. Two systems which have the same structure are said to be structurally equivalent. Since structural properties are properties of the structural graph, they are shared by all the systems which have the same structure. In particular, systems which only differ by the value of their parameters are structurally equivalent, thus making structural properties independent of the values of the system parameters.

From structural equivalence considerations, it can be seen that the structure of a system is indeed independent of the form under which the constraints are expressed. For example, suppose that the level sensor in the single-tank system does not provide an analog output but a quantised one. Then its operation is described by the following table, where α, β, γ are given constants associated with the sensor:

h	$\in [0, \alpha[$	$\in [\alpha, \beta[$	$\in [\beta, \gamma[$	$\geq \gamma$
y	empty	low	medium	high

It can easily be understood that the structure of the system is exactly the same using the analog or the symbolic sensor.

Of course, actual system properties may differ from structural ones, as can be seen from the following simple example. Let

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} a(\boldsymbol{\theta}) & c(\boldsymbol{\theta}) \\ b(\boldsymbol{\theta}) & d(\boldsymbol{\theta}) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

be the model of a system where y_1 and y_2 are known, $\boldsymbol{\theta} \in \mathbb{R}^q$ is some parameter vector, and the observability of the unknowns x_1, x_2 is investigated. The system is observable if the matrix is invertible. The structural condition is that no row (no column) of this matrix contains only zeros. This is necessary, but not sufficient, since the determinant $\Delta(\boldsymbol{\theta}) = a(\boldsymbol{\theta})\, d(\boldsymbol{\theta}) - b(\boldsymbol{\theta})\, c(\boldsymbol{\theta})$ might be zero, so that the property would not hold for the actual system although the structural property holds. Two cases can be distinguished:

1. In the first case, parameters $\boldsymbol{\theta}$ *always* satisfy the relation $\Delta(\boldsymbol{\theta}) = 0$ and thus the structural property is *never* translated into an actual property. This is excluded in structural analysis. First, an algebraic relation like $\Delta(\boldsymbol{\theta}) = 0$ is always supposed to define a manifold of dimension at most $q - 1$, which means that it cannot be satisfied by any $\boldsymbol{\theta} \in \mathbb{R}^q$ or, in other words, that it does not boil down to $0 = 0$. Second, the parameters are always supposed to be independent, which means that they span the whole space $\mathbb{R}^q$. If this was not the case, equation $\Delta(\boldsymbol{\theta}) = 0$ should have been included in the system model.

2. In the second case, the parameters $\boldsymbol{\theta}$ *of the system under investigation* satisfy the relation $\Delta(\boldsymbol{\theta}) = 0$, and thus the structural property is not translated into

an actual property *for that particular system*. Structural analysis however provides interesting conclusions, since under mild assumptions about functions a, b, c, d there always exists a parameter vector $\boldsymbol{\theta}'$ in the neighbourhood of $\boldsymbol{\theta}$ for which the actual property coincides with the structural one.

In conclusion, actual properties are only potential when structural properties are satisfied. They can certainly not be true if the structural properties are not satisfied. In other words, structural properties are properties which hold for actual systems almost everywhere in the space of their independent parameters. It is extremely unlikely that the system under consideration has a parameter vector for which the structural properties do not hold.

5.2.4 Known and unknown variables

The system variables and parameters can be decomposed into known and unknown ones. System input and output are examples of variables that are usually known. Similarly, parameters which enter the model and have been previously identified are known. Known variables are available in real time and they can directly be used in fault diagnosis or fault-tolerant control algorithms. Unknown variables are not directly measured, though there might exist some way to compute their value from the values of known ones. In the tank example, the last four columns of the incidence matrix $\left\{h,\, \dot{h},\, q_i,\, q_o\right\}$ correspond to unknown variables, while the first five ones correspond to known variables and parameters $\{u,\, y,\, h_0,\, r,\, k\}$.

Following that decomposition, the set of the variables is partitioned into

$$\mathcal{Z} = \mathcal{K} \cup \mathcal{X},$$

where $\mathcal{K}$ is the subset of the known variables and parameters and $\mathcal{X}$ is the subset of the unknown ones. Similarly, the set of constraints is partitioned into

$$\mathcal{C} = \mathcal{C}_{\mathcal{K}} \cup \mathcal{C}_{\mathcal{X}},$$

where $\mathcal{C}_{\mathcal{K}}$ is the subset of those constraints which link only known variables and $\mathcal{C}_{\mathcal{X}}$ includes those constraints in which at least one unknown variables appears. $\mathcal{C}_{\mathcal{K}}$ is the largest subset of constraints such that $Q(\mathcal{C}_{\mathcal{K}}) \subseteq \mathcal{K}$. It can be noticed that the relations which define control algorithms belong to $\mathcal{C}_{\mathcal{K}}$ since they introduce constraints between the sensor output, the control objectives (set-points, tracking references, final states) and the control input, which are all known variables.

According to the decomposition of $\mathcal{Z}$ and $\mathcal{C}$, the graph $(\mathcal{C},\, \mathcal{Z},\, \mathcal{E})$ can be decomposed into two sub-graphs which correspond to the two subsystems $(\mathcal{C}_{\mathcal{K}},\, Q(\mathcal{C}_{\mathcal{K}}))$ and $(\mathcal{C}_{\mathcal{X}},\, \mathcal{Z})$. The behaviour model of the subsystem $(\mathcal{C}_{\mathcal{K}},\, Q(\mathcal{C}_{\mathcal{K}}))$ involves only known variables. In some further developments, it will be of interest to focus on the subsystem $(\mathcal{C}_{\mathcal{X}},\, \mathcal{Z})$ which is also called the *reduced structure graph.* This graph includes only those constraints that refer to at least one unknown variable $z_i \in \mathcal{X}$. A fundamental question of fault diagnosis concerns the determination of unknown variables from known variables by means of constraints. The question whether this

is possible or not does only depend on the structure of the subgraph $(\mathcal{C}_\mathcal{X},\ \mathcal{X},\ \mathcal{E}_\mathcal{X})$ that results from the reduced graph by deleting all known variables $z_i \in \mathcal{K}$ together with the corresponding edges. Therefore, in all further examples of structure graphs the known variables are marked grey.

Example 5.7 *Analysis of the structural graph of the tank system*

Consider the tank, whose structure graph is given in Fig. 5.4. Assume that only the input u and the output y are known signals and, furthermore, h_0, r and k are known parameters. Then the decomposition of the variable set

$$\mathcal{Z} = \{h,\ \dot{h},\ q_i,\ q_o,\ u,\ y,\ h_0,\ r,\ k\}$$

into known and unknown variables yields the sets

$$\mathcal{K} = \{u,\ y,\ h_0,\ r,\ k\}$$

and

$$\mathcal{X} = \{h,\ \dot{h},\ q_i,\ q_o\}.$$

By selecting all constraints whose variables are all in the set $\mathcal{K}$, the set $\mathcal{C}_\mathcal{K} = \{c_5\}$ is obtained. All other constraints comprise the set

$$\mathcal{C}_\mathcal{X} = \{c_1,\ c_2,\ c_3,\ c_4,\ c_6\}.$$

Obviously, $Q(\mathcal{C}_\mathcal{K}) = \mathcal{K}$ and

$$Q(\mathcal{C}_\mathcal{X}) = \{u,\ y,\ q_i,\ q_o,\ h,\ \dot{h}\}$$

holds. The bi-partite graph can be re-organised as follows:

	known					unknown			
$\nearrow$	u	y	h_0	r	k	h	$\dot{h}$	q_i	q_0
c_5	1	1	1	1					
c_1							1	1	1
c_2	1							1	
c_3					1	1			1
c_4		1				1			
c_6						1	1		

The known variables are in the left columns and the constraint that refers merely to known variables in the first row. The reduced structure graph which corresponds to the subsystem $\mathcal{C}_\mathcal{X},\ \mathcal{Z}$ is given by the lower part of the incidence matrix. As the variables h_0 and r do not appear in this part of the matrix, their columns are deleted:

	known			unknown			
$\nearrow$	u	y	k	h	$\dot{h}$	q_i	q_0
c_1					1	1	1
c_2	1					1	
c_3			1	1			1
c_4		1		1			
c_6				1	1		

This reduced graph is shown in Fig. 5.6.

For diagnosis, another decomposition of the variables into known and unknown ones is used. The parameters like k, h_0 and r are assumed to be fixed and, hence, ignored in the structure graph. The remaining variables represent signals some of which are measured and the others are unknown. Hence, for the tank system the fixed parameter k is deleted from the structure graph, which results in the following incidence matrix and in Fig. 5.7:

$\nearrow$	u	y	h	$\dot{h}$	q_i	q_0
c_1				1	1	1
c_2	1				1	
c_3			1			1
c_4		1	1			
c_6			1	1		

and the graph depicted in Fig. 5.7 result. □

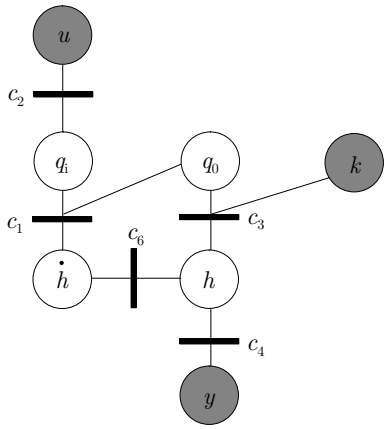


Fig. 5.6. Reduced structure graph of the tank system

5.3 Matching on a bi-partite graph

The basic tool for structural analysis is the concept of matching on a bi-partite graph, which is introduced in this section. In loose terms, a matching is a causal assignment which associates unknown system variables with the system constraint from which they can be calculated. Unknown variables which cannot be matched cannot be calculated. Variables which can be matched in several ways can be determined in different (redundant) ways, which provides a means for fault detection and a possibility for reconfiguration.

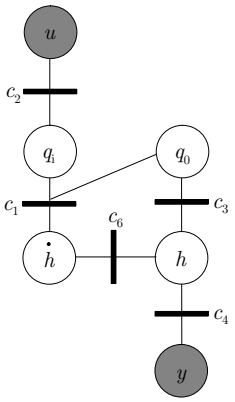


Fig. 5.7. Structure graph of the tank system used in diagnosis

5.3.1 Definitions

Let $(\mathcal{C},\ \mathcal{Z},\ \mathcal{E})$ be a bi-partite graph, $e \in \mathcal{E},\ e = (\alpha, \beta)$ be an edge which links the constraint α and the variable β, and $p_{\mathcal{C}}$ and $p_{\mathcal{Z}}$ be the two projections

$$\begin{aligned} p_{\mathcal{C}} \quad &: \quad \mathcal{E} \to \mathcal{C} \\ &\qquad e \mapsto p_{\mathcal{C}}(e) = \alpha \\ p_{\mathcal{Z}} \quad &: \quad \mathcal{E} \to \mathcal{Z} \\ &\qquad e \mapsto p_{\mathcal{Z}}(e) = \beta. \end{aligned}$$

The projection of the edge on the constraint set is $p_{\mathcal{C}}(e) = \alpha$ (the constraint node of the edge e) and the projection of the edge on the variable set is $p_{\mathcal{Z}}(e) = \beta$ (the variable node of the edge e).

Definition 5.3 (Matching)
A matching $\mathcal{M}$ is a subset of $\mathcal{E}$ such that the restrictions of $p_{\mathcal{C}}$ and $p_{\mathcal{Z}}$ to $\mathcal{M}$ are injective, i.e.

$$\forall e_1, e_2 \in \mathcal{M}:\ e_1 \neq e_2 \Rightarrow p_{\mathcal{C}}(e_1) \neq p_{\mathcal{C}}(e_2) \wedge p_{\mathcal{Z}}(e_1) \neq p_{\mathcal{Z}}(e_2).$$

This means that a matching is a subset of edges such that any two edges have no common node (neither in $\mathcal{C}$ nor in $\mathcal{Z}$). In general, different matchings can be defined on a given bi-partite graph, as illustrated by Fig. 5.8.

The set $\boldsymbol{M}$ of all matchings is a subset of $2^{\mathcal{E}}$, which is partially ordered by the set-inclusion order relation. Thus, maximal elements can be defined.

Definition 5.4 (Maximal matching)
A maximal matching is a matching $\mathcal{M}$ such that $\forall \mathcal{N} \in 2^{\mathcal{E}}$ with $\mathcal{M} \subset \mathcal{N}$, $\mathcal{N}$ is not a matching.

Thus, a maximal matching is a matching such that no edge can be added without violating the *no common node* property. Since the set of matchings $\boldsymbol{M}$ is only partially ordered, it follows that there is in general more than one maximal matching.

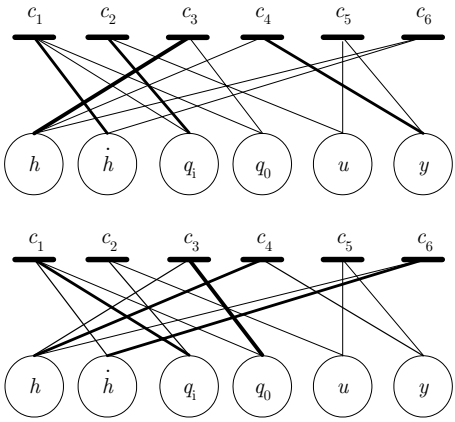


Fig. 5.8. Two possible matchings for the tank system: The edges $e \in \mathcal{M}$ are drawn by thick lines.

Let $\boldsymbol{M}^* \subseteq \boldsymbol{M}$ be the set of maximal matchings. Extending the definition of the projections $p_{\mathcal{C}}$ and $p_{\mathcal{Z}}$ to sets of edges (instead of one single edge),

$$\begin{aligned}
\pi_{\mathcal{C}} \quad &: \quad \boldsymbol{M} \to 2^{\mathcal{C}} \\
&\qquad \mathcal{M} \mapsto \pi_{\mathcal{C}}(\mathcal{M}) = \{c \in \mathcal{C};\ \exists e \in \mathcal{M} \text{ such that } \ c = p_{\mathcal{C}}(e)\} \\
\pi_{\mathcal{Z}} \quad &: \quad \boldsymbol{M} \to 2^{\mathcal{Z}} \\
&\qquad \mathcal{M} \mapsto \pi_{\mathcal{Z}}(\mathcal{M}) = \{z \in \mathcal{Z};\ \exists e \in \mathcal{M} \text{ such that } \ z = p_{\mathcal{Z}}(e)\},
\end{aligned}$$

each matching $\mathcal{M}$ is associated with the subset $\pi_{\mathcal{C}}(\mathcal{M})$ of its matched constraints, and with the subset $\pi_{\mathcal{Z}}(\mathcal{M})$ of its matched variables. Since no more than one constraint (respectively no more than one variable) can be associated with each edge of a matching, it follows that any matching (and therefore also maximal matchings) satisfies the following property

$$\forall \mathcal{M} \in \boldsymbol{M}^* \qquad \begin{aligned} &\pi_{\mathcal{C}}(\mathcal{M}) \subseteq \mathcal{C} \\ &\pi_{\mathcal{Z}}(\mathcal{M}) \subseteq \mathcal{X}, \end{aligned}$$

from which it follows that the relation

$$|\mathcal{M}| \leq \min\{|\mathcal{C}|\,,\ |\mathcal{Z}|\}$$

holds, where the bars $|.|$ denote the cardinality of the sets (which means the number of elements). Matchings for which the equality sign holds are called complete matchings.

Definition 5.5 (Complete matching)
A matching is called complete with respect to $\mathcal{C}$ if $|\mathcal{M}| = |\mathcal{C}|$ holds. A matching is called complete with respect to $\mathcal{Z}$ if $|\mathcal{M}| = |\mathcal{Z}|$ holds.

For a complete matching $\mathcal{M}$ on $\mathcal{C}$ (respectively on $\mathcal{Z}$), each constraint (respectively each variable) belongs to exactly one edge of the matching:

$$\forall c \in \mathcal{C}: \qquad \exists z \in \mathcal{Z} \text{ such that } (c,z) \in \mathcal{M}$$
$$\forall z \in \mathcal{Z}: \qquad \exists c \in \mathcal{C} \text{ such that } (c,z) \in \mathcal{M}.$$

A matching can be represented by selecting at most one "1" in each row and in each column in the incidence matrix of the bi-partite graph, and representing them by the "①" in the examples. Each selected "1" represents an edge of the matching. No other edge should contain the same variable (thus it is the only one in the row) or the same constraint (thus it is the only one in the column).

It is obviously possible to define matchings, maximal matchings, and complete matchings by considering either the whole structure of the system or only subgraphs of its structural graph, i.e. subsets of the constraints and variables instead of the whole sets. As known variables need not to be determined by some constraint, the matching is accomplished in the following for the subgraph containing all unknown variables rather than the whole structure graph. However, the incidence matrices and the graphical representations are usually given for the complete structure graph.

Example 5.8 *Matchings on the reduced structure graph of the tank*

In order to illustrate the notion of maximal and complete matchings, consider the reduced structure graph of the single-tank example. Only the unknown signals and the constraints among them are concerned with. The edges of a matching are identified by a thick line in the graph representation and by "①" in the following incidence matrices.

↗	h	$\dot{h}$	q_i	q_0
c_1		1	①	1
c_2			1	
c_3	①			1
c_4	1			
c_6	1	①		

Matching (a)

↗	h	$\dot{h}$	q_i	q_0
c_1		1	1	1
c_2			①	
c_3	1			①
c_4	①			
c_6	1	①		

Matching (b)

↗	h	$\dot{h}$	q_i	q_0
c_1		①	1	1
c_2			①	
c_3	1			①
c_4	1			
c_6	①	1		

Matching (c)

As in a matching, unknown variables are associated with a constraint by means of which they can be determined, an intuitive graphical representation is given in Fig. 5.9 where the constraints are drawn on the left-hand side and the variables on the right-hand side. The thick edges connect the variables with the constraints by which the variable can be calculated. The graphs are the same as in Fig. 5.7.

Figure 5.9 a) shows an incomplete matching. It is not complete with respect to the constraints since constraints c_2 and c_4 are not matched, nor is it complete with respect to the

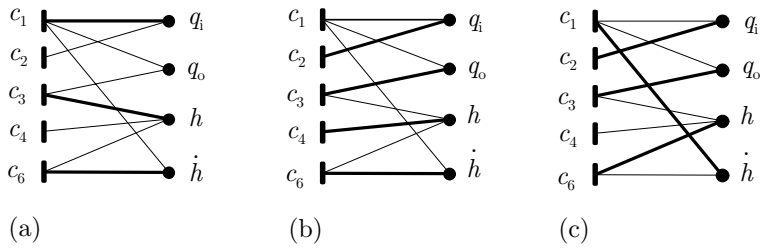


Fig. 5.9. An incomplete (a) and two complete ((b), (c)) matchings

variables since q_o is not matched. However, no edge can be added to the matching without violating Definition 5.3.

Two complete matchings with respect to the unknown variables are shown in Fig. 5.9 b) and Fig. 5.9 c). There is no matching that is complete with respect to $\mathcal{C}_{\mathcal{X}}$, because the number of constraints is larger than the number of variables. Note that it is not guaranteed that a complete matching exists, either with respect to $\mathcal{C}_{\mathcal{X}}$ or to $\mathcal{X}$. □

5.3.2 Oriented graph associated with a matching

Defining a matching on a structure graph introduces some orientations of the edges which, until now, were undirected. Constraints which appear in the system description have no direction, because all variables have the same status. For example, the tank constraint

$$c_1 : q_i(t) - q_o(t) - \dot{h}(t) = 0 \tag{5.9}$$

can be used to compute any of the three variables whenever the two other variables are known. It is written in the non-oriented form to stress that the constraint itself has no preference for any of the three variables.

Once a matching is chosen, this symmetry is broken, since each matched constraint is now associated with one matched variable and some non-matched ones. For a given constraint, matched and non-matched variables are identified in the graph incidence matrix by ①or 1, respectively. For example, according to the matching in Fig. 5.9a, the above constraint is used to compute $q_i(t)$.

In the graphical representation, the unsymmetries associated with a matching are represented by transforming the originally non-oriented edges into oriented ones. Since some constraints might not be matched, the following rules are applied:

- **Matched constraints:** The edges adjacent to a matched constraint are provided with an orientation
  - from the non-matched (input) variables, to the constraint,
  - from the constraint to the matched (output) variable.

- **Non-matched constraints:** All the variables are considered as input and, hence, all edges are oriented from the variables to the constraint.

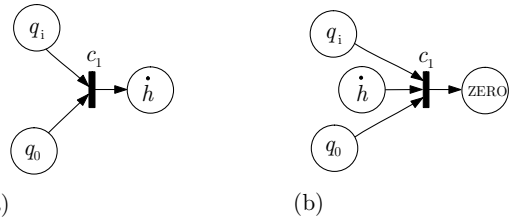


Fig. 5.10. (a) A matched and (b) a non-matched constraint (5.9)

To understand the reason for these rules, consider a matching $\mathcal{M}$ and choose an edge $(c, x) \in \mathcal{M}$. Then the variable x can be considered as the output of the constraint c while the other variables $Q(c)\backslash\{x\}$ are the input[1]. The interpretation is that the matching represents some causality assignment by which the constraint c is used to compute the variable x assuming the other variables to be known. An explicit representation of the constraint c that can be used to determine x is denoted by

$$x = \gamma\left(Q\left(c\right)\backslash\{x\}\right).$$

For non-matched constraints all variables are considered as input and no variable of $Q(c)$ can be considered as an output. Hence, the constraint can be written in the form

$$c\left(Q(c)\right) = 0$$

like Eq. (5.9). If the zero on the right-hand side is considered as output, the constraint can be associated with a ZERO vertex like in Fig. 5.10b. Using no label at all is considered as an implicit ZERO label.

Example 5.9 *Computation of unknown variables of the tank*

For the single-tank system, the reduced graph shown in Fig. 5.7 and the three matchings shown in Fig. 5.9 yield the oriented graphs shown in Fig. 5.11. The directed edges show how the internal variables q_i, q_0, h and $\dot{h}$ can be determined for known values of u and y.

As Matching 1 is incomplete, the unknown variable q_o cannot be computed as shown in the graph. Matchings 2 and 3 are complete in $\mathcal{X}$ but incomplete in $\mathcal{C}_{\mathcal{X}}$. The non-matched constraint c_1 or c_4, respectively, leads to a ZERO output, that is, they have to hold for the variables q_i and $\dot{h}$ or h and y, which have been determined by other constraints or have been measured, respectively. □

Note that subgraphs whose input and output nodes are all known or ZERO variables provide the system input-output relations. By using Matching 2 in Fig. 5.11 the two following input-output relations are found. The first one is provided by constraint c_5 which links only known variables and are, therefore, deleted when drawing the reduced graph, and the second one is provided by the non-matched constraint c_1

[1] In Eq. (5.7), Q has been defined as a mapping from a set of constraints towards the sets of variables. It is used here also for a single constraint c where for notational convenience $Q(\{c\})$ is written as $Q(c)$.

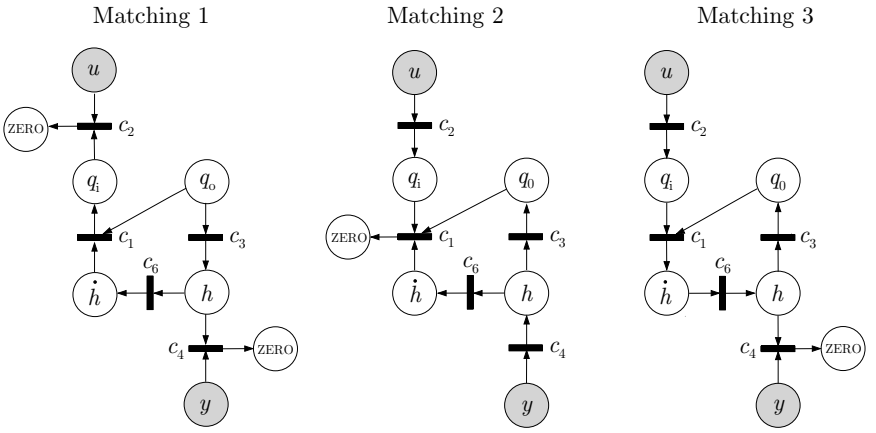


Fig. 5.11. Graphs corresponding to the three matchings

$$c_5\,(u,\,y) = 0$$
$$c_1\,(u,\,\gamma_1\,(\gamma_3\,(k,\,\gamma_4(y))\,,\,\gamma_6\,(\gamma_4(y)))) = 0,$$

where $\gamma_i(z)$ denotes the output of constraint c_i for the input z.

5.3.3 Alternated chains and reachability

The oriented graph associated with a matching has the following property: Any existing path between two nodes (variables or constraints) alternates successively variables and constraints nodes. Such a path is called an *alternated chain*. Its length is the number of constraints that are crossed along the path. Note that if a non-matched constraint belongs to an alternated chain, the chain ends with the ZERO variable associated with the non-matched constraint.

Associated with alternated chains is the notion of reachability.

Definition 5.6 (Reachability)
A variable z_2 is reachable from a variable z_1 if there exists an alternated chain from z_1 to z_2. A variable z_2 is reachable from a subset $\chi \subseteq \mathcal{Z} \backslash \{z_2\}$ if there exists some variable $z_1 \in \chi$ such that z_2 is reachable from z_1. A subset of variables $\mathcal{Z}_2$ is reachable from a subset of variables $\mathcal{Z}_1$ if any variable of $\mathcal{Z}_2$ is reachable from $\mathcal{Z}_1$.

Example 5.10 *Alternated chains in the tank example*

Some alternated chains associated with the oriented graph of the tank example are as follows:

$$y - c_4 - h - c_3 - q_0 - c_1 - q_i$$
$$h - c_6 - \dot{h} - c_1 - q_i.$$

It can be checked that any variable of the set $\left\{q_i,\, q_o,\, h,\, \dot{h}\right\}$ is reachable from y. □

5.3.4 Causal interpretation

The aim of this subsection is to discuss the causal interpretation of the oriented bi-partite graph associated with a matching.

Selecting a pair (c, z) to belong to a matching implies a causality assignment, by which the constraint c is used to compute the variable z, assuming the other variables to be known. The oriented bi-partite graph which results from a causality assignment is named a *causal graph*. Causal graphs are used in qualitative reasoning, alarm filtering or in providing the computation chain needed for the numerical or formal determination of some variables of interest, as shown by the above interpretation. Although this interpretation is straightforward for simple algebraic constraints, it has to be considered more carefully when loops and differential constraints are present.

Algebraic constraints. Let $c \in \mathcal{C}$ be an algebraic constraint, $Q(c)$ the set of the variables constrained by c and let $n_c = |Q(c)|$. In structural analysis, the following assumption is made:

Assumption 5.2 *Any algebraic constraint $c \in \mathcal{C}$ defines a manifold of dimension $n_c - 1$ in the space of the variables $Q(c)$.*

Since the constraint has to be satisfied at any time t, the variables $Q(c)$ cannot behave independently of each other. Assumption 5.2 means that only $n_c - 1$ unknowns can be chosen arbitrarily (or imposed) in constraint c. Hence, there is at least one variable $z \in Q(c)$ such that $\frac{\partial c}{\partial z} \neq 0$ almost everywhere in the space of the variables $Q(c)$. Therefore, from the implicit function theorem, its trajectory can be deduced (at least locally) from the constraint c and the trajectories of the $n_c - 1$ other variables. This is exactly the causal interpretation of matching this variable with constraint c, and it can be interpreted as: constraint c decreases by one the degrees of freedom associated with the variables $Q(c)$.

Example 5.11 *Algebraic constraints*

Consider the constraint

$$c_1 : a_1 x_1 + b_1 x_2 - y_1 = 0, \tag{5.10}$$

where x_1 and x_2 are two unknowns, a_1 and b_1 are parameters, and y_1 is known. This constraint obviously defines a one-dimensional space to which any vector $(x_1, x_2)'$ should belong when it is satisfied. Thus only one degree of freedom is left since only one of the unknowns can be chosen arbitrarily, the possible value(s) of the other one being deduced from (5.10).

Note that the structural point of view considers the most general case of any pair of parameters a_1 and b_1. Particular cases result if a_1 or b_1 equal to zero ((5.10) would still define a one dimensional manifold) or if a_1 and b_1 both equal to zero. In the latter case c_1 would not define a one dimensional manifold when $y_1 = 0$, since any point $(x_1, x_2)'$ in the two dimensional space would satisfy the constraint, and there would be no solution when $y_1 \neq 0$ i.e. the system model would not be sound since constraint (5.10) would allow for no solution. □

The fact that *at least one* variable can be matched in a given constraint under the causal interpretation does not mean that *any* variable has this property. An obvious situation in which $(c,\, x)$ cannot be matched is when c is not invertible with respect to x. The constraint shown in Fig. 5.12 defines a manifold of dimension 1 in $\mathbb{R}^2$, and it is always possible to compute x_2 once x_1 is given. Matching x_2 with this constraint can obviously be interpreted as explained above. However, the interpretation does not apply to the matching of x_1, since $\frac{\partial c}{\partial x_1}$ is not different from zero almost everywhere, thus the constraint c cannot be used to compute x_1 whatever the value of x_2.

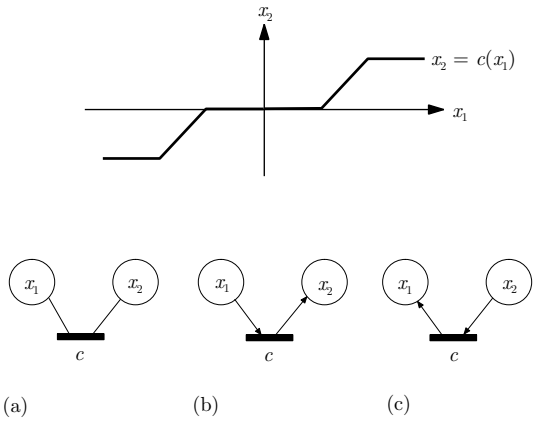


Fig. 5.12. (a) Structure graph, (b) possible and (c) impossible matching

Differential constraints. The case of differential constraints has to be considered more carefully. Remember that differential constraints can always be represented as

$$d : x_2(t) - \frac{d}{dt}x_1(t) = 0 \tag{5.11}$$

and indeed functions $x_1(t)$ and $x_2(t)$ cannot be chosen independently of each other. Obviously, when the trajectory $x_1(t)$ is known, its derivative can always be computed (from an analytical point of view, derivatives are here supposed to exist, and from a numerical point of view, there might be problems rised by the presence of noise, which are not considered here). It follows that the constraint can always be matched for x_2 which is then uniquely defined. This is called *derivative causality*. When $x_2(t)$ is known, matching this constraint for x_1, (which is called *integral causality*), leads to the computation

$$x_1(t) = x_1(0) + \int_0^{'} x_2(\sigma)d\sigma, \tag{5.12}$$

which does not determine $x_1(t)$ uniquely, unless the initial condition $x_1(0)$ is known. Let $(x_1(t),\, x_2(t))'$ be two functions which satisfy constraint d. Then, $(x_1(t) + \alpha,\, x_2(t))'$, where α is any constant function, also satisfies constraint d. Thus, computing x_1 from constraint d may be possible or impossible, depending on the context. Initial values are known in a simulation context, since they are under the control of the user, but this is generally not true in a fault diagnosis context, thus forbidding integral causality in differential constraints.

Remark 5.2 *Consequences for residual generation*

Parity space or identification-based residual generation approaches aim at eliminating the unknown initial values by using the system input-output relations which are obtained through derivative causality. The observer-based approaches use integral causality by implementing an auxiliary system – the observer – which provides results that are (asymptotically) independent of the estimate of the initial state. □

In summary, different cases have to be considered as far as the counterpart of Assumption 5.2 (a differential constraint defines a manifold of dimension $n_c - 1$ in the space of the variables $Q(c)$ is concerned:

- If $x_1(t)$ is known, $x_2(t)$ can be matched in constraint d which brings about using differential causality. This provides a unique result for x_2. Assumption 5.2 is satisfied since constraint d leaves only one degree of freedom in the determination of $(x_1(t),\, x_2(t))$.

- If $x_2(t)$ and the initial value $x_1(0)$ are known, $x_1(t)$ can be matched in constraint d using integral causality. This provides a unique result obtained from Eq. (5.12). Assumption 5.2 is satisfied since constraint d leaves only one degree of freedom in the determination of $(x_1(t),\, x_2(t))$.

- If only $x_2(t)$ is known, Assumption 5.2 is not satisfied, because whatever the matching, two degrees of freedom (the constant function α, and the input function $x_2(t)$) still exist in the determination of $(x_1(t),\, x_2(t))$.

Example 5.12 *Differential system*

A model whose solution exists but is not unique, as the result of Assumption 5.2 being not satisfied, is given by the following simple example

$$\begin{aligned} c_1 \quad &: \quad x_2 - ax_1 - bu = 0 \\ c_2 \quad &: \quad x_2 - \frac{d}{dt}x_1 = 0, \end{aligned}$$

which is a single-input first-order system. Constraint c_1 is an algebraic one which expresses the fact that the vector $(x_1,\, x_2)'$ lives in a linear manifold of dimension one for known input u. Constraint c_2 does not allow to decrease the dimension of the unknown vector. If x_1 were known (which is not the case), one could compute its derivative x_2, but the knowledge of x_2 (which could be obtained as a function of x_1 and u in constraint c_1) is of no help to compute x_1 since one should proceed by integration and the initial value $x_1(0)$ is unknown. □

Example 5.13 *Derivative causality in the tank system*

Consider the following matching in the tank structural model.

↗	h	$\dot{h}$	q_i	q_o	u	y
c_1		①	1	1		
c_2			①		1	
c_3	1			①		
c_4	1					1
c_5					1	1
c_6	①	1				

Although it is complete with respect to the variables $\{h, \dot{h}, q_i, q_o\}$, it cannot be used for their computation since it introduces an integral causality (h should be computed from $\dot{h}$ by constraint c_6, while its initial value is not known because constraint c_4 is not matched).

Derivative causality can be forced, when necessary. To represent this situation, the symbol $\mathbf{x}$ is used, which forbids integral matchings. The previous matching will not be obtained if the tank structural model is written as

↗	h	$\dot{h}$	q_i	q_o	u	y
c_1		1	1	1		
c_2			1		1	
c_3	1			1		
c_4	1					1
c_5					1	1
c_6	$\mathbf{x}$	1				

where $\mathbf{x}$ means that although there is an edge between c_6 and h, h cannot be matched with c_6. □

Loops. In the oriented graph associated with a matching, loops are special subsets of constraints, which have to be solved simultaneously, because the output signals of some constraints in the loop are the inputs of some others in the same loop. Since only one variable is matched with one constraint, the number of matched variables in a loop is equal to the length of the loop, i.e. the number of the constraints which appear in it.

The causal interpretation of a loop is that of a subset of constraints whose solution is the set of the matched variables, when all the other variables (not matched in the loop) are known.

Suppose n_v variables are constrained by a subsystem of n_l constraints, and there is a matching such that they form a loop. Then, n_l variables are internal (matched in the loop), and $n_v - n_l$ variables are external (not matched in the loop).

In structural analysis, an algebraic loop is always supposed to have a unique solution (more precisely: a finite number of solutions), which is the intersection of n_l manifolds of dimension $n_l - 1$, if the external variables are known (by Assumption 5.2). The loop is associated with a subset of n_l constraints

$$h_l(x_l,\, x_e) = 0,$$

where x_l (respectively x_e) are the internal (respectively the external) variables, and each component of x_l is matched with one constraint in h_l.

It is worth noticing that the interpretation associated with causality in single constraints is not directly extendable to loops, as shown by the following example.

Example 5.14 *A system of two non-invertible constraints*

Consider the non-invertible constraint from Example 5.11 and suppose now there are two constraints $\{c_1,\, c_2\}$ of the same form, but with different parameters. The structural graph of this system is

↗	x_1	x_2
c_1	1	1
c_2	1	1

A complete matching is given by

↗	x_1	x_2
c_1	①	1
c_2	1	①

which shows the existence of a loop, and indeed the system of two constraints with two unknowns has a (finite number of) solution(s), as illustrated by Fig. 5.13, although the matching of x_1 with c_1 has certainly not the nice interpretation of computing the variable from the constraint.

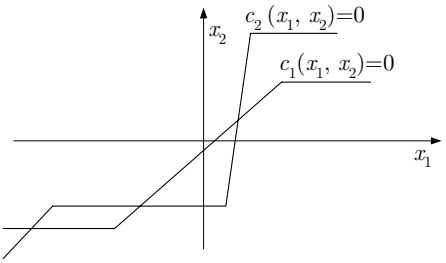


Fig. 5.13. Two algebraic constraints with two unknowns

The correct interpretation, in this case, comes from the fact that each constraint defines a (different) manifold of dimension one in $\mathbb{R}^2$, and that, in general, such two manifolds will

intersect in a finite number of points. No solution at all would be a particular case (which would not satisfy Assumption 5.1), and an infinite number of solutions would be the result of the two manifolds being the same one (at least locally). □

The uniqueness of the solution associated with a loop which contains differential constraints depends upon the context of the problem. For a set of $n_l + n_e$ variables which are constrained by n_l differential equations consider matchings such that this system forms a loop

$$\begin{aligned} \boldsymbol{z}_l &= \boldsymbol{g}_l(\boldsymbol{x}_l, \boldsymbol{x}_e, \boldsymbol{u}) \\ \boldsymbol{z}_l &= \frac{d}{dt}\boldsymbol{x}_l, \end{aligned} \tag{5.13}$$

where $\boldsymbol{x}_l$ is the vector of the variables in the loop, $\boldsymbol{g}_l$ are the constraints in the loop, and $\boldsymbol{x}_e$ are the external variables, supposed to be known. The system (5.13) has a unique solution only if the initial value $\boldsymbol{x}_l(0)$ is known. When this is not the case, the solution will depend on the n_l unknowns $\boldsymbol{x}_l(0)$, and thus it will belong to a manifold of dimension n_l. Such a differential loop is called *non-causal*.

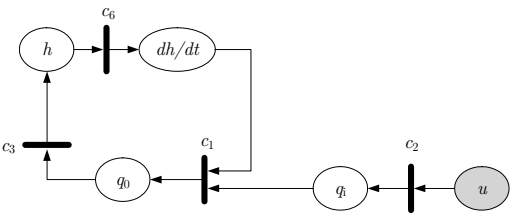


Fig. 5.14. A matching with a differential loop

Example 5.15 *Differential loop in the tank example*

Consider the following matching

↗	h	$\dot{h}$	q_i	q_o	u	y
c_1		1	1	①		
c_2			①		1	
c_3	①			1		
c_4	1					1
c_5					1	1
c_6	1	①				

which is complete with respect to $\{h, \dot{h}, q_i, q_o\}$, and in which differential causality is now used in constraint c_6. This is obviously not enough for being able to compute h, since there is a differential loop: $h - c_6 - \dot{h} - c_1 - q_0 - c_3 - h$, which is shown of Fig. 5.14. □

Following a classical graph theory approach, a loop can be condensed into one single node, which represents a subsystem of constraints to be solved simultaneously. Another approach is to avoid loops (whenever possible) by some transformation of the constraints, leading to diagonal or triangular system structures.

Example 5.16 *Treatment of loops*

Let a subsystem be defined by $\mathcal{Z} = \{x_1,\, x_2,\, y_1,\, y_2\}$, $\mathcal{C} = \{c_1,\, c_2\}$. The variables are real numbers, the constraints are linear, $y_1,\, y_2$ are supposed to be known, and the problem to be solved concerns the computation of $x_1,\, x_2$.

$$\begin{aligned} c_1 \quad &: \quad a\, y_1 + b\, x_1 + c\, x_2 = 0 \\ c_2 \quad &: \quad \alpha\, y_2 + \beta\, x_1 + \gamma\, x_2 = 0. \end{aligned} \tag{5.14}$$

The incidence matrix of the structure graph, and a complete matching w.r.t. $\{x_1,\, x_2\}$ are

↗	x_1	x_2	y_1	y_2
c_1	①	1	1	
c_2	1	①		1

and Fig. 5.15 shows the resulting loop in the associated oriented graph. Note that this matching is valid almost for every value of the coefficients since it obviously supposes that b and γ are non-zero, note also that the solvability condition $b\,\gamma - c\,\beta \neq 0$ cannot be seen from structural considerations.

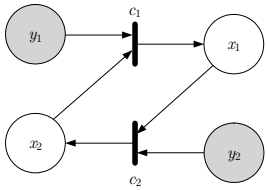


Fig. 5.15. An algebraic loop

Figure 5.16 illustrates the condensation in which the loop is "condensed" into one single node, which means that the system of two equations with two unknowns is solved, but no detail is given about how this is done.

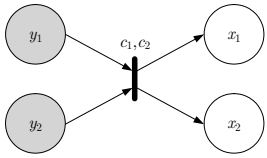


Fig. 5.16. Condensed representation of the loop

Transforming the constraints may also lead to a loop-free oriented graph, since it may give the system a diagonal or a triangular structure (note that this results from manipulations which are not purely structural, but which are done on the system behaviour model). For example, the two following systems are equivalent to (5.14):

$$\begin{array}{lcl} c'_1 & : & a\,\gamma\,y_1 - \alpha\,c\,y_2 + (b\,\gamma - \beta\,c)\,x_1 = 0 \\ c'_2 & : & a\,\beta\,y_1 - \alpha\,b\,y_2 + (c\,\beta - b\,\gamma)\,x_2 = 0 \end{array} \tag{5.15}$$

$$\begin{array}{lcl} c'_1 & : & a\,\gamma\,y_1 - \alpha\,c\,y_2 + (b\,\gamma - \beta\,c)\,x_1 = 0 \\ c''_2 & : & a y_1 + b x_1 + c x_2 = 0. \end{array} \tag{5.16}$$

Figure 5.17 illustrates the loop-free oriented graphs associated with systems (5.15) and (5.16). □

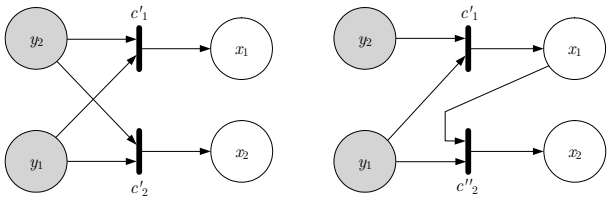


Fig. 5.17. Two equivalent loop-free oriented graphs

5.3.5 Matching algorithms

From the definition, a matching can be represented in the incidence matrix of the bi-partite graph by selecting one "1" at most in each row and in each column. Each selected "1" represents an edge of the matching, and no other edge of the matching should contain the same variable (thus it is the only one in the row) or the same constraint (thus it is the only one in the column). Matchings and maximal matchings can be defined for any graph (not only bi-partite graphs). They have been extensively addressed in graph theory, because they provide the solution to many application problems. We shall first present the classical maximal matching algorithm and then show that the maximal flow algorithm solves this problem in an elegant way for bi-partite graphs. Finally, taking into account the causal interpretation of matchings, we present a ranking algorithm which is well suited when no cycles are present in the graph.

Basic matching algorithm. Let M be a matching on a graph G. An edge is said to be *weak* with respect to M if it does not belong to M. A vertex is *weak* with respect to M if it is only incident to weak edges. An *M-alternating path* is a path

whose edges are alternately in M and not in M (or conversely). An *M-augmenting path* is an alternating path whose end vertices are both weak with respect to M. An *M-alternating tree* with root v is a collection of disjoint *M-alternating paths* except for the root v which is a common vertice.

The basic matching algorithm is built on the following theorem:

Theorem 5.1 *A matching M in a graph G is maximum if and only if there exists no M-augmenting path in G.*

The proof is not given here but roughly, the idea is that if such an augmenting path would exist, a new matching of size $|M| + 1$ would be obtained by *transferring* M i.e. by exchanging the roles of the matched and non-matched edges in the path.

Example 5.17 *Transfer on an M-augmenting path*

A matching M is given by the bold edges in the bi-partite graph of Fig. 5.18.

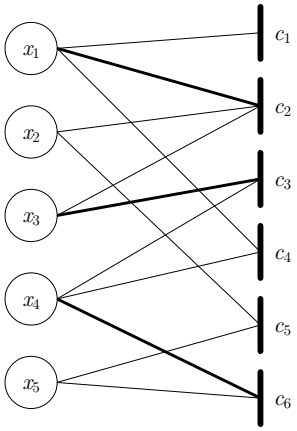


Fig. 5.18. A bi-partite graph

It can be checked that there exists an *M-augmenting path*, namely

$$\underbrace{c_1 - x_1}_{weak} - \underbrace{x_1 - c_2}_{matched} - \underbrace{c_2 - x_3}_{weak} - \underbrace{x_3 - c_3}_{matched} - \underbrace{c_3 - x_4}_{weak} - \underbrace{x_4 - c_6}_{matched} - \underbrace{c_6 - x_5}_{weak}$$

and therefore this matching is not maximal. By transferring M, a matching of size 4 is found

$$\underbrace{c_1 - x_1}_{matched} - \underbrace{x_1 - c_2}_{weak} - \underbrace{c_2 - x_3}_{matched} - \underbrace{x_3 - c_3}_{weak} - \underbrace{c_3 - x_4}_{matched} - \underbrace{x_4 - c_6}_{weak} - \underbrace{c_6 - x_5}_{matched}$$ □

Based on the theorem above, the following algorithm consists in finding an augmenting path (if there is one) and augmenting the size of the current matching by transferring it, until no augmenting path can be found (therefore the current matching is maximum).

Algorithm 5.1 *Algorithm for finding a maximum matching*

Given: The system bi-partite graph and an initial matching M_0 (this can be any matching including the empty matching $M_0 = \{\}$).

1. Let M be the current matching. If the number of weak vertices with respect to M is less than or equal to one, the current matching is maximum. Otherwise, let v be any weak vertice. Build an alternating tree with root v.

2. If the tree contains an *M-augmenting path* then perform a transfer along this path and update the matching on the initial graph. The cardinal of the new matching is increased by one. Go back to 1.

Result: A maximal matching.

Example 5.18 *Maximum matching algorithm*

Let $M_0 = \{(x_1, c_2), (x_3, c_3), (x_4, c_6)\}$ be the initial matching shown on Fig. 5.18. The weak edges are

$$\{(x_1, c_1), (x_1, c_4), (x_2, c_2), (x_2, c_5), (x_3, c_2), (x_4, c_3), (x_4, c_4), (x_5, c_5), (x_5, c_6)\}$$

There are four weak vertices, namely $\{c_1, x_2, c_4, c_5\}$. For the first iteration, chosing e.g. c_1 as the root gives the alternating tree shown on Fig. 5.19 where the current matching is shown in dashed lines. It is easily seen that there are two M_0-augmenting paths, namely $c_1 - x_1 - c_2 - x_2$ and $c_4 - x_4 - c_6 - x_5$. Since they are disjoint, the two transfers can be done simultaneously, resulting in the matching

$$M_1 = \{(x_1, c_1), (x_2, c_2), (x_4, c_4), (x_5, c_6)\}$$

after the first iteration (performing the transfer on a maximal set of shortest disjoint augmenting path according to set inclusion is an improvement of the basic approach). The weak edges are now

$$\{(x_1, c_2), (x_1, c_4), (x_2, c_5), (x_3, c_2), (x_3, c_3), (x_4, c_3), (x_4, c_6), (x_5, c_5)\}$$

and the weak vertices are $\{x_3, c_3, c_5\}$. Chosing c_3 as the root results in the alternated tree of Fig. 5.20, which exhibits the M_1-augmenting path $x_3 - c_2 - x_2 - c_5$. Performing the transfer gives the new matching

$$M_2 = \{(x_1, c_1), (x_2, c_5), (x_3, c_2), (x_4, c_4), (x_5, c_6)\}$$

which is maximal, since the set of weak edges is now

$$\{(x_1, c_2), (x_1, c_4), (x_2, c_2), (x_3, c_3), (x_4, c_3), (x_4, c_6), (x_5, c_5)\}$$

and there remains only one single weak vertice c_3. □

Example 5.19 *Application to the single-tank system*

Let us search for a maximal (complete if possible) matching with respect to the subgraph without the known variables

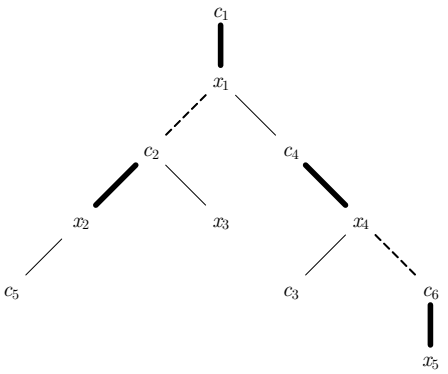


Fig. 5.19. Alternating tree with root c_1

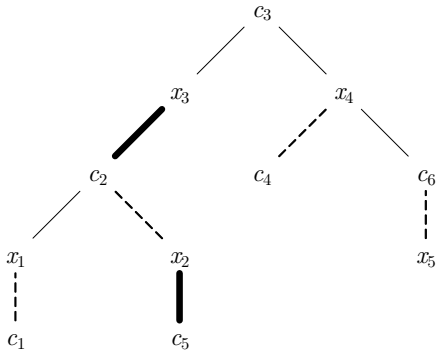


Fig. 5.20. Alternating tree with root c_3

$\nearrow$	h	$\dot{h}$	q_i	q_o	u	y
c_1		1	1	1		
c_2			1		1	
c_3	1			1		
c_4	1					1
c_5					1	1
c_6	1	1				

Let $M_0 = \{\}$. Then all vertices are weak. Selecting e.g. h as the root gives the alternated tree

$$\begin{array}{cccccccccc} & \nearrow & c_6 & \rightarrow & \dot{h} & & & & & \\ h & \longrightarrow & c_4 & & & & & & & \\ & \searrow & c_3 & \rightarrow & q_o & \rightarrow & c_1 & \rightarrow & q_i & \rightarrow & c_2 \end{array}$$

where two disjoint M_0-augmenting paths are respectively $c_6 - \dot{h}$ and $h - c_3 - q_o - c_1 - q_i - c_2$ providing the new matching $M_1 = \left\{ \left(\dot{h}, c_6\right), (h, c_3), (q_o, c_1), (q_i, c_2) \right\}$ which is complete with respect to the unknown variables, hence the algorithm ends. □

Maximal flow algorithm. Finding a maximal matching in a bi-partite graph can also be transformed into a maximal flow problem. The procedure is as follows: construct a network $\mathcal{N}$ associated with the graph $(\mathcal{C}, \mathcal{Z}, \mathcal{E})$ by orienting all edges from $\mathcal{Z}$ to $\mathcal{C}$, inserting a source vertex S with arcs to all vertices of $\mathcal{Z}$ and a sink vertex T with arcs from all vertices of $\mathcal{C}$, and connecting T to S as shown on Fig. 5.21.

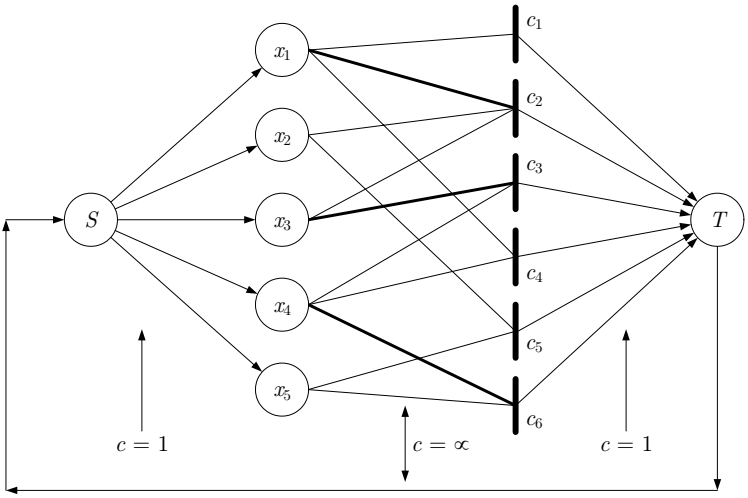


Fig. 5.21. Setting the maximal matching problem as a maximal flow problem

Furthermore, assign the capacity of all arcs from S to $\mathcal{Z}$ or from $\mathcal{C}$ to T as 1. The capacities of all other arcs are set to ∞. Then, the maximal flow on $\mathcal{N}$ is associated with a maximum matching , from the following result.

Theorem 5.2 *In a bi-partite graph , the number of edges in a maximum matching equals the maximum flow in the network $\mathcal{N}$.*

Therefore, a maximal matching can be found by applying the classical Ford and Fulkerson maximum flow algorithm, which in the case of bi-partite graphs is called "the Hungarian method". This algorithm assumes a given matching M is known and attempts to extend M by finding an augmenting path. This is done by marking vertices on weak edges so as to follow possible augmenting paths.

Algorithm 5.2 *Hungarian method*

Given: The system bi-partite graph and an initial matching M_0 (this can be any matching including the empty matching $M_0 = \{\}$)

1. Let M be the current matching. Label with an * all vertices of $\mathcal{Z}$ that are weak with respect to M, and alternately apply steps 2 and 3 until no further labeling is possible.

2. Select a newly labeled vertex in $\mathcal{Z}$, say z_i, and label with z_i all unlabelled vertices of $\mathcal{C}$ that are connected to z_i by a weak edge with respect to M. Repeat this step on all vertices of $\mathcal{Z}$ that were labeled in the previous step.

3. Select a newly labeled vertex of $\mathcal{C}$, say c_j and label with c_j the vertice of $\mathcal{Z}$ which is connected to c_j in M. Repeat this process on all vertices of $\mathcal{C}$ labelled in the previous step.

4. The labelings will continue to alternate until one of two possibilities occurs:

END 1: A weak vertex $\mathcal{C}$ has been labeled. If END 1 occurs, an M-augmenting path has been found, and it can be constructed by working backwards through the labels until the vertex of $\mathcal{Z}$ which is labeled *. Transferring this path gives an extended matching and the algorithm is repeated by going back to step 2.

END 2: It is not possible to label any more vertices and END 1 has not occured. If END 2 occurs, M is already a maximal matching.

Result: A maximal matching.

Example 5.20 *Hungarian method*

Let $M_0 = \{(x_1, c_2), (x_3, c_3), (x_4, c_6)\}$ be the initial matching shown on Fig. 5.18. The table below shows the initial labelling and the sequence of labels obtained as steps 2 and 3 alternate. The bar over vertices c_1, c_4 and c_5 is to remind that they are weak with respect to the current matching.

	x_1	x_2	x_3	x_4	x_5	$\bar{c}_1$	c_2	c_3	$\bar{c}_4$	$\bar{c}_5$	c_6
Step 1		*			*						
Step 2-1							x_2			x_2	x_5
Step 3-1	c_2			c_6							
Step 2-2						x_1		x_4	x_1		
Step 3-2			c_3								

In this example, the first iteration would stop at step 2-1 since the weak vertice c_5 has been labelled (END 1). Tracking the labels backwards until a * vertice is found gives the $M-$augmenting path $c_5 - x_2$, which results in the augmented matching

$$M_1 = \{(x_1, c_2), (x_2, c_5), (x_3, c_3), (x_4, c_6)\}$$

from which the next iteration starts. □

Example 5.21 *Hungarian method in the single-tank system*

Let $M_0 = \left\{\left(\dot{h}, c_6\right)\right\}$ be the initial matching. The first iteration gives the labels

	h	$\dot{h}$	q_i	q_o	$\bar{c}_1$	$\bar{c}_2$	$\bar{c}_3$	$\bar{c}_4$	$\bar{c}_5$	c_6
Step 1	*		*	*						
Step 2					q_i	q_i	h	h		h

On the first labelling step, END 1 occurs and the matching can be updated as

$$M_1 = \left\{\left(\dot{h}, c_6\right), (h, c_3), (q_i, c_1)\right\}$$

The second iteration gives

	h	$\dot{h}$	q_i	q_o	c_1	$\bar{c}_2$	c_3	$\bar{c}_4$	$\bar{c}_5$	c_6
Step 1				*						
Step 2-1					q_0		q_0			
Step 3-1	c_3		c_1							
Step 2-2						q_i		h		h

In step 2-2, END 1 occurs, showing the M_1-augmenting path $c_4 - h - c_3 - q_0$ and the updated matching is therefore

$$M_2 = \left\{\left(\dot{h}, c_6\right), (h, c_4), (q_i, c_1), (q_o, c_3)\right\}$$

which is complete with respect to the unknown variables and stops the algorithm (note that the solution is different from the one previously found, this illustrates the fact that maximal matchings are not unique). □

A ranking algorithm. From the causal interpretation, a complete causal matching over the unknown variables identifies the computations to be done in order to express each of them as a function of the known variables. If the matching is not complete with respect to the constraints, non-matched constraints exist, and replacing the unknown variables by their values in these constraints results in redundancy relations that must be satisfied if actual constraints are the same as the model constraints (Example 5.8 shows such a situation for the tank system). This remark is the basis of the following constraint propagation (or ranking) algorithm, which can be used to find a matching. It does only generate oriented graphs without loops, so it may not find any complete matching, even if it exists. The idea is to start with some variable (in applications the starting nodes are the known variables), and to "propagate"the knowledge, step by step, by matching, at each step, the variables which intervene in constraints where all other involved variables are matched or known.

Algorithm 5.3 *Ranking of the constraints*

Given: Incidence matrix or structure graph

1. Mark all known variables.
   $i = 0$
2. Find all constraints in the current table with exactly one unmarked variable. Associate rank i with these constraints and mark these constraints as well as the corresponding variable.
3. Set $i := i + 1$.
4. If there are unmarked constraints whose variables are all marked, associate them with rank i, mark them and connect them with the pseudo-variable ZERO.
5. If there are unmarked variables or constraints, continue with Step 2.

Result: Ranking of the constraints.

In the first step, all known variables $\mathcal{K}$ are marked and all unknown variables remain unmarked. Then every constraint that contains at most one unmarked variable is assigned rank 0. The constraint is matched for the unmarked variables (or zero, if there is none), and the variable is marked. This step is repeated with an increasing rank number, until no new variables can be matched.

As every matched variable is also given a number, this approach is called ranking algorithm. The rank can be interpreted as the number of steps needed to calculate an unknown variable from the known ones.

Example 5.22 *Constraint propagation in the single-tank system*

The constraint propagation algorithms applied to the tank example as follows. As u, y are known, only the variable set $\left\{q_i,\, q_o,\, h,\, \dot{h}\right\}$ has to be matched.

Starting set (rank 0): $\{u,\, y\}$
First step (rank 1): match q_i with c_2, match h with c_4
Second step (rank 2): match q_0 with c_3, match $\dot{h}$ with c_6
End (every variable is matched)

The obtained matching is the following one.

$\nearrow$	h	$\dot{h}$	q_i	q_o	u	y
c_1		1	1	1		
c_2			①		1	
c_3	1			①		
c_4	①					1
c_5					1	1
c_6	1	①				

Hence, the ranking algorithm can be used to get a complete matching for the tank system. □

5.4 System canonical decomposition

5.4.1 Canonical subsystems

This section recalls a classical result from bi-partite graph theory, which states that any finite-dimensional graph can be decomposed into three subgraphs with specific properties, respectively associated with an over-constrained, a just-constrained and an under-constrained subsystem. This decomposition is canonical, i.e. for a given system, it is unique. The three subsystems play a major role in the analysis of the system structural properties: observability, controllability, monitorability, reconfigurability.

Definition 5.7 (Over-constrained, just-constrained, under-constrained graph)
A graph $(\mathcal{C}, \mathcal{Z}, \mathcal{E})$ *is called*

- *over-constrained if there is a complete matching on the variables* $\mathcal{Z}$ *but not on the constraints* $\mathcal{C}$*,*
- *just-constrained if there is a complete matching on the variables* $\mathcal{Z}$ *and on the constraints* $\mathcal{C}$*,*
- *under-constrained if there is a complete matching on the constraints* $\mathcal{C}$ *but not on the variables* $\mathcal{Z}$*.*

In an over-constrained graph, there remains a complete matching on $\mathcal{Z}$ after any single constraint has been removed from the set $\mathcal{C}$.

Example 5.23 *Property of the reduced graph of the tank system.*

Matching 2 of the tank graph is complete with respect to the variables, but there is still one non-matched constraint: the reduced graph of the tank system is over-constrained.

$\nearrow$	h	$\dot{h}$	q_i	q_o	u	y
c_1		1	①	1		
c_2			1		1	
c_3	1			①		
c_4	①					1
c_6	1	①				

It can be furthermore noticed that any of the 5 constraints can be removed, and there still is a complete matching on the resulting graph. □

The graph of a given system $\mathcal{S} = (\mathcal{C}, \mathcal{Z})$ may of course fail to conform to any of the three properties defined above. In this case, it can be proved that there exists a unique decomposition of $\mathcal{S}$ into three subsystems

$$\begin{aligned} \mathcal{S}^+ &= (\mathcal{C}^+, \mathcal{Z}^+) \\ \mathcal{S}^0 &= (\mathcal{C}^0, \mathcal{Z}^+ \cup \mathcal{Z}^0) \\ \mathcal{S}^- &= (\mathcal{C}^-, \mathcal{Z}^+ \cup \mathcal{Z}^0 \cup \mathcal{Z}^-) \end{aligned}$$

such that

(a) $(\mathcal{C}^-, \mathcal{C}^0, \mathcal{C}^+)$ form a partition of $\mathcal{C}$,
(b) $(\mathcal{Z}^-, \mathcal{Z}^0, \mathcal{Z}^+)$ form a partition of $\mathcal{Z}$,
(c) $(\mathcal{C}^+, \mathcal{Z}^+)$ is over-constrained,
(d) $(\mathcal{C}^0, \mathcal{Z}^0)$ is just-constrained,
(e) $(\mathcal{C}^-, \mathcal{Z}^-)$ is under-constrained.

- $\mathcal{S}^+$ is called the *over-constrained subsystem* because it follows from the definition that $|\mathcal{C}^+| > |\mathcal{Z}^+|$, which means that the variables $\mathcal{Z}^+$ (let n^+ be their number) have to satisfy more than n^+ constraints.

- $\mathcal{S}^0$ is called the *just-constrained subsystem* because it introduces as many new variables as constraints ($\left|\mathcal{C}^0\right| = \left|\mathcal{Z}^0\right|$).

- Similarly, $\mathcal{S}^-$ is called the *under-constrained subsystem*. It is characterised by $|\mathcal{Z}^-| > |\mathcal{C}^-|$ (it introduces more new variables than constraints).

Figure 5.22 illustrates the canonical decomposition of the structure graph, showing the partition of $\mathcal{C}$ into $\left\{\mathcal{C}^+, \mathcal{C}^0, \mathcal{C}^-\right\}$ and the partition of $\mathcal{Z}$ into $\left\{\mathcal{Z}^+, \mathcal{Z}^0, \mathcal{Z}^-\right\}$ which define the three canonical components on its incidence matrix. White areas are zeros, grey areas contain zeros and ones, and the thick line represents a matching. The figure applies after the rows and columns have been rearranged so that the matched variables and constraints appear on the diagonal.

It can be noticed that the over-constrained subsystem may contain several smaller over-constrained or just-constrained ones (at least one smaller over-constrained subsystem has to exist, otherwise the overall graph would be just-constrained, and of course no under-constrained subsystem can be found in an over-constrained

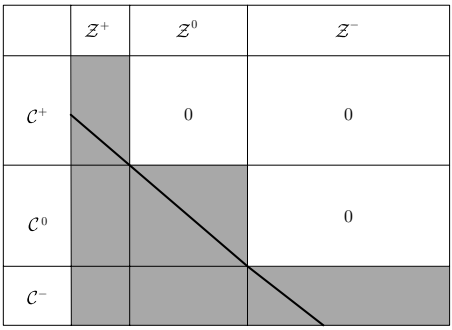


Fig. 5.22. Canonical decomposition of the structure graph

one). Similarly, the just-constrained subsystem can contain several smaller just-constrained ones (no over-constrained, no under-constrained subsystem can exist otherwise it would not be just-constrained), and in an under-constrained subsystem, several under-constrained ones (no over-constrained, no just-constrained) may exist.

Subsystems which cannot be decomposed into smaller ones are minimal subsystems.

Example 5.24 *Minimal subsystems*

The following over-constrained system

↗	x_1	x_2	x_3	x_4	x_5	x_6
c_1	1					1
c_2		1				1
c_3			1	1	1	1
c_4	1			1		
c_5		1			1	
c_6			1			1
c_7						1

contains six minimal subsystems, five being just-constrained

$$(\{c_7\},\ \{x_6\})$$
$$(\{c_1\},\ \{x_1,\ x_6\})$$
$$(\{c_2\},\ \{x_2, x_6\})$$
$$(\{c_6\},\ \{x_3,\ x_6\})$$
$$(\{c_4\},\ \{x_1,\ x_4\})$$

and the last one

$$\{c_5,\ c_3\}, \qquad \{x_2,\ x_3,\ x_4,\ x_5,\ x_6\}$$

being over-constrained. This can be seen from the following rearrangement of the rows and columns of the incidence matrix.

↗	x_6	x_1	x_2	x_3	x_4	x_5
c_7	1					
c_1	1	1				
c_2	1		1			
c_6	1			1		
c_4		1			1	
c_5			1			1
c_3	1			1	1	1

Example 5.25 *Reduced graph of the tank*

Re-arranging the rows and columns of the graph of the tank, the incidence matrix becomes:

↗	h	$\dot{h}$	q_o	q_i	u	y
c_4	1					1
c_6	1	1				
c_3	1		1			
c_2				1	1	
c_1		1	1	1		

Three minimal just-constrained subsystems

$$(\{c_4\},\, \{h\}), \quad \left(\{c_6\},\, \{h, \dot{h}\}\right), \quad (\{c_3\},\, \{h, q_o\})$$

and one over-constrained subsystem

$$\left(\{c_1,\, c_2\},\, \{\dot{h},\, q_o,\, q_i\}\right)$$

are obtained. Note that the decomposition is independent of the known variables, which are added to the right of the table for completeness. □

5.4.2 Interpretation of the canonical decomposition

In this section, the canonical subsystems are analysed from the point of view of the above assumptions, i.e. from the point of view of the existence of solutions, thus providing a key for the analysis of the system observability and controllability.

First, it is clear that Assumption 5.1 (a) must be satisfied by each of the subsets of constraints $\mathcal{C}^+$, $\mathcal{C}^0$ and $\mathcal{C}^-$. If this was not true, the system model would have no solution, which contradicts the fact that it describes the behaviour of a physical system (which indeed has a solution).

Second, in the structural point of view, any algebraic constraint is assumed to satisfy Assumption 5.2, thus a subset of n variables completely matched within a subset of n constraint, is uniquely defined, while the result depends on the causality and on the existence of differential loops when constraints of the form

$$d: \quad z_2(t) - \tfrac{d}{dt} z_1(t) = 0$$

are considered.

Finally, it will be seen that there are cases in which Assumption 5.1 (b) cannot hold true.

Static systems. For clarity, let us start the analysis with static systems, whose behaviour model contains only algebraic constraints.

In the **over-constrained subsystem**, $(\mathcal{C}^+, Q(\mathcal{C}^+))$, the variables $\mathcal{Z}^+$ have to satisfy more than n^+ constraints, which satisfy Assumption 5.2 and 5.1 (a), where n^+ is the number of variables. Therefore, they belong to the intersection of the manifolds associated with the constraints $\mathcal{C}^+$. Since there are more manifolds than variables no solution can exist if they also satisfy Assumption 5.1 (b). Because the model should have at least one solution for a given physical system, one concludes that the constraints in $\mathcal{C}^+$ are not independent, i.e. the system description is redundant. In other words, for the system to have a solution, some compatibility conditions must hold. Structural analysis always assumes the most general case, i.e. the minimum number of relations between the system parameters. This means that the number of independent constraints is maximal, thus leading to the following equivalent conclusions:

- The over-constrained subsystem has a unique solution (more generally it has a finite number of solutions).
- The number of independent constraints in $\mathcal{C}^+$ is n^+.
- The number of compatibility conditions is $|\mathcal{C}^+| - n^+$.

In the **just-constrained subsystem**, $(\mathcal{C}^0, Q(\mathcal{C}^0))$, the n^0 variables in the set $\mathcal{Z}^0$ have to satisfy exactly n^0 constraints, which satisfy Assumption 5.2 and 5.1 (a). A unique solution exists, which is the intersection of the manifolds associated with the constraints $\mathcal{C}^0$, which are assumed to satisfy Assumption 5.1 (b). This being the most general case, structural analysis proposes the following conclusions.

- The just-constrained subsystem has a unique solution.
- The number of independent constraints in $\mathcal{C}^0$ is n^0.
- There is no compatibility condition.

In the **under-constrained subsystem**, $(\mathcal{C}^-, Q(\mathcal{C}^-))$, the n^- variables in the set $\mathcal{Z}^-$ have to satisfy less than n^- constraints, which satisfy Assumption 5.2 and 5.1 (a). All what the model can tell is that the unique solution of the physical system belongs to the intersection of less than n^- manifolds, and thus it is not uniquely defined by the model. It belongs to a manifold of dimension $n^- - |\mathcal{C}^-|$ if the constraints also satisfy Assumption 5.1 (b). This being the most general case, structural analysis proposes the following conclusions.

- The under-constrained subsystem has no unique solution.
- The constraints in $\mathcal{C}^-$ are independent.
- There is no compatibility condition.

Example 5.26 *Compatibility conditions in an over-constrained subsystem*

Consider the set of linear constraints

$$\begin{aligned} c_1: &\quad a_1\,x_1 + b_1\,x_2 - y_1 = 0 \\ c_2: &\quad a_2\,x_1 + b_2\,x_2 - y_2 = 0 \\ c_3: &\quad a_3\,x_1 + b_3\,x_2 - y_3 = 0, \end{aligned} \tag{5.17}$$

where $a = (a_1,\, a_2,\, a_3)'$, $b = (b_1,\, b_2,\, b_3)'$ are known parameters and $y = (y_1,\, y_2,\, y_3)'$ are known variables. This system is clearly over-constrained with respect to the unknown variables $(x_1,\, x_2)$. Let us consider the following cases.

1. rank$[a,\, b,\, y] = 3$, i.e. the parameters a, b and the known variables y are independent vectors, i.e. their nine components can be chosen arbitrarily. The system has in general no solution, since the three constraints are incompatible. Assumptions 5.1 and 5.2 cannot hold simultaneously.
2. If rank$[a,\, b,\, y] = 2$, one solution exists. Note that the parameters and the known variables are no longer independent (namely $[a,\, b,\, y]$ has one null eigenvalue, thus $\exists\ \lambda,\, \mu \in R\backslash\{0\}$ such that $y = \lambda a + \mu b$, which is the compatibility condition). The unique solution is $x_1 = \lambda$ and $x_2 = \mu$.
3. If rank$[a,\, b,\, y] = 1$, $[a,\, b,\, y]$ has two null eigenvalues, thus $\exists\ \lambda,\, \mu \in R\backslash\{0\}$ such that $y = \lambda a = \mu b$, and more than one solution exists. Any pair (x_1, x_2) such that $x_1 + x_2 - \lambda\mu = 0$ satisfies the system of equations. Note that in that case, two compatibility conditions exist, and Assumption 5.1 does not hold.
4. The last case is rank$[a,\, b,\, y] = 0$, i.e. $a = b = y = 0$. In this case, all parameters are specified and any pair $(x,\, y)' \in \mathbb{R}^2$ satisfies the system of equations. Assumption 5.2 does not hold.

Since (5.17) is the behaviour model of a physical system, it should exhibit at least one solution. Then obviously the most general situation is case number 2 in which only one relation holds between the parameters. This is what is assumed by the structural point of view. □

Dynamical systems. Remember that, when differential constraints are considered, matching all the variables in a subsystem guarantees that there is a unique solution under integral causality, i.e. when the initial conditions are known. Under derivative causality, the solution is unique if and only if there is a matching which avoids differential loops.

Let n_1^+ (respectively n_1^0, n_1^-) be the maximal number of variables which can be matched in the over-constrained subsystem (respectively in the just-constrained, the under-constrained subsystems) without introducing any differential loop. One obviously has $n_1^+ \leq n^+,\ n_1^0 \leq n^0,\ n_1^- \leq n^-$.

The over-constrained (respectively the just-constrained) subsystem is called *causal* if there exists a complete matching with respect to the variables $\mathcal{Z}^+$ (respectively $\mathcal{Z}^0$) which does not contain any differential loop, i.e. if one has $n_1^+ = n^+$ (respectively $n_1^0 = n^0$). Note that the under-constrained subsystem can obviously not be causal, since there does not even exist any complete matching with respect to $\mathcal{Z}^-$.

When the over-constrained (respectively the just-constrained) subsystem is causal, the same conclusions as above obviously apply for the interpretation of the three

canonical subsystems. A non-causal subsystem does not exhibit all unique solution, since there are $n^+ - n_1^+$ variables in the over-constrained subsystem (respectively $n^0 - n_1^0$ variables in the just-constrained one) which are matched in differential loops, i.e. which are defined up to an unknown constant.

Example 5.27 *Causal over-constrained system*

As an example, consider the following system

$$\begin{aligned} c_1: &\quad x_2 - a\,x_1 - bu = 0 \\ c_2: &\quad x_2 - \alpha\,x_1 - \beta u = 0 \\ c_3: &\quad x_2 - \tfrac{d}{dt}x_1 = 0, \end{aligned}$$

which is over-constrained with respect to the variables $(x_1,\, x_2)$ and where u is supposed to be known. The system is causal since $(x_1,\, x_2)$ can be matched with $(c_1,\, c_2)$ and this introduces no differential loop. Thus, there is a unique solution, which is obtained from the intersection of the two manifolds associated with $(c_1,\, c_2)$:

$$\begin{aligned} x_1 &= \frac{\beta - b}{a - \alpha} u \\ x_2 &= \left(\frac{a\,\beta - \alpha\, b}{a - \alpha}\right) u. \end{aligned}$$

$a - \alpha$ is assumed not to be zero. Moreover, constraint c_3 is redundant, and acts as a compatibility condition which has to be satisfied for the system solution to exist, namely

$$\left(\frac{a\,\beta - \alpha\, b}{a - \alpha}\right) u - \frac{\beta - b}{a - \alpha}\dot{u} = 0.$$

Suppose now that constraint c_2 does not exist, then the system is just-constrained but it is not causal, and its solution is defined up to the constant $x_1(0)$, which is unknown under differential causality. □

5.5 Observability

5.5.1 Observability and computability

Known and unknown variables. The system variables $\mathcal{Z}$ are decomposed into known (the set $\mathcal{K}$) and unknown ones (the set $\mathcal{X}$). Known variables are available in real time, while unknown variables are not directly measured. However, there might exist some way to compute their value from the values of known ones (past and present values are considered in discrete time models, while variables and their derivatives are considered in continuous time models). Analysing the system observability coincides with identifying those unknown variables for which such possibility exists.

Considering the general system described by the Eqs. (5.1), (5.2), (5.3) and (5.4)

$$\dot{\boldsymbol{x}}_d = \boldsymbol{g}\,(\boldsymbol{x}_d,\, \boldsymbol{x}_a,\, \boldsymbol{u}) \tag{5.18}$$

$$\boldsymbol{0} = \boldsymbol{m}\,(\boldsymbol{x}_d,\, \boldsymbol{x}_a,\, \boldsymbol{u}) \tag{5.19}$$

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}_d, \boldsymbol{x}_a, \boldsymbol{u}) \tag{5.20}$$

$$\dot{\boldsymbol{x}}_d = \frac{d}{dt}\boldsymbol{x}_d, \tag{5.21}$$

the set of known variables is $\mathcal{K} = \boldsymbol{u} \cup \boldsymbol{y}$, the set of unknown variables $\mathcal{X} = \boldsymbol{x}_a \cup \boldsymbol{x}_d \cup \dot{\boldsymbol{x}}_d$ and the set of constraints $\mathcal{C} = \boldsymbol{g} \cup \boldsymbol{m} \cup \boldsymbol{h} \cup \frac{d}{dt}$. Following the decomposition of $\mathcal{Z}$ into $\mathcal{K} \cup \mathcal{X}$, $\mathcal{C}$ is decomposed into $\mathcal{C}_\mathcal{K} \cup \mathcal{C}_\mathcal{X}$, where

$$\begin{aligned} \mathcal{C}_\mathcal{K} &= \{c \in \mathcal{C};\ Q(c) \cap \mathcal{X} = \emptyset\} \\ \mathcal{C}_\mathcal{X} &= \{c \in \mathcal{C};\ Q(c) \cap \mathcal{X} \neq \emptyset\}. \end{aligned}$$

$\mathcal{C}_\mathcal{K}$ is obviously the largest subset of constraints such that $Q(\mathcal{C}_\mathcal{K}) \subseteq \mathcal{K}$. It can be noticed that the relations which define control algorithms belong to $\mathcal{C}_\mathcal{K}$ since they introduce constraints between the sensor output, the control objectives (set-points, tracked trajectories, final states) and the control output, which are all known variables. The aim being to analyse the possibility of computing the unknowns $\mathcal{X}$, only the subgraph $(\mathcal{C}_\mathcal{X}, \mathcal{X}, \mathcal{E}_\mathcal{X})$ needs to be decomposed.

Remark 5.3 *Observability and computability*
Consider the set $\mathcal{X} = \boldsymbol{x}_a \cup \boldsymbol{x}_d \cup \dot{\boldsymbol{x}}_d$ of the unknown variables. For the static variables $\boldsymbol{x}_a$, the term "observable" has obviously the same meaning as the term "computable". Namely, it means that their trajectories can be determined from the knowledge of the trajectories of the known variables. Consider a dynamical variable $x_d^i \in \boldsymbol{x}_d$. It appears along with its derivative in the system behaviour model. Therefore, several cases have to be considered:

Case 1:	x_d^i and $\dot{x}_d^i$ can be computed.
Case 2:	$\dot{x}_d^i$ can be computed, but not x_d^i.
Case 3:	x_d^i can be computed, but not $\dot{x}_d^i$.
Case 4:	nor x_d^i neither $\dot{x}_d^i$ can be computed.

Case 3 obviously cannot exist, since the knowledge of $x_d^i(t)$ implies the knowledge of $\dot{x}_d^i(t)$. In case 1, x_d^i and $\dot{x}_d^i$ are both computable and x_d^i is observable. In case 2, x_d^i is unobservable, although $\dot{x}_d^i$ is computable, and in case 4 no one is computable, and x_d^i is unobservable. Therefore, the set of computable variables includes (but is not restricted to) the set of observable variables, if "variable" means "component of the state vector" as usual in the control literature. In the framework of this chapter, since "variable" means any component in $\boldsymbol{x}_a \cup \boldsymbol{x}_d \cup \dot{\boldsymbol{x}}_d$, observability and computability do indeed coincide. For example in case 1, we say that variable $\dot{x}_d^i$ is observable and so is also variable x_d^i, in case 2, $\dot{x}_d^i$ is observable while x_d^i is not, and in case 4, no one of them is observable. □

5.5.2 Structural observability conditions

Let

$$\begin{aligned} \mathcal{S}^+ &= (\mathcal{C}_\mathcal{X}^+, \mathcal{X}^+) \\ \mathcal{S}^0 &= (\mathcal{C}_\mathcal{X}^0, \mathcal{X}^+ \cup \mathcal{X}^0) \\ \mathcal{S}^- &= (\mathcal{C}_\mathcal{X}^-, \mathcal{X}^+ \cup \mathcal{X}^0 \cup \mathcal{X}^-) \end{aligned}$$

be the canonical decomposition of the subgraph $(\mathcal{C}_{\mathcal{X}},\, \mathcal{X},\, \mathcal{E}_{\mathcal{X}})$ associated with system (5.18) – (5.21).

Theorem 5.3 (Structural observability)
A necessary and sufficient condition for system (5.18) – (5.21) to be structurally observable is that, under derivative causality

1. *all the unknown variables are reachable from the known ones,*
2. *the over-constrained and the just-constraint subsystems are causal,*
3. *the under-constrained subsystem is empty.*

Condition 1 expresses that there does not exist any subsystem whose behaviour is not reflected in the behaviour of the known variables, while conditions 2 and 3 express that all the variables can be matched using causal matchings, i.e. they belong to the intersection of at least as many manifolds as unknowns, and thus they are uniquely defined once the known variables are given.

Note that since derivative causality is invoked, condition 1 could as well have been stated as: all the variables $\boldsymbol{x}_a \cup \boldsymbol{x}_d$ are reachable from the known ones (since under derivative causality, $\dot{\boldsymbol{x}}_d$ can always be reached from $\boldsymbol{x}_d$).

Example 5.28 *Non-reachability*

Consider the following incidence matrix, in which the variable x_3 is not reachable from the output.

$\nearrow$	x_1	x_2	x_3	$\dot{x}_1$	$\dot{x}_2$	$\dot{x}_3$	u	y
c_1	1	1		1			1	
d_1	**x**			1				
c_2	1	1			1			
d_2		**x**			1			
c_3			1			1		
d_3			**x**			1		
m	1							1

The system equations associated with such a structure are obviously of the form

$$\begin{array}{lrcl} & \dot{x}_1 & = & g_1\,(x_1,\, x_2, u) \\ \text{Subsystem 1} & \dot{x}_2 & = & g_2\,(x_1,\, x_2) \\ & y & = & h\,(x_1) \\ \text{Subsystem 2} & \dot{x}_3 & = & g_3\,(x_3) \end{array} \tag{5.22}$$

and it is seen that subsystem 2 can by no means be observable. □

Example 5.29 *Observability of a nonlinear system*

Consider the following non-linear dynamical system with two states, two input signals, one parameter and one sensor:

$$\begin{aligned} c_1 &: \quad \dot{x}_1 = (\theta - 1) x_2 \, u_1 \\ c_2 &: \quad \dot{x}_2 = u_2 \\ m &: \quad y = x_1. \end{aligned}$$

This system is over-constrained, and it satisfies the three conditions of the above theorem. The following matching allows to compute the state.

$\nearrow$	$\dot{x}_1$	$\dot{x}_2$	x_1	x_2	u_1	u_2	y
c_1	1			①	1		
c_2		1				1	
d_1	①		x				
d_2		①		x			
m			①				1

It can be noticed that x_2 can be reached from the known variables if and only if the matching (c_1, x_2) can be used, which means that the two conditions

$$u_1 \neq 0 \quad \text{and} \quad \theta \neq 1$$

simultaneously hold. If not, the system is not observable, since there is no matching by means of which x_2 could be computed under derivative causality. This example emphasises the fact that structural properties provide results which are valid for almost every value of the system parameters *and variables*. It can be noticed that the system being over-constrained, constraint c_2 is not matched and provides an input-output relation whose expression is

$$\frac{d}{dt} \frac{\dot{y}(t)}{u_1(t)} = u_2(t). \quad \square$$

5.5.3 Observability of linear systems

Let us consider the linear time-invariant deterministic system

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) \tag{5.23}$$

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}, \tag{5.24}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are respectively of dimensions n and p. The state is observable if and only if the following condition holds

$$\exists\, s \in \mathcal{N} \quad \text{such that rank} \begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \ldots \\ \boldsymbol{CA^s} \end{pmatrix} = n, \tag{5.25}$$

for which a necessary condition is

$$\operatorname{rank} \begin{pmatrix} \boldsymbol{A} \\ \boldsymbol{C} \end{pmatrix} = n. \tag{5.26}$$

Note that (5.26) means, in structural terms, that the unknown variables $\boldsymbol{x}$ belong to a causal just-constrained or over-constrained subsystem, when derivative causality is imposed. The structural graph is

$\nearrow$	$\dot{\boldsymbol{x}}$	$\boldsymbol{y}$	$\boldsymbol{x}$
$\boldsymbol{d}$	$\boldsymbol{I}$		$\mathbf{x}$
$\boldsymbol{m}$		$\boldsymbol{I}$	$\boldsymbol{S}_C$
$\boldsymbol{c}$	$\boldsymbol{I}$		$\boldsymbol{S}_A$

where $\boldsymbol{d}$ are the mathematical constraints (which express that dots mean derivatives), $\boldsymbol{m}$ are the constraints from the measurement Eq. (5.24), and $\boldsymbol{c}$ are the system constraints (5.23). $\boldsymbol{S}_C$ and $\boldsymbol{S}_A$ are the structures associated with matrices $\boldsymbol{C}$ and $\boldsymbol{A}$. Since no variable in $\boldsymbol{x}$ can be matched from any constraint in $\boldsymbol{d}$, the system $(\boldsymbol{c} \cup \boldsymbol{m},\ \dot{\boldsymbol{x}} \cup \boldsymbol{x} \cup \boldsymbol{y})$ must indeed be over-constrained with respect to $\boldsymbol{x}$. It can be noted that this does not constitute a sufficient condition, because the system parameters might have values such that (5.25) – or (5.26) – is not satisfied.

Example 5.30 *Observability of linear systems*

Consider the unobservable linear time-invariant system

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & c \\ 0 & 0 & d \\ a & b & e \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{5.27}$$

$$y = \begin{pmatrix} 0 & 0 & f \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \tag{5.28}$$

where the parameters a, b, c, d, e, f can take any real value. Its structure is given by the incidence matrix

$\nearrow$	$\dot{x}_1$	$\dot{x}_2$	$\dot{x}_3$	x_1	x_2	x_3	y
c_1	1					1	
c_2		1				1	
c_3			1	1	1	1	
d_1	1			$\mathbf{x}$			
d_2		1			$\mathbf{x}$		
d_3			1			$\mathbf{x}$	
m						1	1

where the constraints c_1, c_2, c_3 are the system (5.27), the constraints d_1, d_2, d_3 express the derivative link between the x_1, x_2, x_3 and the $\dot{x}_1$, $\dot{x}_2$, $\dot{x}_3$ (remember that only derivative

causality is allowed) and m is the measurement Eq. (5.28). This system can be decomposed into a just-constrained part $\mathcal{C}_{\mathcal{X}}^{0} = \{c_1, c_2, d_3, m\}$, $\mathcal{X}^{0} = \{\dot{x}_1, \dot{x}_2, \dot{x}_3, x_3\}$ from which $\dot{x}_1$, $\dot{x}_2$, $\dot{x}_3$ and x_3 can be computed as functions of y, (for almost all values of the parameters), and an under-constrained one $\mathcal{C}_{\mathcal{X}}^{-} = \{c_3, d_1, d_2\}$, $\mathcal{X}^{-} = \{x_1, x_2\}$ in which x_1 and x_2 should both be computed from constraint c_3 (since they can be matched neither with d_1 nor with d_2). It can be checked that adding $\dot{y}$ and the associated constraints, the subsystem $(\{c_3, d_1, d_2\}, \{x_1, x_2\})$ remains under-constrained and that this will always be the case when more signals $y^{(i)}$ will be considered. This means that the information available from the sensor is enough to place (x_1, x_2) in a subspace of dimension one (since they are linked by one constraint which is known to be linear), but is not enough to compute the actual value of this vector. The observability matrix

$$\begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \boldsymbol{CA}^2 \end{pmatrix} = \begin{pmatrix} 0 & 0 & f \\ af & bf & ef \\ aef & bef & (ac+bd+e^2)f \end{pmatrix}$$

is not full rank, whatever the coefficients a, b, c, d, e, f, and it can be checked that no more than the linear form $ax_1 + bx_2$ can be determined from the observation $(y, \dot{y}, ...y^{(s)})$ for any $s \geq 1$.

Consider the same linear time-invariant system, in which the second state is now measured (the system is now observable)

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & c \\ 0 & 0 & d \\ a & b & e \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{5.29}$$

$$y = \begin{pmatrix} 0 & f & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}. \tag{5.30}$$

The structural graph is

$\nearrow$	$\dot{x}_1$	$\dot{x}_2$	$\dot{x}_3$	x_1	x_2	x_3	y
c_1	1					1	
c_2		1				1	
c_3			1	1	1	1	
d_1	1			**x**			
d_2		1			**x**		
d_3			1			**x**	
m					1		1

and the following causal matching shows that all the components of the state can be computed from y and its derivatives.

↗	$\dot{x}_1$	$\dot{x}_2$	$\dot{x}_3$	x_1	x_2	x_3	y
c_1	1					1	
c_2		1				①	
c_3			1	①	1	1	
d_1	①			x			
d_2		①			x		
d_3			①			x	
m					①		1

□

5.5.4 Graph-based interpretation and formal computation

Since an oriented graph can be associated with each matching, the observability property can be analysed from a graph-theoretical point of view. Let x be an observable variable. Then x can be matched with a constraint the input of which is either known or a set of observable variables. Repeating this argument, it follows that for x to be observable, it is necessary that there exists at least one subgraph (a set of alternated chains) which links this variable (call it the target) and the set of the known ones $\boldsymbol{u} \cup \boldsymbol{y}$, and that no non-observable variable acts as an input in any constraint of this subgraph. This means that the subgraph with the observable target x may contain algebraic loops, but it does not contain any differential loop.

The constraints along the alternated chains show the computations which are to be performed in order to compute x (they could be automatically analysed in order to provide the formal expression of x). The crossing of a simple algebraic constraint means that the matched variable is computed as a function of the non-matched ones. An algebraic loop shows that a system of simultaneous constraints has to be solved, providing as a solution all the variables matched within the loop. The crossing of a derivative constraint means that the non-matched variable has to be derivated in order to obtain the matched variable (remember that only derivative causality is allowed). The number of derivative constraints which are crossed between a given input and the target shows the maximum order of derivation needed on this input for computing this target.

Note that this interpretation expresses that x belongs to a just or an over-constrained causal subsystem. If x were to belong to an under-constrained subsystem, the corresponding subgraph would have less constraints than variables, i.e. some unknown variables would be input signals to constraints while being output of no other constraint (i.e. being not matched, thus non-observable). Figure 5.23 shows the two graphs associated with the linear systems (5.27), (5.28) and (5.29), (5.30) which are respectively non-observable and observable. It can be seen that in the first case, either x_2 or x_1 stands as an unknown input of constraint c_3 while in the second case, both can be matched thus providing all the states with known predecessors at some level.

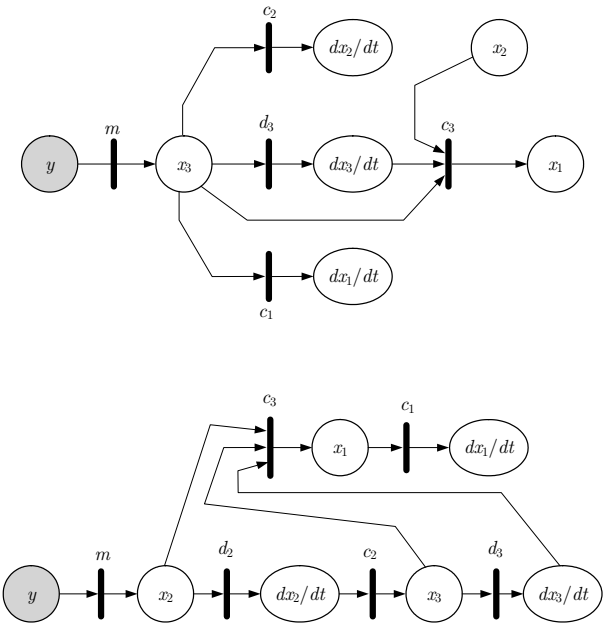


Fig. 5.23. Graph-based interpretation of the observability property

When different estimation subgraphs with the same target exist, they provide different computation schemes for the same variable. This feature is of interest when monitorability and reconfigurability are considered.

5.6 Monitorability

A system is said to be *monitorable* if it can be determined, using only (trajectories of) known variables, whether the system constraints are satisfied or not. This section is concerned with the analysis of system monitorability, and with the design of fault detection and isolation algorithms based on *Analytical Redundancy Relations* (ARRs). Analytical redundancy-based fault diagnosis tries to identify faults by comparing the actual behaviour of the system, which is observed through the time evolution of the known variables, with the theoretical behaviour described by the system constraints. This comparison can be performed only if some redundancy exists in the system. ARRs are the constraints that express this redundancy. In this section, the analytical redundancy relation-based approach to fault diagnosis is first briefly recalled and stated in the frame of structural analysis, leading to characterise the structurally monitorable part of the system. The problem of designing robust and structured residuals is then addressed.

5.6.1 Analytical redundancy-based fault detection and isolation

Analytical redundancy relations are static or dynamical constraints which link the time evolution of the known variables when the system operates according to its normal operation model. Once ARRs are designed, the fault detection procedure checks at each time whether they are satisfied or not, and when not, the fault isolation procedure identifies the system component(s) which is (are) to be suspected. The existence of ARR is thus a prerequisite to the design of fault diagnosis procedures. Moreover, in order for the fault diagnosis procedure to work properly, ARR should have the following properties:

- Robust, i.e. insensitive to unknown input and unknown parameters. This insures that they are satisfied when no fault is present, so that false alarms are not issued by the fault diagnosis algorithm.
- Sensitive to faults: This insures that they are not satisfied when faults are present, so that there is no missed detection.
- Structured: This insures that in the presence of a given fault, only a subset of the ARRs are not satisfied, thus allowing to recognise (from the subset of satisfied and the subset of not satisfied ARR), the fault which occurred.

Faults. Analysing the fault diagnosis possibilities of a system needs faults to be precisely defined. In structural analysis, a fault is defined as *a change in some constraint*. A system is the interconnection of a number of components, each of them being described by its behaviour model in normal operation. Let $\{COMP_i, i \in \Sigma\}$ be the set of the system components. Each of them is a subsystem $(\phi_i,\ Q(\phi_i))$ which imposes the set of constraints ϕ_i to the system variables $Q(\phi_i)$, where $Q(\phi_i) \cap \mathcal{X}$ are unknown (unmeasured state variables, unknown input, unknown parameters) while $Q(\phi_i) \cap \mathcal{K}$ are known (input, output, known parameters). A fault in component $COMP_i$ is defined as a change in at least one of the constraints $\varphi \in \phi_i$ (the notation φ is used – as a mnemonic for "fault" – instead of c which was a mnemonic for "constraint"). Note that this allows to consider different fault modes associated with the same component (each subset of ϕ_i can in fact be considered as a fault mode of $COMP_i$). Note also that since only the structure is of interest, there is no need to define, nor to model the nature of the change (e.g. using additive or multiplicative fault models).

Example 5.31 *An insulated pipe*

Consider an insulated pipe, and suppose one is interested in modelling the mass and the heat transfers. A simple model of the pipe is given by the two constraints

$$\begin{aligned} \varphi_1 \quad &: \quad q_i(t) - q_o(t) = 0 \\ \varphi_2 \quad &: \quad q_i(t)\,\theta_i(t) - q_o(t)\,\theta_o(t) = 0, \end{aligned}$$

where q_i (respectively q_o) is the input (respectively the output) massic flow of the (incompressible) fluid, and θ_i (respectively θ_o) is the input (respectively the output) fluid temperature. A defect in the insulation would obviously result in φ_2 being violated, while a leak in the pipe would be modeled by φ_1 and φ_2 being violated. □

Considering the whole set of components, the overall system is $(\mathcal{C}, \mathcal{Z})$, where

$$\begin{aligned} \mathcal{C} &= \cup_{i\in\Sigma}\, \phi_i \\ \mathcal{Z} &= \cup_{i\in\Sigma}\, Q(\phi_i). \end{aligned}$$

Direct redundancy. Consider any constraint $\varphi \in \mathcal{C}_\mathcal{K}$ (remember that $\mathcal{C}_\mathcal{K}$ is the subset of constraints such that $Q(\mathcal{C}_\mathcal{K}) \subseteq \mathcal{K}$), and let $COMP$ be the component to which φ belongs. This constraint is an ARR since it links only known variables, and it can be checked in real time if it is satisfied or not, by taking the numerical values of the known variables, putting them into constraint φ, and testing whether the result is ZERO or not. Note that when the constraint is not satisfied, it can be concluded that the system is not in normal operation, while when the constraint is satisfied it can only be said that the normal operation hypothesis is not contradicted (or falsified) by the values of the observations.

In practical situations, variables are not very precisely known, measurements are corrupted by noise, and models only approximate the system actual behaviour, thus the obtained value will never be exactly zero, even in normal operation. Let $r_\varphi(\mathcal{K})$ be the obtained value. $r_\varphi(\mathcal{K})$ is called the *residual* associated with ARR φ, and fault detection boils down to decide whether it is small enough so that the ZERO hypothesis can be accepted. Fault isolation obviously follows fault detection since only a fault in component $COMP$ could cause constraint φ not to be satisfied.

In all systems, the control algorithms are direct ARR, since the subset $\mathcal{C}_\mathcal{K}$ includes the constraints which describe them. Hence, they can be used to check whether the controller is working properly. Although this might be of practical interest, such direct redundancy relations are of little interest as far as structural analysis is concerned, because the result is obvious. Therefore, the aim of the following is to find ARRs in the subsystem $(\mathcal{C}_\mathcal{X}, \mathcal{Z})$ which includes unknown variables.

Deduced redundancy. Consider some constraint $\varphi \in \mathcal{C}_\mathcal{X}$ and again let $COMP$ be the component to which φ belongs. Let $\mathcal{X}_\varphi = Q(\varphi) \cap \mathcal{X}$ be the subset of unknowns which appear in constraint φ, and suppose

$$\mathcal{X}_\varphi \subseteq \mathcal{X}_{obs}, \tag{5.31}$$

where $\mathcal{X}_{obs}$ is the subset of the observable variables. Then, any variable $x \in \mathcal{X}_\varphi$ can be expressed as a function of the known ones (including eventually their derivatives), using the normal operation system model (this results from the existence of one or several alternated chains between the known variables and the target x). Suppose that there exists at least one alternated chain with target x which does not cross constraint φ (this means that even when constraint φ is removed, x can still be matched and computed as a function of the known variables, which indicates that constraint φ belongs to an over-constrained subsystem, as it will be seen later). When this is true, this alternated chain can be used to compute x as a function of the known variables, and one can put the obtained expression into φ, which obviously produces an ARR. The associated residual $r_\varphi(\mathcal{K})$ should be ZERO when the

system operates properly, which is used, as previously, for fault detection. However, fault isolation will be slightly different since the residual associated with φ will be nonZERO not only when $COMP$ is not performing well, but also when the actual values of the $\mathcal{X}_\varphi$ variables are different from those computed from the observations via the normal operation model. This may happen when the fault changes some constraint which belongs to an alternated chain whose target is in $\mathcal{X}_\varphi$. The conclusion is that when $r_\varphi(\mathcal{K})$ is nonZERO, there is an associated set of components to be suspected instead of a single one[2]. It can be easily determined from the graph-based interpretation.

Example 5.32 *Single-tank system*

Consider the tank whose structure graph is shown in Fig. 5.3. Obviously, there are two redundancy relations. The first one is given by constraint c_5 and is of no interest since it is a direct redundancy relation which only duplicates the control algorithm. The second one is given by c_2 which should be satisfied when the system operates normally, and which will be false if one of the constraints $\{c_1, c_2, c_3, c_4\}$ is not satisfied (c_6 is a mathematical constraint which is not linked with any hardware or software component and thus it cannot be faulty). □

5.6.2 Structurally monitorable subsystems

Unfortunately, not every fault can be detected. It is easily seen that when (5.31) does not hold, it is impossible to obtain any ARRs using constraint φ. Thus, any fault which would change constraint φ could not be detected.

Definition 5.8 (Structurally monitorable subsystem)
The structurally monitorable part of the system is the subset of the constraints such that there exists ARRs which are structurally sensitive to their change.

Then, from above, it can be characterised by the following theorem:

Theorem 5.4 (Monitorability)
The following two necessary conditions for a fault φ to be monitorable are equivalent:

(i) $\mathcal{X}_\varphi$ is structurally observable - according to (5.31) - in the system $(\mathcal{C}\backslash\{\varphi\}, \mathcal{Z})$,
(ii) φ belongs to the structurally observable over-constrained part of the system $(\mathcal{C}, \mathcal{Z})$.

Let $(\mathcal{C}_\mathcal{X}, \mathcal{X})$ be a structurally observable over-constrained subsystem, then there exists a subset $\mathcal{S}_\mathcal{X} \subset \mathcal{C}_\mathcal{X}$ of $n = |\mathcal{X}|$ constraints which (from a structural point of

[2] This set is called the *structure of the residual* in the control community and it is called a *conflict* in the Artificial Intelligence community.

view) can be solved uniquely for the variables $\mathcal{X}$ (notation $\mathcal{S}$ is used as a mnemonic for Solve). These variables can thus be computed as functions of the known variables $\mathcal{K}$. Putting the obtained values into the remaining constraint set $\mathcal{R}_\mathcal{X} = \mathcal{C}_\mathcal{X} \backslash \mathcal{S}_\mathcal{X}$ (notation $\mathcal{R}$ is used as a mnemonic for Remaining, or Redundant), one obtains $|\mathcal{C}_\mathcal{X}| - |\mathcal{X}|$ relations which link only known variables and which are, therefore, redundancy relations. For a more convenient notation the function

$$\mathcal{X} = \Gamma_\mathcal{X}(\mathcal{K}) \tag{5.32}$$

is introduced for the computation of the unknown variables, leading to expressing the set of constraints $\mathcal{C}_\mathcal{X}$ under the equivalent form

$$\begin{aligned} \mathcal{S}_\mathcal{X} : &\qquad \mathcal{X} - \Gamma_\mathcal{X}(\mathcal{K}) = 0 \\ \mathcal{R}_\mathcal{X} : &\qquad (\mathcal{C}_\mathcal{X} \backslash \mathcal{S}_\mathcal{X}) \circ \Gamma_\mathcal{X}(\mathcal{K}) = 0, \end{aligned} \tag{5.33}$$

where $\circ$ means the substitution of $\mathcal{X}$ by $\Gamma_\mathcal{X}(\mathcal{K})$.

It can be noted that in general, several different complete matchings can be performed in a given causal over-constrained subsystem, thus providing different means of computing the unknown variables $\mathcal{X}$ from the known ones. This fact will be used when fault-tolerant observation schemes will be considered, but it can also provide another interpretation of redundancy, since obviously the unknown variables $\mathcal{X}$ have to be the same for all matchings. For example suppose two matchings exist such that $\mathcal{X}$ is associated with $\mathcal{S}_\mathcal{X} \subset \mathcal{C}_\mathcal{X}$ in the first one, providing $\mathcal{X} = \Gamma_\mathcal{X}(\mathcal{K})$, and with $\mathcal{P}_\mathcal{X} \subset \mathcal{C}_\mathcal{X}$ in the second one, providing $\mathcal{X} = \Lambda_\mathcal{X}(\mathcal{K})$. The redundancy relations

$$\Gamma_\mathcal{X}(\mathcal{K}) - \Lambda_\mathcal{X}(\mathcal{K}) = 0$$

directly follows from the fact that the two results should obviously be the same.

Example 5.33 *Sensor redundancy*

The simplest illustration of this idea is provided by sensor hardware redundancy. Suppose that two sensors measure the same unknown variable x. The measurement equations are given by

$$\begin{aligned} &\text{Sensor 1} \quad c_1 : \quad y_1 - x - \varepsilon_1 = 0 \\ &\text{Sensor 2} \quad c_2 : \quad y_2 - x - \varepsilon_2 = 0, \end{aligned}$$

where ε_1 and ε_2 denote measurement noise with known distribution. The structure graph has the following incidence matrix.

	known				unknown
$\nearrow$	y_1	y_2	ε_1	ε_2	x
c_1	1		1		1
c_2		1		1	1

ε_1 and ε_2 are here considered as known variables because their probability distribution is known. This system is over-constrained with $\mathcal{C}_\mathcal{X} = \{c_1, c_2\}$ and $\mathcal{X} = \{x\}$. The unknown x can be matched with each of the two constraints and, hence, be calculated by each of the sensor equations. This is not only true from the structural point of view because x can also be

calculated numerically if $\frac{dc_1}{dx}$ and $\frac{dc_2}{dx}$ are both non-zero. Otherwise at least one of the sensors would be completely useless.

For the matching

	known				unknown
↗	y_1	y_2	ε_1	ε_2	x
c_1	1		1		①
c_2		1		1	1

the oriented graph is given by Fig. 5.24, in which the unknown x is computed by

$$x = \gamma_1(y_1, \varepsilon_1)$$

and c_2 is used as a redundancy relation which can be written as

$$c_2(\gamma_1(y_1, \varepsilon_1), y_2, \varepsilon_2) = 0.$$

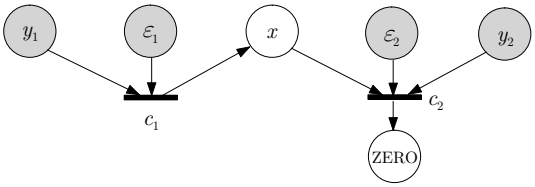


Fig. 5.24. Oriented structure graph for the sensor redundancy investigations

Choosing the second possible matching

	known				unknown
↗	y_1	y_2	ε_1	ε_2	x
c_1	1		1		1
c_2		1		1	①

provides

$$x = \gamma_2(y_2, \varepsilon_2)$$

and the redundancy relation

$$c_1\left(y_1, \varepsilon_1, \gamma_2\left(y_2, \varepsilon_2\right)\right) = 0.$$

Since two matchings exist, the remark that the (same) value of x can be computed either from the first or from the second one leads to the redundancy relation, which takes the form

$$\gamma_1\left(y_1, \varepsilon_1\right) - \gamma_2\left(y_2, \varepsilon_2\right) = 0. \quad \square$$

5.6.3 Design of analytic redundancy relations

Redundancy relations are subgraphs of the structure graph, which are associated with complete causal matchings of the unknown variables associated with the over-constrained subsystem of the reduced bi-partite graph. Redundancy relations are composed of alternated chains, which start with known variables and which end with non-matched constraints whose output is labelled ZERO. Designing a set of residuals calls for building maximal matchings on the given structural graph, under derivative causality, and identifying the redundancy relations as the non-matched constraints in which all the unknowns have been matched. Algorithms which find maximal matchings have been previously presented (cf. Section 5.3.5), and some hints have been given, using the tank and the sensor hardware redundancy examples, on the residuals design procedure. Let us now give a complete illustration, first using the simple single-tank system, and then giving a larger practical example with the two-tank system.

Example 5.34 *Ranking the constraints of the single-tank system*

Consider once more the single-tank example, and recall the incidence matrix of its reduced structure graph from Example 5.8.

$\nearrow$	h	$\dot{h}$	q_i	q_0
c_1		1	1	1
c_2			1	
c_3	1			1
c_4	1			
c_6	**x**	1		

The result of the ranking algorithm is shown in the following table and in Fig. 5.25. The matching is identical with the second matching on Example 5.8. Note that a new column has been introduced to mark constraints which have the output ZERO. Since ZERO is not a variable, it may be matched several times.

	unknown				Ranking	
$\nearrow$	h	$\dot{h}$	q_i	q_0	ZERO	Rank
c_1		1	1	1	①	2
c_2			①			0
c_3	1			①		1
c_4	①					0
c_6	**x**	①				1

Sorted according to the rank, the following constraint set is obtained.

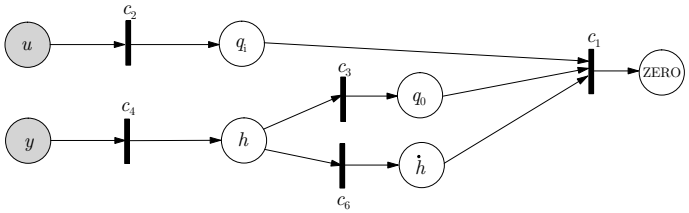


Fig. 5.25. Ranking for the single tank system

Rank	Constraint	Output
0	c_2	$q_i(t)$
	c_4	$h(t)$
1	c_3	$q_o(t)$
	c_6	$\dot{h}$
2	c_1	ZERO

If the reduced structure graph is redrawn according to the ranking of the constraints, Fig. 5.25 is obtained. The figure shows how the internal variables q_i, h, q_0 and $\dot{h}$ can be successively determined. The constraints are ordered according to their associated rank. Finally, the constraint c_1 is used to test wether the variables are consistent with the model.

As all constraints are ranked, the system is fully observable and monitorable. By solving the constraints for the matched variables, the following equations are obtained. The right hand column shows the path of the matching.

$$\begin{array}{lll} c_2: & q_i(t) = \alpha \cdot u(t) & c_2(u) \rightarrow q_i \\ c_4: & h(t) = y(t) & c_4(y) \rightarrow h \\ c_3: & q_0(t) = k\sqrt{h(t)} & c_6(h) \rightarrow q_0 \\ c_6: & \dot{h}(t) = \frac{d}{dt}h(t) & c_6(h) \rightarrow \dot{h} \\ c_1: & 0 = \dot{h}(t) + q_o(t) - q_i(t) & c_1(\dot{h}, q_i, q_0) \rightarrow \text{zero} \end{array} \tag{5.34}$$

These equations can be simplified to obtain the redundancy relations in analytic form:

$$c_1: \quad 0 \quad = \frac{d}{dt}y(t) + k\sqrt{y(t)} - \alpha u(t).$$

The order of operations on constraints was

$$c_1(c_6(c_4(y)), c_2(h), c_3(c_4(y))) \rightarrow \text{zero}.$$

Note that all variables on the right-hand side of the two equations are known. Hence, these equations can be tested for given measurements u and y which are marked in Fig. 5.25 to illustrate this fact. □

5.6.4 Structural detectability and isolability

Assume the over-constrained sub-system has been determined by finding a complete matching on the unknown variables. Then, the main results of structural analysis are obtained from the following steps.

- List all analytic redundancy relations that exist for the system.
- For all such relations, determine an explicit form if the constraints are explicitly known.
- List which violations of constraints are detectable.
- List which violations of constraints isolable.

When a matching has been found, the set $\mathcal{C}^{(u)} \subset \mathcal{C}$ of unmatched constraints $\mathcal{C}^{(u)} = \{c : c(x_c, k_c) \rightarrow 0, x_c \in \mathcal{X},\ k_c \in \mathcal{K}\}$ is determined. To obtain analytical redundancy relations for diagnosis (also called parity relations), the unknown variables in each $c \in \mathcal{C}^{(u)}$ must be substituted by known ones entering through matched constraints. Backtracking along alternated chains in the matching will enable such an elimination of the unknown variables. Finally, each unmatched constraint c will give one parity relation r to be used for diagnosis, since a violation of any constraint that is used in constructing the parity relation would give a non-zero residual when all known variables are replaced by their values.

Furthermore, analytical redundancy relations show with residuals depend on which constraints. One view on these relations is the Boolean mapping,

$$F : r \leftarrow M \qquad (c \neq 0)$$

from which structural detectability can be analysed. It can be checked that the following definition is the practical translation of the monitorability condition in Theorem 5.5.

Lemma 5.1 (Structural detectability)
A violation of a constraint c is structurally detectable if and only if it has a nonzero BFoolean signature in some residual r

$$c \in \mathcal{C}_{\text{detectable}} \Leftrightarrow \exists r :\ c \neq 0 \ \Rightarrow\ r \neq 0.$$

Moreover, since for a given constraint c the set of all parity relations can be partitioned into those in which its Boolean signature is zero and those in which its Boolean signature is non-zero, the following definition is straightforward.

Lemma 5.2 (Structural isolability)
A violation of a constraint c_i is structurally isolable if and only if it has a unique signature in the residual vector, i.e. column m_i of M is independent of all other columns in M

$$c_i \in \mathcal{C}_{\text{isolable}} \Leftrightarrow \forall j \neq i : m_i \neq m_j.$$

Example 5.35 *Single-tank parity relation - backtracking*

The single-tank example above had one unmatched constraint c_1. Backtracking through the matching obtained

$$c_1(c_6(c_4(y)), c_2(u), c_3(c_4(y))) \rightarrow zero$$

The way constraints enter into the associated parity relation (residual) is

$$r_1 \leftarrow \begin{bmatrix} c_1 & c_2 & c_3 & c_4 & c_6 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Hence a violation in any of the constraints is structurally detectable but none are structurally isolable. □

Example 5.36 *Nonlinear parity relation for ship*

Given the nonlinear model of a ship with dual measurements of heading angle ψ and with no disturbance from waves,

$$\begin{array}{lllll} c_1: & \dot{\omega}_3 & = & b(\eta_1\omega_3 + \eta_3\omega_3^3) + b\delta \\ c_2: & \dot{\psi} & = & \omega_3 \\ d_1: & \frac{d\omega}{dt} & = & \dot{\omega} \\ d_2: & \frac{d\psi}{dt} & = & \dot{\psi} \\ m_1: & y_1 & = & \psi \\ m_2: & y_2 & = & \psi \\ m_3: & y_3 & = & \dot{\psi} \\ m_4: & y_4 & = & \delta. \end{array}$$

The set of unknown variables is $X = \{\delta, \omega_3, \dot{\omega}_3, \psi, \dot{\psi}, \omega_w\}$, the set of known variables is $K = \{y_1, y_2, y_3\}$. A complete matching on the unknown variables is traced in the left column below, the right column shows the backtracking to known variables.

$$\begin{array}{llll} m_1(y_1) & \rightarrow & \psi & \\ m_2(y_2, \psi) & \rightarrow & zero & \Rightarrow \quad m_2(y_2, m_1(y_1)) \rightarrow zero \\ m_3(y_3) & \rightarrow & \dot{\psi} & \\ m_4(y_4) & \rightarrow & \delta & \\ d_2(\psi, \dot{psi}) & \rightarrow & zero & \Rightarrow \quad d_2(m_2(y_2), m_3(y_3)) \rightarrow zero \\ c_2(\dot{\psi}) & \rightarrow & \omega_3 & \\ d_1(\omega_3) & \rightarrow & \dot{\psi} & \\ c_1(\delta, \omega_3, \dot{\omega}_3) & \rightarrow & zero & \Rightarrow \quad c_1(m_4(y_4), c_2(m_3(y_3)), d_1(m_3(y_3))) \rightarrow zero \end{array} \tag{5.35}$$

The way constraints are used in the three parity relations is as follows,

$$\begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} \leftarrow \begin{array}{c} \begin{matrix} m_1 & m_2 & m_3 & m_4 & c_1 & c_2 & d_1 & d_2 \end{matrix} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array}.$$

A violation of any constraint is mapped onto the residuals so all faults are detectable. Considering isolability, five columns are independent: m_1, m_2, m_3, m_4, d_2 hence it is only violations in these constraints that are structurally isolable.

The analytical form of the parity relations are obtained from the symbolic expressions from the backtracking. This gives the expected result

$$\begin{array}{lll} r_1 & = & y_2 - y_1 \\ r_2 & = & \dot{y_2} - y_3 \\ r_3 & = & \dot{y_3} - b(\eta_1 y_3) + \eta_3 y_3^3) - b y_4 \ . \square \end{array} \tag{5.36}$$

5.6.5 Design of robust and structured residuals

Robust residuals. The set of constraints that describe the nominal operation of the system never represents its actual behaviour. Discrepancies follow from the existence of unknown inputs (or disturbances) and from the fact that system parameters values are never exactly known (uncertain parameters). Such discrepancies may result in residuals firing false alarms.

Example 5.37 *Residual discrepancies caused by unknown inputs*

The unknown variables of the single-tank example were computed from the following constraints

Component	Constraint number	Constraint expression
Pump	c_2	$q_i(t) = \alpha \cdot u(t)$
Level sensor	c_4	$h(t) = y(t)$
Output pipe	c_3	$q_0(t) = k\sqrt{h(t)}$
Mathematical constraint	c_6	$\dot{h}(t) = \frac{d}{dt}h(t).$

Putting their expression into constraint

$$c_1 : \dot{h}\,(t) - q_i(t) + q_o\,(t) = 0,$$

which describes the tank, the residual

$$r(t) = \dot{y}(t) + k\sqrt{y(t)} - \alpha u(t)$$

was obtained. Assume that the level sensor output is affected by a constant bias δ (unknown input):

Component	Nominal model constraint	Actual constraint
Level sensor	$h(t) = y(t)$	$h(t) = y(t) - \delta.$

Simple calculations show that the residual computed using the nominal model constraints would have a non-zero value

Case	Residual value
Nominal	$\dot{y}(t) + k\sqrt{y(t)} - \alpha u(t) = 0$
Sensor bias	$\dot{y}\,(t) - \alpha \cdot u(t) + k\sqrt{y(t) - \delta} = k\left(\sqrt{y(t) - \delta} - \sqrt{y(t)}\right) \neq 0.$ □

Example 5.38 *Residual discrepancies caused by uncertain parameters*

Consider now the two following cases

Component	Nominal model constraint	Actual constraint
Pump	$q_i(t) = \alpha \cdot u(t)$	$q_i(t) = \bar{\alpha} \cdot u(t)$
Output pipe	$q_0(t) = k\sqrt{y(t)}$	$q_0(t) = \bar{k}\sqrt{y(t)},$

which refer to uncertainties in the pump and output pipe parameters. Then, the residual computed using the nominal model constraints would have the following values

Case	Residual value
Nominal	$\dot{y}(t) + k\sqrt{y(t)} - \alpha u(t) = 0$
Uncertainty on the pump model	$\dot{y}(t) + k\sqrt{y(t)} - \bar{\alpha} \cdot u(t) = (\alpha - \bar{\alpha})\, u(t) \neq 0$
Uncertainty on the output pipe model	$\dot{y}(t) - \alpha \cdot u(t) + \bar{k}\sqrt{y(t)} = \left(\bar{k} - k\right) \sqrt{y(t)} \neq 0.$

□

Robustness refers to the property that residuals would not fire any false alarm as the result of unknown inputs acting on the system, or as the result of uncertainties in the values of the system parameters. One means of designing robust residuals is the so-called exact decoupling approach, in which the designed residuals are insensitive to unknown input and unknown or uncertain parameters. Therefore, they are satisfied when no fault is present, whatever the value of the unknown input or uncertain parameters. Note that the robustness problem is automatically solved in structural analysis, using the exact decoupling approach presented in Chapter 6, since it exhibits ARRs which are, by definition, only dependent on known variables. Unknown variables which affect the structurally monitorable subsystem are eliminated so that no residual can depend on them. When unknown variables cannot be eliminated, the part of the system they affect is not monitorable. When uncertain parameters are present, the solution to the exact decoupling problem is simply to design the fault diagnosis system considering them as unknown variables (this boils down to use the subset of residuals in which no uncertain parameter intervenes). Of course, the number of ARRs will in that case be smaller.

Structured residuals. As defined above, the structure of a residual is the set of the constraints which can be suspected when this residual is not ZERO. Let $\mathcal{R}$ be a set of residuals, and let $\Phi(r) \in 2^{\mathcal{C}}$ be the structure of residual $r \in \mathcal{R}$. This means that r is expected to be non-ZERO when at least one of the constraints in $\Phi(r)$ is faulty (in fact r will be non-ZERO only for detectable faults). Similarly, when some constraint $\varphi \in \mathcal{C}$ is faulty, then all the residuals whose structure contains φ are expected to be non-ZERO. The pattern of ZERO and non-ZERO residuals associated with a given fault is called its signature. Faults which have different signatures are isolable from each other, while faults which share the same signature are non-isolable. Let $\mathcal{R} = \mathcal{R}_0(t) \cup \mathcal{R}_1(t)$ be the decomposition of the set of residuals provided at some given time t by the decision procedure, where $\mathcal{R}_0(t)$ is the subset of the ZERO residuals and $\mathcal{R}_1(t)$ is the subset of non-ZERO ones. The subset of *suspected* constraints (the constraints which might be unsatisfied) at time t is given by

$$\mathcal{C}_{\text{susp}}(t) = \cap_{r \in\, \mathcal{R}_1(t)}\, \Phi(r).$$

Note that it is possible to define the subset of *exonerated* constraints (the constraints which are certainly satisfied) at time t by

$$\mathcal{C}_{\text{exo}}(t) = \cup_{r \in\, \mathcal{R}_0(t)}\, \Phi(r),$$

but one must be aware that this supposes all faults to be detectable. Exoneration is based on the assumption that if a constraint is not satisfied then it will necessarily show through the residuals whose structure it belongs to. The diagnosis at time t is

$$\mathcal{C}_{\mathrm{diag}}(t) = \mathcal{C}_{\mathrm{susp}}(t) \setminus \mathcal{C}_{\mathrm{exo}}(t).$$

In order to obtain good isolability properties, it may be of interest to design residuals with given structure. Suppose one wishes to design residuals which are insensitive to the faults of a subset of constraints $\mathcal{C}'$ and are sensitive to the faults of the subset of constraints $\mathcal{C} \setminus \mathcal{C}'$. A direct approach to do this is to consider only the system $(\mathcal{C} \setminus \mathcal{C}', \mathcal{Z})$ in the design process. However, from the structural monitorability condition, it is seen that the residuals can be made sensitive only to the faults in the monitorable subsystem of $(\mathcal{C} \setminus \mathcal{C}', \mathcal{Z})$, which may be smaller than that of $(\mathcal{C}, \mathcal{Z})$, since the former contains less constraints.

Example 5.39 *Structured redundancy relations for the two-tank system*

Consider the two-tank system and assume that the flow q_{12} between the two tanks can be measured in addition to the input u and the outflow q_m, which leads to the additional measurement constraint

$$m_1 : \quad q_{12} = q_{12,m}.$$

The following matching shows that the system is over-constrained. Therefore, the two remaining conditions can be used as parity conditions, as marked here.

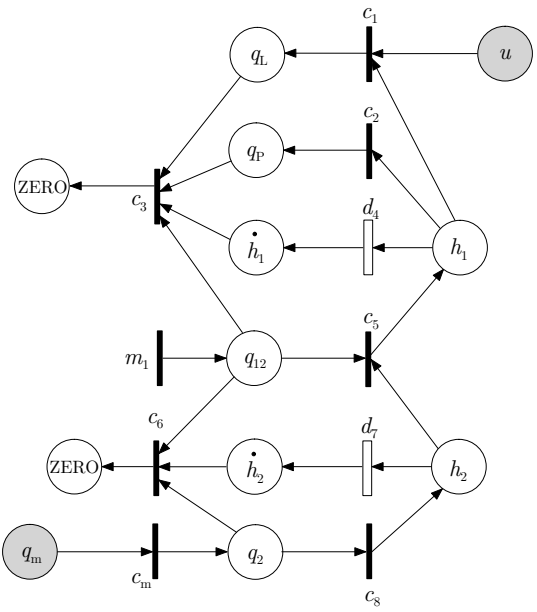


Fig. 5.26. Oriented graph, in which the arrows indicate the order of matching

$q_L = c_L\sqrt{h_1}$
$q_P = u{\cdot}f(h_1)$
$0 = -q_L + q_p - q_{12} - A\dot{h}_1$
$\dot{h}_1 = \frac{d}{dt}h_1$
$h_1 = h_2 + \left(\frac{q_{12}}{k_1}\right)^2$
$0 = q_{12} - A\dot{h}_2 - q_2$
$\dot{h}_2 = \frac{d}{dt}\,h_2$
$h_2 = \left(\frac{q_2}{k_2}\right)^2$
$q_{12} = q_{12,m}$
$q_2 = \frac{q_m}{k_m}$

↗	q_L	q_P	$\dot{h}_1$	h_1	q_{12}	$\dot{h}_2$	h_2	q_2	R
c_1	①			1					3
c_2		①		1					3
c_3	1	1	1		1				4
d_4			①	1					3
c_5				①	1		1		2
c_6					1	1		1	4
d_7						①	1		2
c_8							①	1	1
m_1					①				0
c_m								①	0

This matching results in the oriented graph shown in Fig. 5.26. Following the orientation of the edges, it is easy to see that the first parity relation depends on the variables

$$\{u,\, q_L,\, q_P,\, \dot{h}_1,\, h_1,\, q_{12},\, q_{12,m},\, h_2,\, q_2,\, q_m\}$$

only, while the second depends on

$$\{q_{12},\, q_2,\, h_2,\, \dot{h}_2,\, q_{12,m},\, q_m\}.$$

So these two conditions can be used to selectively monitor different parts of the system (Tank 1 and Tank 2, to be precise). Only a fault in the connection flow q_{12} or its measurement would affect both constraints.

By signification the following two residuals are obtained:

$$0 = u \cdot f(h_1) - q_{12,m} - A\dot{h}_1 - c_L\sqrt{h_1}$$
$$0 = q_{12,m} - A\dot{h}_2 - \frac{q_m}{k_m}$$

with

$$h_1 = h_2 + \left(\frac{q_{12,m}}{k_1}\right)^2$$
$$h_2 = \left(\frac{q_m}{k_m k_2}\right)^2. \;\square$$

The following example summarises the structural approach to fault diagnosis for the two-tank system.

Example 5.40 *Two-tank system*

The two-tank system introduced in Section 2.1 will be considered. u is the known control input and q_m the measured outflow. The following equations lead to the structure graph of the system (Fig. 5.27):

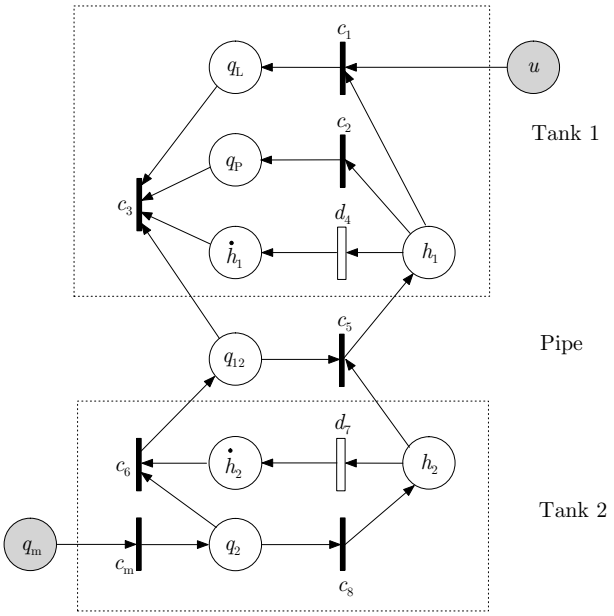


Fig. 5.27. Structure graph of the two-tank system

$$
\begin{aligned}
c_1: &\quad q_L = c_L\sqrt{h_1}\\
c_2: &\quad q_P = u \cdot f(h_1)\\
c_3: &\quad \dot{h}_1 = \frac{1}{A}\,(q_P - q_L - q_{12})\\
d_4: &\quad \dot{h}_1 = \frac{d}{dt}\,h_1\\
c_5: &\quad q_{12} = k_1\sqrt{h_1 - h_2}\\
c_6: &\quad \dot{h}_2 = \frac{1}{A}\,(q_{12} - q_2)\\
d_7: &\quad \dot{h}_2 = \frac{d}{dt}\,h_2\\
c_8: &\quad q_2 = k_2\sqrt{h_2}\\
c_m: &\quad q_m = k_m\,q_2.
\end{aligned}
$$

A, k_1, k_2 and k_m are known parameters. c_L is the unknown parameter describing the size of the fault. It can be assumed to be zero for the faultless case. In the structure graph the constraints c_1, c_2, c_3 and d_4 representing the Tank 1 are separated from constraints c_6, d_7, c_8 and c_m describing the Tank 2.

The following matching is found using the ranking algorithm, where the last column shows the rank of the constraints obtained.

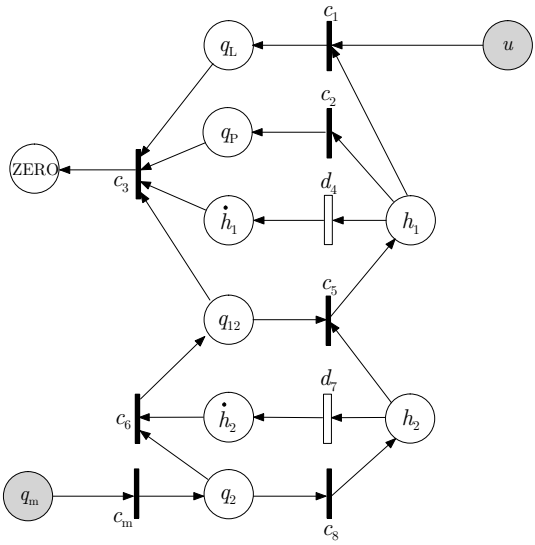


Fig. 5.28. Oriented graph of the two-tank system

$q_L = c_L\sqrt{h_1}$
$q_P = u \cdot f(h_1)$
$0 = -q_L + q_p - q_{12} - A\dot{h}_1$
$\dot{h}_1 = \frac{d}{dt}\, h_1$
$h_1 = h_2 + \left(\frac{q_{12}}{k_1}\right)^2$
$q_{12} = A\dot{h}_2 + q_2$
$\dot{h}_2 = \frac{d}{dt} h_2$
$h_2 = \left(\frac{q_2}{k_2}\right)^2$
$q_2 = \frac{q_m}{k_m}$

↗	q_L	q_P	$\dot{h}_1$	h_1	q_{12}	$\dot{h}_2$	h_2	q_2	R
c_1	①			1					5
c_2		①		1					5
c_3	1	1	1		1				6
d_4			①	1					5
c_5				①	1		1		4
c_6					①	1		1	3
d_7						①	1		2
c_8							①	1	1
c_m								①	0

The equations shown on the left are already solved for the matched variable. The corresponding oriented graph is shown in Fig. 5.28. Simplifying these equations results in the following redundancy relation:

$$0 = u(t) \cdot \boldsymbol{f}\left(h_1(t)\right) - A\dot{h}_2 + \frac{q_m(t)}{k_m} - A\dot{h}_1 - c_L\sqrt{h_1} \tag{5.37}$$

with

$$h_1(t) \quad = \quad h_2(t) + \left(\frac{A\dot{h}_2}{k_1} + \frac{q_m(t)}{k_m\, k_1}\right)^2 \tag{5.38}$$

$$h_2(t) \quad = \quad \left(\frac{q_m(t)}{k_m\, k_2}\right)^2. \tag{5.39}$$

Equations (5.37) – (5.39) can be used to monitor the two-tank system. By using Eq. (5.39), $h_2(t)$ and, hence, $\dot{h}_2$ can be determined for given measurement $q_m(t)$. Then Eq. (5.38) yields

$h_1(t)$ and $\dot{h}_1$. Finally, Eq. (5.37) is checked for known $u(t)$, $q_m(t)$ and for $h_1(t)$, $\dot{h}_1$ and $\dot{h}_2$ just obtained.

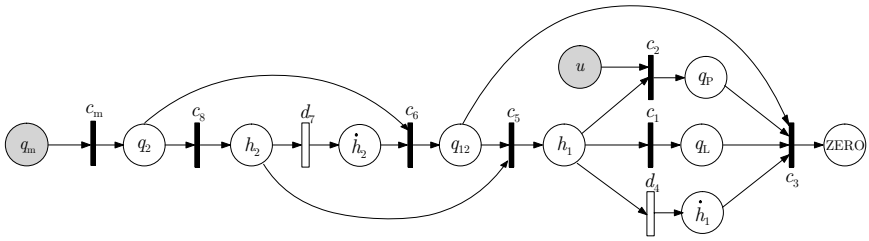


Fig. 5.29. Graph showing the order in which the unknown variables can be determined for given q_m

After redrawing the structure graph, Fig. 5.29 is obtained. This graph shows in which order the constraints can be used to determine all internal variables for given measurement q_m. Finally, constraint c_3 is used to test the consistency of the variables with the model.

A simulation result is shown in Fig. 5.30 which shows from top to bottom the signals $u(t)$, $x_1(t)$ and $x_2(t)$, the measurement $q_m(t)$ and the right-hand side of Eq. (5.37). Note that the states are reconstructed very nicely. The residual (which is in fact the loss of water through the leak) shows the occurrence of the fault very precisely and without any delay. The little spike at time 155 s is due to the reversal of the flow direction in the connection pipe, which represents a singular point in the linearised system.

Of course, the parity relation is very sensitive to measurement noise due to the two differentiations. Even a very small noise can disturb the fault detection scheme heavily as shown in Fig. 5.31. Although the noise is not visible in the measured signal q_m, its differentiation is large and calls for the use of statistical decision making algorithms in order to make fault detection possible.

Alternatively, if a filter is included when determining $\dot{h}_1$ and $\dot{h}_2$ from h_1 or h_2, respectively, the residuals increase slightly delayed after the fault has occurred. Due to the phase lag introduced by the filter, the residual given by Eq. (5.37) is no longer zero for the faultless case. Using digital filters without phase lag more precise results are possible, but these filters delay the calculation of the residual further. □

5.7 Controllability

Controllability is a property which only concerns that part of the model which describes the links between the unknown variables and the input variables, independently of the fact that some unknown variables might be measured or not. Thus, it can be analysed from the system bi-partite graph in which the measurement constraints have been removed. Roughly speaking, controllability is concerned with the possibility of finding controls (this explains why the input variables will be considered as unknowns to be computed) so as to achieve objectives, which are defined in terms of the values one wishes the system variables to be given.

The reachable set of a system (the meaning of reachability is here of course different from the meaning it has in graph theory) is the set of states such that there

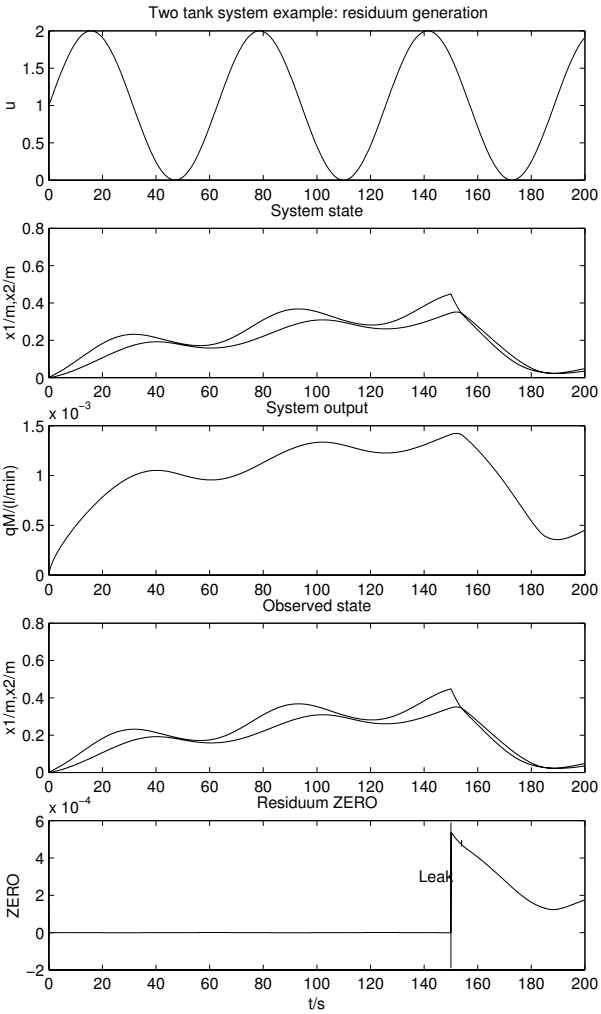


Fig. 5.30. Simulation results of the two-tank system. From top to bottom: input u; tank levels $h_1,\ h_2$; measured q_m; reconstructed levels $h_1,\ h_2$; right-hand side of Eq. (5.37).

exists a control which drives the system state, from some given initial value, to one of them. Global controllability is a strong property, which states that the reachable set is the whole state space. Local controllability is a weaker property, which states that any point in the open ball around a reachable point is also reachable (which means that the states which are reachable from a given state are not restricted to a r-dimensional manifold, where r is less than n, the dimension of the state space). For linear systems, local and global properties obviously coincide.

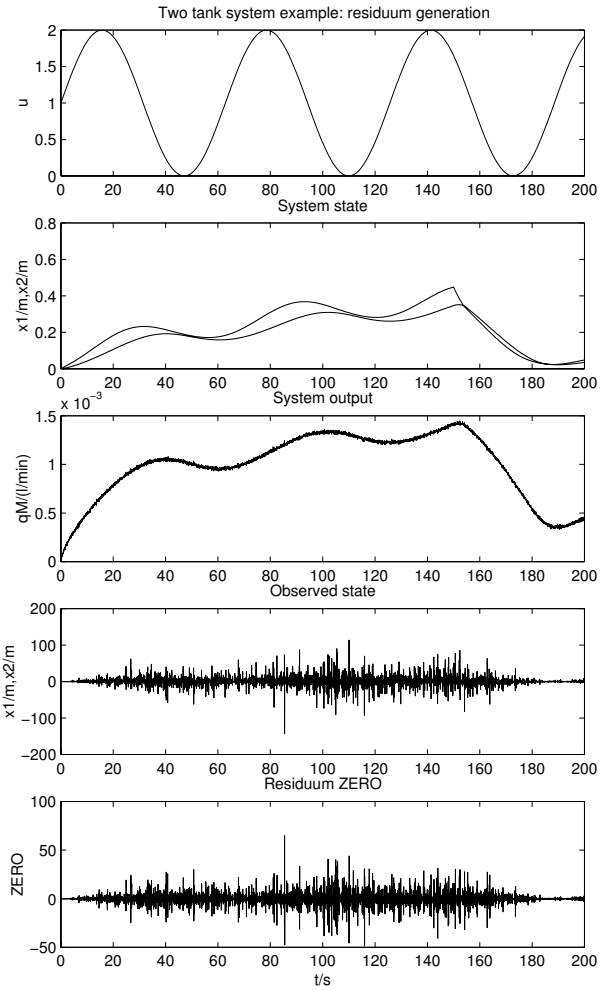


Fig. 5.31. Simulation result similar to Fig. 5.30 but with a small noise of the measured signal q_m

Let us first consider static systems $(\mathcal{C},\ \mathcal{Z})$ like

$$0 = \boldsymbol{h}\,(\boldsymbol{x}_a,\ \boldsymbol{u}), \tag{5.40}$$

where $\mathcal{C} = \boldsymbol{h}, \mathcal{Z} = \boldsymbol{x}_a \cup \boldsymbol{u}$. For such systems, global controllability means that Eq. (5.40) can be solved for the unknown variables $\boldsymbol{u}$ (to be determined) for any value of the known (wished) variables $\boldsymbol{x}_a$, thus justifying the decomposition of $\mathcal{Z}$ into $\mathcal{Z} = \mathcal{K} \cup \mathcal{X}$, where $\mathcal{K} = \boldsymbol{x}_a$, $\mathcal{X} = \boldsymbol{u}$.

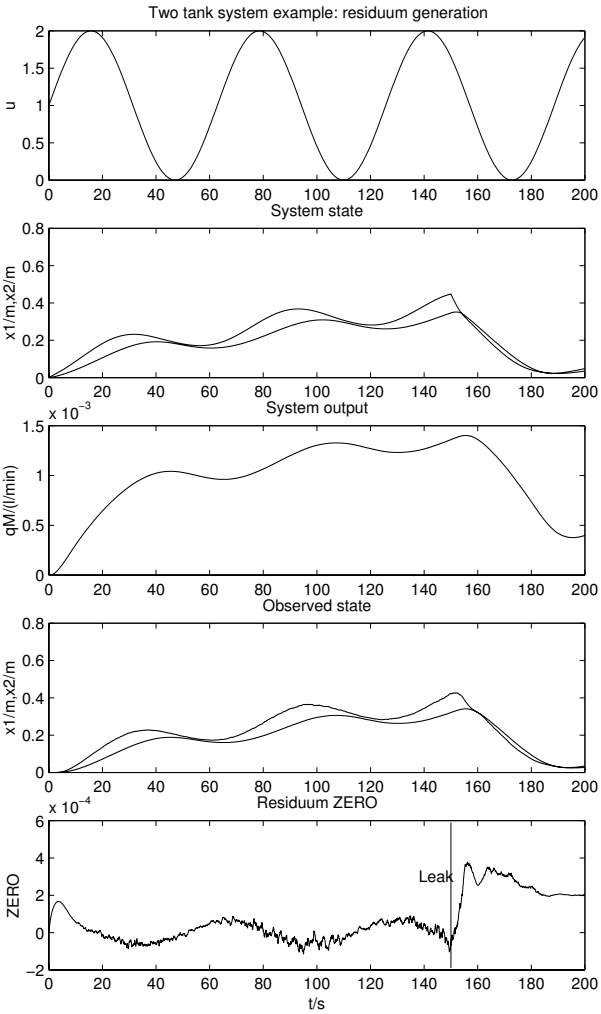


Fig. 5.32. Simulation result similar to Fig. 5.31 but with filtered measurement; q_M is the filtered system output.

Theorem 5.5 (Controllability of static systems)
A necessary and sufficient condition for system (5.40) to be structurally controllable is:

(i) $\mathcal{K}$ is reachable from the input,
(ii) the canonical decomposition of $(\mathcal{C}_\mathcal{X},\ \mathcal{X},\ \mathcal{E}_\mathcal{X})$ contains no over-constrained subsystem.

If $\mathcal{K}$ were not reachable from the input, there would be a decomposition of $\boldsymbol{x}_a$ into $\boldsymbol{x}'_a$ (the reachable part), and $\boldsymbol{x}''_a$ (the unreachable part), such that

$$\begin{aligned} 0 &= h'(\boldsymbol{x}'_a, \boldsymbol{u}) \\ 0 &= h''(\boldsymbol{x}''_a) \end{aligned}$$

and obviously there is no solution to the above system for *any* $\boldsymbol{x}_a$, namely when $\boldsymbol{x}_a$ is such that the part $\boldsymbol{x}''_a$ does not satisfy the second equation. On another hand, if the canonical decomposition did contain an over-constrained subsystem, the known variables would satisfy some compatibility condition, which would result in the existence of some manifold

$$\alpha(\boldsymbol{x}_a) = 0$$

and in the impossibility to find any control $\boldsymbol{u}$ when the wished system states lie out of this manifold.

The case of dynamical systems is more complex, and except for linear systems, only the reachability condition of the above result can be extended. Consider the general system

$$\dot{\boldsymbol{x}}_d = \boldsymbol{g}(\boldsymbol{x}_d, \boldsymbol{x}_a, \boldsymbol{u}, t) \tag{5.41}$$

$$\mathbf{0} = \boldsymbol{m}(\boldsymbol{x}_d, \boldsymbol{x}_a, \boldsymbol{u}, t) \tag{5.42}$$

$$\dot{\boldsymbol{x}}_d = \frac{d}{dt}\boldsymbol{x}_d, \tag{5.43}$$

where the known variables are $\mathcal{K} = \boldsymbol{x}_a \cup \dot{\boldsymbol{x}}_d$, the unknown variables are $\mathcal{X} = \boldsymbol{x}_d \cup \boldsymbol{u}$ and the constraints are $\mathcal{C} = \boldsymbol{g} \cup \boldsymbol{m} \cup \frac{d}{dt}$.

Note that since the initial conditions $\boldsymbol{x}_d(0)$ are known, derivative as well as integral causality can be used.

Theorem 5.6 (Reachability condition)
A necessary condition for system (5.41) – (5.43) to be structurally controllable is that $\mathcal{K}$ can be reached from the input $\boldsymbol{u}$.

This condition expresses that there does not exist any subsystem whose dynamical behaviour is independent of the input. The "no over-constrained subsystem" condition cannot be extended to the general case, but it holds for linear systems. For simplicity, let us drop algebraic equations, and consider the system (5.44), (5.45) with the known variables $\mathcal{K} = \dot{\boldsymbol{x}}_d$, the unknown variables $\mathcal{X} = \boldsymbol{x}_d \cup \boldsymbol{u}$, and the constraints $\mathcal{C} = \boldsymbol{g} \cup \frac{d}{dt}$.

$$\dot{\boldsymbol{x}}_d = \boldsymbol{g}(\boldsymbol{x}_d, \boldsymbol{u}, t) \tag{5.44}$$

$$\dot{\boldsymbol{x}}_d = \frac{d}{dt}\boldsymbol{x}_d \tag{5.45}$$

Theorem 5.7 (Linear continuous systems)
If the constraints $\boldsymbol{g}$ are linear, a necessary and sufficient condition for system (5.44), (5.45) to be structurally controllable is that

(i) $\mathcal{K}$ is reachable from the input,
(ii) the canonical decomposition of $(\mathcal{C}_{\mathcal{X}}, \mathcal{X}, \mathcal{E}_{\mathcal{X}})$ contains no over-constrained subsystem.

The existence of an over-constrained subsystem would imply that the known variables (here $\dot{\boldsymbol{x}}_d$) satisfy some compatibility conditions. For linear systems, these would be expressed as

$$\boldsymbol{\alpha}' \dot{\boldsymbol{x}}_d = 0, \tag{5.46}$$

where $\boldsymbol{\alpha}$ is some constant vector, from which it follows that any system trajectory would belong to the manifold

$$\boldsymbol{\alpha}' \boldsymbol{x}_d(t) - \boldsymbol{\alpha}' \boldsymbol{x}_d(t_0) = 0$$

and therefore that it is not possible to drive the system state to any point in the state space.

Condition (ii) does not extend to nonlinear systems, because in order to define a manifold the compatibility constraints (5.46) which would now be non-linear should also be integrable. This property does not follow from structural considerations.

5.8 Structural analysis of fault tolerance

A system is said to be fault tolerant if the system objectives can be achieved, in spite of the faults. Objectives can be classified into two classes, associated either with the control, or with the estimation, of a given subset of the system variables. Objectives can be achieved, in spite of the faults, if at least one of two procedures, namely *fault accommodation* and *system reconfiguration* can be used successfully. Fault accommodation means to control or estimate the variables of interest in *the faulty system* (which means under the constraints associated with the faulty components). System reconfiguration means to switch off faulty components (and eventually switch on healthy ones, which were off before the fault occurs), so that controlling or observing *the resulting new system* can achieve the control or estimation objectives under the set of constraints associated with the reconfigured system.

Whatever the specific form of the control or estimation problem (deterministic, stochastic, with uncertainties, etc), control objectives obviously need the system variables of interest to be controllable, while estimation objectives need them to be observable. Therefore, as far as fault tolerance is concerned, it is of interest to analyse those system structural properties, not only in the normal, but also in faulty operation modes.

5.8.1 Faults and the system structure

Faulty operation modes are modes in which some component(s) being faulty, one or several of the system constraints are not the nominal ones. Let $(\mathcal{C}_{\rm nom}, \mathcal{Z}_{\rm nom}, \mathcal{E}_{\rm nom})$ (respectively $(\mathcal{C}_{\rm fault}, \mathcal{Z}_{\rm fault}, \mathcal{E}_{\rm fault})$) be the bi-partite graphs associated with the nominal (respectively the faulty) system. The changes in the constraints, which follow the occurrence of a fault, may be classified into changes which affect the structure of the system (*structural faults*) and changes which do not affect the structure of the system (*non-structural faults*).

- **Non-structural faults** are such that

  $$(\mathcal{C}_{\rm nom}, \mathcal{Z}_{\rm nom}, \mathcal{E}_{\rm nom}) = (\mathcal{C}_{\rm fault}, \mathcal{Z}_{\rm fault}, \mathcal{E}_{\rm fault})\,.$$

  Therefore, the only changes caused by the fault are in the mathematical expressions of the constraints $\mathcal{C}_{\rm nom}$. Parametric faults, in which only the values of the parameters are changed, are a very common example of non-structural faults. For example, consider a resistor, whose resistance changes from the nominal value R_n to another one, $0 < R_f < \infty$, following the short circuit of some subset of electric turns. The occurrence of the fault just changes the constraint $u - R_n i = 0$ into the constraint $u - R_f i = 0$.

- **Structural faults** change the set of the constraints and variables which are to be considered. For example, fault effects might not only concern parameters, or they might be associated with special values of the parameters, so that the sets $\mathcal{C}_{\rm nom}$, $\mathcal{Z}_{\rm nom}$, $\mathcal{E}_{\rm nom}$ are changed. For the above resistor, examples of such faults are $R_f = 0$ (the result of a complete short-circuit), and $R_f = \infty$ (the result of the resistor blow-off due to over heating). These two internal faults change the constraint $u - R_n i = 0$ respectively into $u = 0$ (which does not constrain i) and into $i = 0$ (which does not constrain u). Another example is that of an external fault, caused by operating the resistor not in the nominal temperature range, in which the resistance value can no longer be considered as constant, but depends on the temperature θ, thus changing the constraint into $u - R(\theta) i = 0$ with $R_f,\, R_n \neq 0$.
  In many cases, the set of the variables which are linked by the constraints does not change, but the nature of the relationship is modified, resulting in a change of the possible matchings. In other cases, the new constraints introduce or suppress system variables.

Example 5.41 *On-off valve*

Consider the on-off input valve of the single-tank system, whose normal operation is described by

$$c_2 : \begin{cases} u(t) = 1 \Longrightarrow q_i(t) = \alpha \\ u(t) = 0 \Longrightarrow q_i(t) = 0. \end{cases}$$

Suppose the valve is blocked open, then the constraint it introduces becomes

$$c_2 : \begin{cases} u(t) = 1 \Longrightarrow q_i(t) = \alpha \\ u(t) = 0 \Longrightarrow q_i(t) = \alpha. \end{cases}$$

Obviously, the structural graph remains the same, but the nature of the relationship between u and q_i is changed.

Let us first suppose that both u and q_i are unknown variables: the set of the possible matchings is changed. In the nominal situation, both u and q_i can be matched with constraint c_2 while in the faulty situation, the only possible matching is $(c_2,\ q_i)$.

Let us now suppose (which is the usual situation) that the control u is known and that q_i is unknown. The only possible matching $(c_2,\ q_i)$ does not represent any real dependency, since the value of q_i is no longer a consequence of the value of the control u, but it is forced to α. This can be modelled by the fact that when the fault is present, the node c_2 is removed from the graph and consequently, the edges $(u,\ c_2)$ and $(q_i,\ c_2)$ disappear. The nodes u and q_i which remain become nodes with known values, and analysing the change in the observability and controllability properties of the system reduces to the analysis of the consequence of q_i changing its status from unknown to known (but constant).

Suppose now that the valve still closes perfectly, but there is a leak when it is open. The constraint becomes

$$c_2 : \begin{cases} u(t) = 1 \Longrightarrow q_i(t) + q_l(t) = \alpha \\ u(t) = 0 \Longrightarrow q_i(t) = 0, \end{cases}$$

where $q_l(t)$ is the (unknown) leakage flow. The new constraint introduces the extra variable $q_l(t)$ thus changing the structure of the system. It may be investigated whether it is possible to build a model such that the structural graph remains the same in the presence of such faults, by considering the fault model at the beginning. In this case, the normal operation could be described by $q_l(t)$ being a known variable (thus introducing the constraint c_2')

$$\begin{aligned} c_2 \quad &: \quad \begin{cases} u(t) = 1 \Longrightarrow q_i(t) + q_l(t) = \alpha \\ u(t) = 0 \Longrightarrow q_i(t) = 0 \end{cases} \\ c_2' \quad &: \quad q_l(t) = 0, \end{aligned}$$

while the faulty operation would be modelled by removing node c_2', thus again changing the graph and the status of $q_l(t)$ from known to unknown. □

5.8.2 Knowledge about faults

Changes which result from faults may also be classified into known and unknown ones. This depends on the performance of the fault diagnosis algorithms.

The complete knowledge of changes (whether they affect or not the structure of the system) requires fault detection (indicates that fault has occurred), fault isolation (indicates which constraints have been changed), and fault estimation (indicates the new mathematical expression of the constraints, or the new values of the parameters in the changed constraints). Note that in some cases, it may happen that only a subset of values is proposed by the fault diagnosis algorithm for the new parameters (thus providing an uncertain model of the faulty system).

However, in most cases, only fault detection and fault isolation are available. Then changes are only partially known, namely the subsets of changed and unchanged constraints are known, but their model is not available.

According to the class of the faults, and to the performances of the fault diagnosis algorithms, three levels can be distinguished for the analysis of the changes in the system structural analysis, namely the no structural change, known stuctural changes, and unknown structural changes levels.

5.8.3 Fault tolerance with respect to non-structural faults

From their definition, non-structural faults do not change the system structural graph. Thus, no specific structural analysis problem has to be stated, since structural observability and controllability properties are the same in the faulty and in the healthy system. This results from structural analysis conclusions being independent of the mathematical expressions of the constraints (as long as the underlying assumptions are satisfied) and of the values of the parameters.

When the model of the faulty system is known (which means that the fault diagnosis procedure provides also the estimation of the new system constraints and parameters), the fault accommodation problem can be set, and fault accommodation can be performed, if this problem has a solution (remember that even when structural properties are true, they do not guarantee actual properties to hold). When only uncertain values can be assigned to the new parameters, a robust fault accommodation problem can be set, and robust fault accommodation can be performed if this problem has a solution. Finally, when no value can be provided for the new parameters, it is just known that the system remains structurally controllable (observable), but the fault-tolerant control cannot be achieved using the accommodation strategy, and reconfiguration has to be investigated.

5.8.4 Fault tolerance with respect to structural faults

Structural faults change the system structural graph. As seen above, changes to be considered are: changing the status of variables from known to unknown (or from unknown to known), and suppressing some constraint nodes while suppressing or not the associated variables. Two cases are to be distinguished, according to whether the fault diagnosis algorithm provides or does not provide the knowledge of the changes.

- When the model of the faulty system is known, it is possible to set an accommodation problem, which may have solutions. Structural observability and controllability properties are necessary for such a solution to exist, and they can be analysed from the (known) structural graph of the faulty system.
- When the model of the faulty system is not known (the changes in the constraints can only be detected and isolated), fault accommodation is not possible, and only system reconfiguration can be used to provide the design engineer with a known system model, which results from switching off the faulty components (and maybe also switching on some components which were previously

off). Structural observability and controllability properties, can then be analysed from the (known) graph of the reconfigured system.

In both cases, structural observability and controllability are analysed from the changed structural graph (the change results from the faults in the accommodation strategy, while it results from switching off and on some components in the reconfiguration strategy). Remember that a structurally observable (controllable) variable x is the target of a subgraph (a set of alternated chains) with specific properties (e.g. causal with respect to derivative causality) which links this variables and the set of the known ones (respectively the set of the input signals). Therefore, the analysis of fault tolerance is concerned with answering the question whether such subgraphs do or do not disappear, when nodes are suppressed, or when the status of the variables is changed from known to unknown status or vice versa.

The question has to be answered on-line, as soon as the structure graph of the faulty system becomes available from fault diagnosis, when fault accommodation is used. It can be answered off-line, beforehand, for a given set of faults, when reconfiguration is envisaged. In that case, the problem is to decide about the *reconfigurability* of the control and estimation scheme in the presence of given faults, i.e. to decide whether some structurally observable (controllable) variable x still remains structurally observable (controllable) when the faulty components are switched off, or in other words, removed from the system (which means that some vertices – and the associated edges – are removed from the structure graph).

This is obviously true as long as there remains at least one estimation (control) subgraph whose target is x, when those vertices are removed. Equivalently, this means that several estimation (control) subgraphs with target x were existing in the nominal bi-partite graph, and that at least one of them does not contain any constraint which belongs to the faulty component. It follows that it is possible to evaluate the extent to which a system is fault tolerant by reconfiguration, by counting the number of such different observation (control) possibilities.

Definition 5.9 (Redundancy degrees)
The estimation redundancy degree of an observable variable x with respect to a given fault φ is the number of estimation subgraphs whose target is x and which are not affected by φ. Similarly, the control redundancy degree of a controllable variable x with respect to a given fault is the number of control subgraphs whose target is x and which are not affected by that fault.

This definition can obviously be extended to the consideration of a given subset of faults, and its interpretation is straightforward. Let $\delta(x, \Phi)$ be one of the redundancy degrees (estimation or control) of x with respect to $\Phi \subseteq C$. It characterises the number of system configurations which still allow to achieve the objective of estimating and controlling the unknown variable x through reconfiguration when the subset of constraints Φ is faulty. For any pair (x, Φ) it can be easily determined from structural analysis and it can be used as a measure of fault tolerance (reconfigurability) of the estimation or control problem.

Definition 5.10 (Critical sets) *For each given variable x to be estimated or controlled, a critical fault set $\bar{\Phi}(x)$ is the minimal set of constraints such that x is no longer observable or controllable when they are all faulty:*

$$\bar{\Phi}(x) = \left\{\bar{\Phi} \subseteq \mathcal{C}, \delta\left(x, \bar{\Phi}\right) = 0\right\}.$$

There may be several critical fault sets for a given variable. The one with the least cardinality and the one with the largest cardinality are respectively associated with the so-called "strong" and "weak" redundancy degrees.

Example 5.42 *Two-tank reconfiguration*

Consider the two-tank system whose oriented graph is given by Fig. 5.28. Suppose that in addition to the single sensor q_m (which provides the redundancy relation given by Fig. 5.29) we use sensors which measure the levels h_{1m} and h_{2m} in the two tanks

$$\begin{aligned} c_{h1}: &\qquad h_1 = h_{1m} \\ c_{h2}: &\qquad h_2 = h_{2m}. \end{aligned}$$

The upper part of Fig. 5.33 shows the alternated chains associated with the same matching, with obvious extension to the new known variables h_{1m} and h_{2m} that are known and, hence, drawn with dashed lines.

Assume some fault on sensor q_m, which is unknown, and therefore cannot be compensated (fault accommodation is not possible). The system can nevertheless still be operated provided reconfiguration is possible. This is indeed the case, as shown by the new alternated chain

$$h_{2m} - c_{h2} - h_2 - c_8 - q_2$$

by which q_2 can now be computed (and therefore it remains observable, as well as all the other system variables). Note that using this new structure, one redundancy relation is lost, namely the relation associated with the alternated chain

$$h_{2m} - c_{h2} - h_2 - c_8 - q_2 - c_m - q_m.$$

The new situation is shown in the lower part of the figure. □

Example 5.43 *Redundancy degrees and critical sets*

Consider a system with five sensors y_i, $i = 1, ..., 5$, and assume we are interested in the estimation of the unknown variable x_1. Let the structural graph be given by its incidence matrix

	x_1	x_2	x_3	x_4	y_1	y_2	y_3	y_4	y_5
c_1	1	1	1						
c_2	1			1		1			
c_3	1			1					1
c_4		1			1				
c_5			1			1	1		
c_6				1		1		1	
c_7				1				1	

Taking x_1 as the target one obtains the graph given by Fig. 5.34.

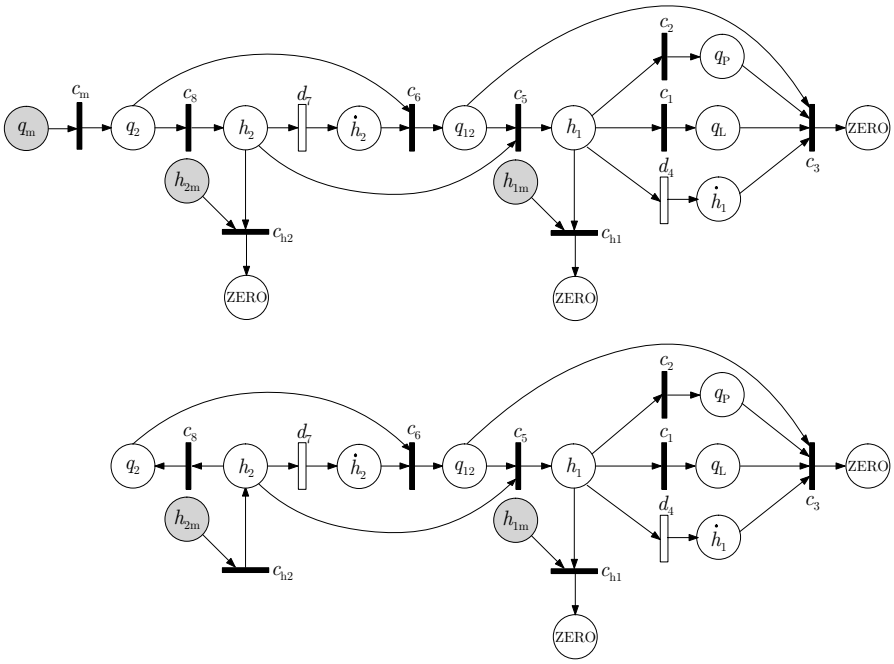


Fig. 5.33. Reconfigurability of the two-tank system (with additional sensors)

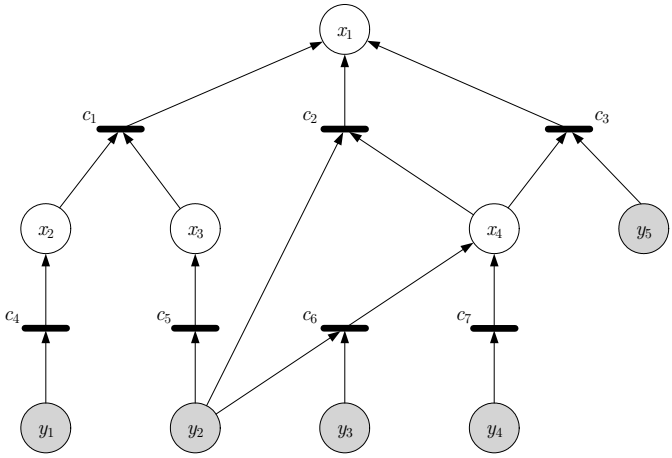


Fig. 5.34. Structure graph with target x_1

There are three different ways to compute the estimation of x_1, either by using constraint c_1, or by c_2 or c_3. Backtracking the alternated chains one obtains a total of five possibilities (i.e. five versions of the estimation service) $v_1, ..., v_5$, whose formal representations are:

$$\begin{aligned}
v_1 : \quad & x_1 = \gamma_1(\gamma_4(y_1), \gamma_5(y_2, y_3)) \\
v_2 : \quad & x_1 = \gamma_2(y_2, \gamma_6(y_2, y_3)) \\
v_3 : \quad & x_1 = \gamma_2(y_2, \gamma_7(y_4)) \\
v_4 : \quad & x_1 = \gamma_3(\gamma_6(y_2, y_3), y_5) \\
v_5 : \quad & x_1 = \gamma_3(\gamma_7(y_4), y_5).
\end{aligned}$$

When sensor y_1 is faulty and disconnected from the system, the remaining versions are

$$\begin{aligned}
v_2 : \quad & x_1 = \gamma_2(y_2, \gamma_6(y_2, y_3)) \\
v_3 : \quad & x_1 = \gamma_2(y_2, \gamma_7(y_4)) \\
v_4 : \quad & x_1 = \gamma_3(\gamma_6(y_2, y_3), y_5) \\
v_5 : \quad & x_1 = \gamma_3(\gamma_7(y_4), y_5)
\end{aligned}$$

and therefore the estimation redundancy degree of x_1 with respect to the fault of y_1 is 4. Faults on sensor y_2 are more severe, since they leave only version v_5 operational. The sets $\{y_2, y_4\}$, $\{y_2, y_5\}$, $\{y_3, y_4\}$ are critical, since when these sensors are disconnected, there is no estimation version at all. Therefore, the strong redundancy degree is 2. □

5.9 Evaluation of structural analysis

Structural analysis is an important tool, which is of interest in the early stage of the control and supervision system design, when detailed models are not available. The fault diagnosis and fault-tolerant control results it provides are the identification of the monitorable part of the system, and the identification of the reconfiguration possibilities of the estimation (respectively the control) scheme. Since detailed behaviour models need only to be developed for those parts of the system, structural analysis is also a tool for deciding which modelling investments must be done for the design of the control and supervision system.

Observability analysis is the main step to identify the system monitorable part, which is the over-constrained subsystem within the observable one. It can be noticed that structural analysis not only provides the computation mechanisms for the estimation algorithms and their reconfiguration, but it can also suggest which sensors should be implemented so as to change the status of system components from non-monitorable to monitorable.

Structural analysis cannot help in defining fault accommodation strategies, since these strategies are aimed at investigating the means of achieving the system objectives, in spite of faults, without changing its structure. On the contrary, structural analysis is of prime importance as far as reconfiguration is concerned, since the results are expressed with reference to graph properties, whose changes can be analysed when vertices and edges disappear, as the consequence of switching off some system components, after a fault has occurred.

5.10 Exercises

Exercise 5.1 *Structural analysis for industrial actuator*
Make a structural model of the actuator shown in Fig. 5.35.

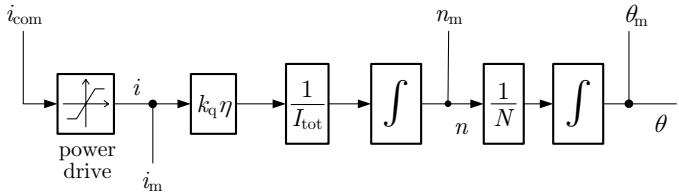


Fig. 5.35. Position actuator open loop

1. Determine the sets K (known variables), X (unknown variables) and Z (all variables).
2. List the set of constraints that describe the system shown in Fig. 5.35.
3. Write the incidence matrix and draw the structure graph.
4. Ignore causality and determine a complete matching on X that is non-causal.
5. Use the ranking algorithm to determine a complete causal matching on X. List the unmatched constraints.
6. Determine the parity relations found from the unmatched constraints by backtracking the structure graph to known variables along the paths of the matching,
$$c_i(K_i) = 0 \wedge \ K_i \subseteq K.$$
7. Express the parity relations in analytical form using the constraints from question 2. □

Exercise 5.2 *Structural analysis with unknown input*
Consider the speed control closed in Fig. 5.36 where n_{ref} is the reference speed.

1. Using the known variables
$$K = \{i_m, n_m, \theta_m, n_{\mathrm{ref}}\}$$
and the unknown variables
$$X = \left\{i, Q_l, n, \dot{n}, \theta, \dot{\theta}\right\},$$
determine the set of constraints that describe the system.
2. Make the structure graph for the system. Describe the graph as an incidence matrix and draw the structure graph.
3. Apply the ranking algorithm on the graph to determine at least one causal matching. List which constraints were unmatched.
4. For each unmatched constraint, determine a parity relation $c_i(K_i) = 0, \ K_i \subseteq K$. □

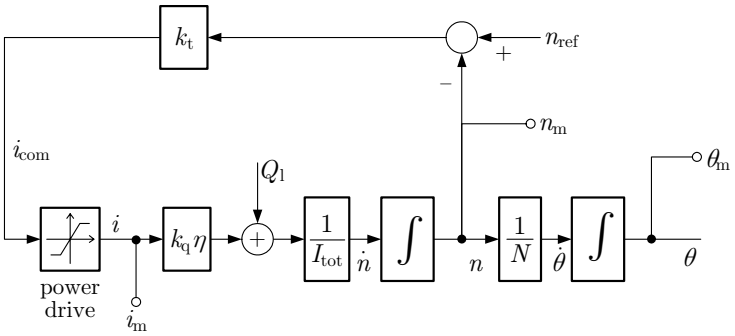


Fig. 5.36. Block diagram of DC motor with load torque and closed speed loop

Exercise 5.3 *Parity relations for single axis satellite*

This exercise considers structural analysis for a "single-axis" satellite described by the block diagram in Fig. 5.37. The figure illustrates a single axis of a satellite.

There are two input signals u_1 and u_2 to actuators 1 and 2, respectively.

There is one unknown input d.

There are five measurements: y_1 measures the state x_1, y_2 and y_3 measure x_2; y_4 and y_5 measure torque from actuators 1 and 2, respectively.

1. Determine the sets of known variables, K, and unknown variables X. Verify that the intersection $Z = K \cup X$ gives the total set of variables.
2. Determine the set of constraints that describe the system.
3. Determine the causal structure graph for the system. Make a graph as an incidence matrix and as a bi-partite graph.
4. Use the ranking algorithm on the graph to find one or more complete matchings. List which constraints were unmatched.
5. From the unmatched constraints, determine the parity relations in analytic form:

$$c_i(K_i) = 0,\ K_i \subseteq K.$$

You may wish to use the MATLAB programme *SaTool* to cope with the complexity of matching or for checking your results. A GNU open source license of *SaTool* is available from the book homepage. □

Exercise 5.4 *Parity relations and addition of a sensor*

Let a system be composed of 3 interconnected components, c_1, c_2, c_3. Each component is described by one constraint according to the system

$$c_1: \quad \dot{x}_1 - x_1 = 0 \tag{5.47}$$

$$c_2: \quad \dot{x}_1 - 2\dot{x}_2 = 0 \tag{5.48}$$

$$c_3: \quad y + 3x_1 - x_2 = 0. \tag{5.49}$$

The variables x_1, x_2 which characterise the operation of components c_1, c_2 are not measured, only the output y of component c_3 is known.

1. Draw the structure graph of the system.

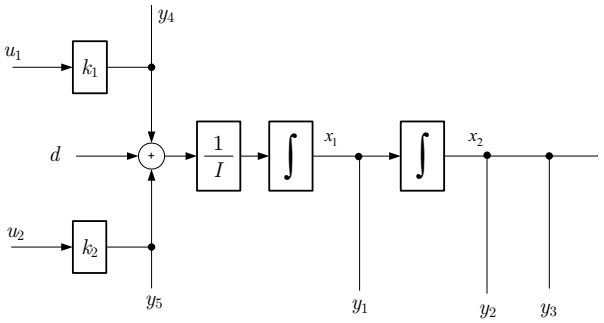


Fig. 5.37. Block diagram of single axis satellite with input from two redundant actuators, redundant measurements of attitude (angle), measurement of angular rate and measurement of delivered actuator torques

2. Find a redundancy relation which allows to detect a fault in one of the components.
3. Would it be worth to add a fourth component, that would measure x_1 according to

   $c_4 : z = x_1$

   (z is now an extra known variable, but of course component n°4 may also be faulty). □

Exercise 5.5 *A specialised arithmetic circuit*

The following specialised computation circuit is composed of 3 multipliers M_1, M_2 and M_3 and two adders A_1 and A_2.

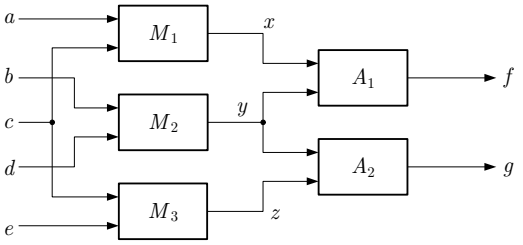


Fig. 5.38. Specialised computation circuit

1. Write the model of each system component.
2. Give the incidence matrix of the structure graph (distinguish the known and the unknown variables).
3. Find the analytical redundacy relations by eliminating the unknown variables.
4. For each ARR, give the list of the components the faults of which it is sensitive to.
5. Is there any non-detectable or non-isolable fault?

6. What are the possible diagnostics associated with the following measurements?.

a	b	c	d	e	f	g
2	2	3	3	2	10	12

□ (5.50)

5.11 Bibliographical notes

Structural concepts and bi-partite graphs have been introduced very early and have been the subject of extensive research in the applied mathematics community [50], [51]. Maximal matchings have been extensively studied because of the numerous applications they allow to solve. An algorithm of complexity $O(\mathcal{N}^3)$ for finding maximal matchings has been proposed by [52], while [95] have applied an algorithm of complexity $O(\mathcal{N}^{2.5})$ to bi-partite graphs. Maximal matchings can also be found from the solutions to the assignment problem [120], or from the maximal flow problem [58], [59]. For details on the algorithms and more bibliographical notes, refer to [12], [41], [83].

Structural concepts have been used in the 60 and 70's for the decomposition of large systems of equations in view of their hierarchical resolution [89], [241], and for the analysis of system structural properties, like observability and controllability, where most works use a digraph representation and address linear systems [82], [126], [127], [172]. They have also been extended to the design of multivariable control systems, including considerations like disturbance rejection for example [207], [214], with numerous applications in chemical engineering, [77], [130], [171] since complex large scale systems are often encountered in that field. An important issue in that field is the solvability of large scale differential and algebraic equations systems, whose structural analysis is adressed in [123], [258].

In the field of fault diagnosis, structural concepts have been used at the beginning of the 90's, for the analysis of system monitorability [47], [234] and for the design of structured residuals [80], which provide straightforward decision procedures for fault isolation [45]. A good description can be found in [233], while a significant application is described in [102]. The Artificial Intelligence approach of causality in device behaviour [101], which is used in the theory of model-based diagnosis, is also very close to the concept of matching in bi-partite graphs. Since the obtained models are mainly under a graphic form, Bond-graphists have developed many specific tools for structural analysis. Applications to fault diagnosis can be found in [247].

Structural analysis was also found useful to cope with the complexity of analysis in cases of multiple faults [19] which also reported on a marine application of the method. An extension of the structural analysis to advise on possibilities of active isolation was suggested in [24].

Finally structural concepts have recently been applied to the problem of sensor selection [37], [169], and in fault-tolerant control, for the analysis of system reconfigurability [72], [232].

Chapter 6

Fault diagnosis of continuous-variable systems

This chapter provides solutions to the fault detection, isolation and estimation problems when the model of the supervised process is either a deterministic or a stochastic continuous-variable system. The chapter considers faults that can be modelled as additive signals acting on the process. The solution of these problems leads to a diagnostic system which is separated in two parts: a residual generation module and a residual evaluation module. Particular attention is paid to the link between these two parts when using stochastic models.

6.1 Introduction

Continuous-variable models (or analytical models) consist of sets of differential or difference equations. They can be deduced by application of the laws of physics, chemistry etc. to the supervised and/or controlled process. The external variables entering these equations are called inputs. One distinguishes control inputs, which are known and can be manipulated, from disturbances which cannot be manipulated. The disturbances that are not measured are called unknown inputs. Besides, imperfections in the model and measurement noise may be represented by stochastic processes (or sequences) appearing as additional inputs. When such random input is used, one speaks about stochastic models, as opposed to deterministic models. In this chapter, the design of fault detection, isolation and/or estimation systems for processes described by deterministic or stochastic continuous-variable models with unknown input will be solved. Such systems are made of two parts as already indicated in Chapter 1: a residual generation module and a residual evaluation module (or decision system) (Fig. 6.1).

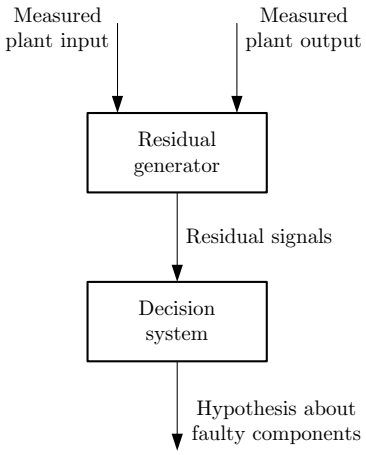


Fig. 6.1. Structure of a fault diagnosis system

The residuals are signals that, in the absence of faults, deviate from zero only due to modelling uncertainties, with nominal value being zero, or close to zero under actual working conditions. If a fault should occur, the residuals deviate from zero with a magnitude such that the new condition can be distinguished from the fault free working mode. The role of the decision system is to determine whether the residuals differ significantly from zero and, from the pattern of zero and non-zero residuals, to decide which are the most likely fault effects, and in turn, which component(s) could be the origin of a fault. When the diagnostic system is used in a fault-tolerant controller, as described in Chapters 1 and 7, details in the diagnostic task will depend on the type of faulty device and on the way the faulty condition could be treated.

Sensor faults can often be handled through estimating the faulty output signal using an estimator based on other available measurements less the one isolated as faulty. Observability of the reduced system is naturally required in this case. For this type of sensor fault, the diagnostic system needs only to perform fault detection and isolation to determine which measured signals should be disregarded. For an actuator fault which does not cause a complete loss of command, a remedial action could be to modify the control signal to the set of actuators by an increment computed in such a way that the fault is compensated. In this case, an estimate of the fault signal is needed.

The fundamental notion on which residual generation for continuous-variable systems rests is analytical redundancy. Analytical redundancy relations are equations that are deduced from an analytical model, which solely use measured variables as input. Analytical redundancy relations must be consistent in the absence of a fault, and can thus be used for residual generation. A simple example is given to introduce this notion, before considering more formal developments in subsequent sections.

Example 6.1 *Residuals for the ship autopilot*

Consider the following part of the ship autopilot example (see Section 2.2). The turn rate ω_3 and the heading angle ψ are related through

$$\dot{\psi}(t) = \omega_3(t). \tag{6.1}$$

Let us neglect the effect of waves and assume that the measurements can only be affected by a bias. Hence sensor faults are represented by additive signals and the measurement equations can be written:

$$\psi_m(t) = \psi(t) + f_\psi(t) \tag{6.2}$$
$$\omega_{3m}(t) = \omega_3(t) + f_\omega(t) \tag{6.3}$$

where the index m denotes measured quantities, and $f_\psi(t)$, $f_\omega(t)$ are the potential biases. Since most supervision systems are implemented as a software, only sampled data are available. They are linked through the following discrete model deduced from (6.1):

$$\psi(k+1) = \psi(k) + \omega_3(k)T_s, \tag{6.4}$$

where T_s stands for sampling period. By considering the equation error, r, resulting from (6.4) when the variables are substituted by their measured value, the following expression is obtained:

$$r(k) = \psi_m(k) - \psi_m(k-1) - \omega_{3m}(k-1)T_s. \tag{6.5}$$

This quantity has the properties expected for a residual. Indeed, introducing (6.2), (6.3) into (6.5) yields

$$r(k) = f_\psi(k) - f_\psi(k-1) - f_\omega(k-1)T_s.$$

This shows that, in the absence of a fault (namely when $f_\psi(k) = f_\psi(k-1) = f_\omega(k-1) = 0$), $r(k)$ is zero. Upon occurrence of a bias in the measurement of ω_3 say at time k_0, $r(k)$ takes a constant non-zero value for all $k \geq k_0$. Finally the appearance of a bias on the measurement of ψ at time instant k_0 shows up as a spike at time k_0, but has no permanent effect on r. Both faults thus affect r and this signal is zero in the absence of fault. Hence it can be named a residual signal. For decision making, it suffices to compare the residual to a specified threshold. The latter should be chosen in such a way that biases that appear to be significant for the considered application are detected.

When measurement noise is significant, comparison to a simple threshold might not be practicable, because the change in the mean of the residual due to the fault can be hidden by the effect of the noise on the residual. This noise needs to be taken into account as described in the following two discretised "noisy" versions of (6.2), (6.3):

$$\psi_m(k) = \psi(k) + f_\psi(k) + v_\psi(k) \tag{6.6}$$
$$\omega_{3m}(k) = \omega_3(k) + f_\omega(k) + v_\omega(k), \tag{6.7}$$

where $v_\psi(i), v_\omega(i), i = 1, 2, \ldots$ are mutually uncorrelated white noise sequences with variance $E(v_\psi^2(k)) = Q_\psi$ and $E(v_\omega^2(k)) = Q_{\omega_3}$ respectively.

Substituting (6.6) and (6.7) into (6.5) yields:

$$r(k) = f_\psi(k) - f_\psi(k-1) - f_\omega(k-1)T_s + v_\psi(k) - v_\psi(k-1) - v_\omega(k-1)T_s.$$

$(r(1), \ldots, r(k))$ is now a random sequence which must be evaluated by suitable algorithms. Only its mean value is equal to zero in the absence of fault. □

The difference of treatment between deterministic and stochastic models is reflected in the organisation of the chapter: Sections 6.2 to 6.5 deal with the first class of models and 6.7 to 6.8, with the second one. Analytical redundancy relations

(ARR) based on a deterministic model were already addressed in the previous chapter. Structural models were used for their determination. The link with this chapter is the object of Section 6.2 were the principle of the determination of ARR from a deterministic nonlinear state-space model is presented. Next, the particular case of deterministic linear state-space model is considered in Section 6.3, and a complete algorithm is provided for the design of parity relations (a specific type of analytical redundancy relations). A more formal presentation of parity relations for fault detection, isolation and estimation is then presented in Section 6.4 from a linear input-output model of the supervised process. The method of Sections 6.2 to 6.4 assures perfect insensitivity (or decoupling) of the residuals to an unknown input. This can only be achieved when the number of unknown input signals is lower than the number of measured output signals. When this condition does not hold, approximate decoupling of the residual with respect to the unknown input can be obtained by an optimisation approach. This is the objective of Section 6.5. A presentation of algorithms aimed at detecting changes in the mean of a stochastic random sequence is given in Section 6.7. These tools are used as parts of the systems for fault detection, fault isolation and fault estimation based on stochastic models that are described in Section 6.8.

6.2 Analytical redundancy in nonlinear deterministic systems

6.2.1 Logical background

Analytical redundancy can be seen as a tool for obtaining conditions, based on available measurements, that are necessarily fulfilled when the supervised system works in a specific operating mode. In order to illustrate the principle of analytical redundancy, consider deterministic systems described in normal operation by state and measurement equations

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta, t) \tag{6.8}$$

$$\boldsymbol{y}(t) = \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta, t), \tag{6.9}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the state vector, which is not available, $\boldsymbol{u} \in \mathbb{R}^m$ is the control input vector, $\boldsymbol{d} \in \mathbb{R}^{n_d}$ is an uncontrolled deterministic vector (disturbance). θ is a parameter vector which is considered to be known, and $\boldsymbol{y} \in \mathbb{R}^p$ is the measurement vector. Let $\mathcal{H}_0$ be the situation corresponding to normal operation, and $\mathcal{H}_1 = \neg\mathcal{H}_0$ some faulty situation. The following logical statements are true

$$\mathcal{H}_0 \iff [\dot{\boldsymbol{x}}(t) = \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta, t)] \wedge [\boldsymbol{y}(t) = \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta, t)]$$

$$\mathcal{H}_1 \iff [\dot{\boldsymbol{x}}(t) \neq \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta(t))] \vee [\boldsymbol{y}(t) \neq \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta(t))] .$$

The violation of equality constraints that results from faults may be described in two ways:

- In the first option, faults are assumed to result from parametric variations, which is represented as

$$\theta_f(t) \neq \theta \iff \theta_f(t) = \theta + \boldsymbol{f}(t), \boldsymbol{f}(t) \neq 0,$$

where $\theta_f(t)$ stands for the parameter vector associated with the faulty system.

- In the second option, no hypothesis is made about the origin of the discrepancy, which is just represented as an additive vector

$$\begin{aligned}
&[\dot{\boldsymbol{x}}(t) \neq \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta(t))] \vee [\boldsymbol{y}(t) \neq \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta(t))] \\
&\iff \\
&\exists \left(\boldsymbol{f}_x, \boldsymbol{f}_y\right) \neq (0,0) : \\
&\qquad [\dot{\boldsymbol{x}}(t) = \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta(t)) + \boldsymbol{f}_x(t)] \\
&\qquad \vee \left[\boldsymbol{y}(t) = \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \theta(t)) + \boldsymbol{f}_y(t)\right].
\end{aligned}$$

In both cases, the normal and the faulty system are represented using some "fault vector" $\boldsymbol{f}(t)$ where normal operation is associated with $\boldsymbol{f}(t) = 0$. Most often, the preliminary analysis of the system has identified a set of faults that are likely to occur, and that the FDI system to be designed should detect, isolate and estimate. When such knowledge is available, it results in the logical statement

$$i \in I : \mathcal{H}_i \iff \boldsymbol{f}(t) = \boldsymbol{f}_i(\eta_i, t) \neq 0,$$

where $\mathcal{H}_i$ denotes the i^{th} fault situation, $I = \{1, 2, \cdots, n_f\}$ where n_f is the number of possible fault modes, and the knowledge available about each fault is modelled by the possible time evolution of the vector $\boldsymbol{f}$ which depends on some unknown parameters η_i (fault estimation therefore directly refers to the estimation of these parameters).

6.2.2 Analytical redundancy relations with no unknown inputs

Introducing the fault vector $\boldsymbol{f}(t)$ in the state and measurement equations, and setting $\boldsymbol{d}(t) = 0$, for all t one gets [1]

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{f}(t)) \qquad \boldsymbol{y}(t) = \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{f}(t)), \tag{6.10}$$

where, since θ is known, the dependency of the state and measurement equations on the parameter is no longer made explicit, and time invariant systems are considered in order to shorten the notations. It turns out that from (6.10), it is possible to construct *residuals*, i.e. quantities which can be computed in real time from the available data, and whose behaviour is different under the different situations $\mathcal{H}_0$ and $\mathcal{H}_1$. Such residuals are obtained from a two step construction:

Step 1: Derivation of the outputs. Assuming that all functions are differentiable with respect to their arguments, it is possible to construct the derivative $\dot{\boldsymbol{y}}(t)$ of the output signal $\boldsymbol{y}(t)$:

$$\dot{\boldsymbol{y}}(t) = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}(\cdot)\dot{\boldsymbol{x}}(t) + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{u}}(\cdot)\dot{\boldsymbol{u}}(t) + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{f}}(\cdot)\dot{\boldsymbol{f}}(t)$$

[1] The same symbols g and h as in (6.8), (6.9) are used by an abuse of notation

Replacing $\dot{\boldsymbol{x}}(t)$ by its value, one gets

$$\begin{aligned}\dot{\boldsymbol{y}}(t) &= \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}(\cdot)\boldsymbol{g}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{f}(t)) + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{u}}(\cdot)\dot{\boldsymbol{u}}(t) + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{f}}(\cdot)\dot{\boldsymbol{f}}(t) \\ &:= \boldsymbol{h}_1(\boldsymbol{x}(t),\bar{\boldsymbol{u}}^{(1)}(t),\bar{\boldsymbol{f}}^{(1)}(t)),\end{aligned}$$

where $\bar{\boldsymbol{u}}^{(1)}(t)$ is a short notation for $\left(\boldsymbol{u}', \dot{\boldsymbol{u}}'(t)\right)'$. Iterating this process until some order of derivation q (to be determined later), and assuming the existence of all required derivatives, one obtains

$$\bar{\boldsymbol{y}}^{(q)}(t) = H^q\left(\boldsymbol{x}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)\right) \tag{6.11}$$

which is a set of $(q+1)\,p$ equations - or constraints - (the dimension of $\bar{\boldsymbol{y}}^{(q)}(t)$), where the different variables have the following dimensions: $\boldsymbol{x} \in \mathbb{R}^n, \bar{\boldsymbol{u}}^{(q)}(t) \in \mathbb{R}^{(q+1)\times m}, \bar{\boldsymbol{f}}^{(q)}(t) \in \mathbb{R}^{(q+1)\times n_f}$. The known variables are $\bar{\boldsymbol{y}}^{(q)}$ and $\bar{\boldsymbol{u}}^{(q)}$ while the unknown variables are $\boldsymbol{x}$. $\bar{\boldsymbol{f}}^{(q)}(t)$ has a particular status, since it is known (equal to zero) when $\mathcal{H}_0$ is true, while it is unknown when $\mathcal{H}_1$ is true.

Example 6.2 *Redundancy in a nonlinear system*

The variable t is omitted below. Applying the above procedure with $s = 2$ to the system

$$\begin{aligned}\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} &= \begin{pmatrix} -x_1 + x_2^2 + u + f_1 \\ -2x_2 + f_2 \end{pmatrix} \\ y &= x_1 + f_3\end{aligned}$$

gives

$$\begin{aligned}\dot{y} &= -x_1 + x_2^2 + u + f_1 + \dot{f}_3 \\ \ddot{y} &= x_1 - 5x_2^2 - u - f_1 + 2x_2 f_2 + \dot{u} + \dot{f}_1 + \ddot{f}_3.\end{aligned}$$

(6.11) is thus a system of three equations

$$\bar{y}^{(2)} = \begin{pmatrix} y \\ \dot{y} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} x_1 + f_3 \\ -x_1 + x_2^2 + u + f_1 + \dot{f}_3 \\ x_1 - 5x_2^2 - u - f_1 + 2x_2 f_2 + \dot{u} + \dot{f}_1 + \ddot{f}_3 \end{pmatrix}. \ \square \tag{6.12}$$

Step 2: Elimination of the state. Assume that $(q+1)\,p > n$ and the Jacobian $\frac{\partial H^q(\cdot)}{\partial \boldsymbol{x}}$ is of rank n. Note that the first condition gives a lower bound on the order of derivation that is necessary in establishing (6.11). It follows that (6.11) can be decomposed into

$$\begin{pmatrix} \bar{\boldsymbol{y}}_m^{(q)}(t) \\ \bar{\boldsymbol{y}}_{nm}^{(q)}(t) \end{pmatrix} - \begin{pmatrix} H_m^q\left(\boldsymbol{x}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)\right) \\ H_{nm}^q\left(\boldsymbol{x}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)\right) \end{pmatrix} = 0 \tag{6.13}$$

where the first subsystem is of dimension n and allows to compute $x(t)$ (at least locally) as a function of the other variables

$$\boldsymbol{x}(t) = \phi(\bar{\boldsymbol{y}}_m^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t))$$

(this results from the implicit function theorem). Replacing $\boldsymbol{x}(t)$ by its value in the second subsystem, which is of dimension $(q+1)\,p - n$, one obtains a system that is equivalent to (6.11)

$$\begin{aligned} \boldsymbol{x}(t) &= \phi(\bar{\boldsymbol{y}}_m^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)) && (6.14)\\ 0 &= \boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)), && (6.15) \end{aligned}$$

where

$$\begin{aligned} &\boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t))\\ &= \bar{\boldsymbol{y}}_{nm}^{(q)}(t) - H_{nm}^{q}(\phi(\bar{\boldsymbol{y}}_m^{q}(t)), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)). \end{aligned}$$

The set of constraints (6.15) is seen to contain only inputs, outputs and fault signals (along with their derivatives). It is called an analytical redundancy relations (ARR) associated with the pair $(\boldsymbol{g}, \boldsymbol{h})$ and $\boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}, \bar{\boldsymbol{u}}^{(q)}, \bar{\boldsymbol{f}}^{(q)})$ is called the residual vector.

Remark 6.1 *Link to structural approach*
A structural condition for ARR to exist is that (6.11) is overconstrained with respect to the unknowns $\boldsymbol{x}(t)$ i.e. there is a matching which is complete with respect to $\boldsymbol{x}(t)$. Decomposing the set of constraints (6.11) into matched (index m) and non-matched ones (index nm) yields (6.13), where the matched subsystem has n constraints while the non-matched subsystem has $(q+1)\,p - n$ constraints. From the interpretation of matchings in the previous chapter, $\boldsymbol{x}(t)$ is computed in the matched subsystem, as a function of the other variables $\phi\left(\bar{\boldsymbol{y}}_m^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)\right)$ and replacing $\boldsymbol{x}(t)$ by its value in the non-matched subsystem gives the redundancy relations. □

Example 6.2 (cont.) *Redundancy in a nonlinear system*

Step 2 is now applied to (6.12). The state $(x_1, x_2)'$ can be computed from the first two equations of (6.12) leading to the equivalent system

$$\begin{aligned} x_1 &= y - f_3\\ x_2 &= \pm\sqrt{\dot{y} + y - f_3 - u - f_1 - \dot{f}_3}\\ 0 &= \ddot{y} - y + f_3 + 5\left(\dot{y} + y - f_3 - u - f_1 - \dot{f}_3\right) + u + ...\\ &\quad ... + f_1 + 2\left(\sqrt{\dot{y} + y - f_3 - u - f_1 - \dot{f}_3}\right) f_2 - \dot{u} - \dot{f}_1 - \ddot{f}_3, \end{aligned} \qquad (6.16)$$

where the third equation is seen to depend only on the available inputs and outputs and on the faults. □

6.2.3 Unknown inputs, exact decoupling

When unknown inputs are present, a state-space model of the system takes the form[2]

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \boldsymbol{f}(t)) \\ \boldsymbol{y}(t) &= \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{d}(t), \boldsymbol{f}(t)). \end{aligned} \tag{6.17}$$

Applying the same technique as above leads to

$$\bar{\boldsymbol{y}}^{(q)}(t) = H^q\left(\boldsymbol{x}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{d}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)\right). \tag{6.18}$$

Under the condition that $(q+1)\,p > n + (q+1)\,n_d$ and the Jacobian

$$\left[\begin{array}{cc} \frac{\partial H^q(\cdot)}{\partial \boldsymbol{x}} & \frac{\partial H^q(\cdot)}{\partial \bar{\boldsymbol{d}}^{(q)}} \end{array}\right]$$

is of rank $n + (q+1)\,n_d$ both the state and the unknown inputs can be eliminated, leading to the equivalent system

$$\left(\begin{array}{c} \boldsymbol{x}(t) \\ \bar{\boldsymbol{d}}^{(q)}(t) \end{array}\right) = \left(\begin{array}{c} \phi_x(\bar{\boldsymbol{y}}_m^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)) \\ \phi_d(\bar{\boldsymbol{y}}_m^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)) \end{array}\right) \tag{6.19}$$

$$0 = \boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)), \tag{6.20}$$

where (6.20) are the analytical redundancy relations, which are independent of the unknown inputs, hence the name "exact decoupling" which is given to this approach. Note that exact decoupling is possible only if the structural graph of system (6.18) is overconstrained with respect to both the unknowns $\boldsymbol{x}$ and $\bar{\boldsymbol{d}}^{(q)}$.

6.2.4 How to find analytical redundancy relations

There are several procedures by which ARR can be found. They all rest on the elimination of $\boldsymbol{x}(t)$ (and $\bar{\boldsymbol{d}}^{(q)}(t)$ when unknown inputs are present), either by starting with (6.8) and (6.9) or by establishing first (6.11).

Elimination procedures fit the nature of the functions $\boldsymbol{g}$ and $\boldsymbol{h}$. When all functions are linear, projection approaches are well suited: this is the parity space approach which will be described in Section 6.3. Most often, nonlinear models involve polynomial functions (because polynomials can approximate any smooth nonlinear funtion). There are, basically, three elimination techniques for polynomial functions. All three require the components of the state to be eliminated according to some selected order. *Elimination theory* rests on Euclidean division and derivation. *Gröbner bases* uses Euclidean division and the computation of so called S-polynomials. *Characteristic sets* (also called Ritt's algorithm) rest on Euclidean division and derivation. The state is directly eliminated from the system (6.8) (6.9), and ARR with minimum derivative order can be obtained.

[2] Again the same functions g and h as above are used by an abuse of notation.

6.2.5 ARR-based diagnosis

Fault detection. In the absence of unknown inputs, or when exact decoupling is possible, the following logical statements hold

$$\begin{array}{lllll} (6.10) & \Longleftrightarrow & (6.14), (6.15) & \Rightarrow & \boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)) = 0 \\ (6.17) & \Longleftrightarrow & (6.19) & \Rightarrow & \boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)) = 0 \end{array} \tag{6.21}$$

From (6.21) it follows that in both cases necessary conditions for normal system operation are given by

$$\mathcal{H}_0 \Rightarrow \boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), 0) = 0$$

Therefore, fault detection immediately follows from

$$\boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), 0) \neq 0 \quad \Rightarrow \mathcal{H}_1.$$

Remark 6.2 *Non detectable faults*
Note that $\boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), 0) = 0$ does not imply $\mathcal{H}_0$ since the condition expressed by the analytical redundancy relation is only necessary. In fact, $\boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), 0) = 0$ is to be read: $\mathcal{H}_0$ is not falsified by the observations, or in other terms "it is not impossible that the system is healthy". In fact, special fault values that are not detectable through analytical redundancy could exist. They correspond to nonzero values of $\boldsymbol{f}(t)$ that yield $\boldsymbol{r}(\bar{\boldsymbol{y}}^{(q)}(t), \bar{\boldsymbol{u}}^{(q)}(t), \bar{\boldsymbol{f}}^{(q)}(t)) = 0$. □

Example 6.2 (cont.) *Redundancy in a nonlinear system*

The redundancy relation in (6.16) writes

$$\begin{aligned} & \ddot{y} + 5\dot{y} + 4y - 4u - \dot{u} \\ = \quad & f_1 - 2\left(\sqrt{\dot{y} + y - f_3 - u - f_1 - \dot{f}_3}\right) f_2 + 4f_3 + \dot{f}_1 + 5\dot{f}_3 + \ddot{f}_3. \end{aligned}$$

Therefore, the residual is

$$r\left(\bar{y}^{(2)}, \bar{u}^{(2)}, 0\right) = \ddot{y} + 5\dot{y} + 4y - 4u - \dot{u}$$

and the fault detection rule is

$$\ddot{y} + 5\dot{y} + 4y - 4u - \dot{u} \neq 0 \Rightarrow \mathcal{H}_1. \; \square$$

Fault isolation. Fault isolation is approached in a similar way, by the design of so-called structured residuals. Assume it is possible to separate the set of faults I into two subsets I_1 and I_2 such that $I = I_1 \cup I_2$. Set

$$\boldsymbol{f}(t) = \left(\begin{array}{cc} \boldsymbol{f}_{I_1}(t)' & \boldsymbol{f}_{I_2}(t)' \end{array} \right)',$$

where only $\boldsymbol{f}_{I_1}(t)$ ($\boldsymbol{f}_{I_2}(t)$) is nonzero upon occurrence of a fault in I_1 (I_2). If the set of residuals can also be separated in two subsets

$$\boldsymbol{r}(\bar{y}^{(s)}(t), \bar{u}^{(s)}(t), \bar{f}^{(s)}(t)) = \begin{pmatrix} \boldsymbol{r}_1(\bar{y}^{(s)}(t), \bar{u}^{(s)}(t), \bar{f}^{(s)}(t)) \\ \boldsymbol{r}_2(\bar{y}^{(s)}(t), \bar{u}^{(s)}(t), \bar{f}^{(s)}(t)) \end{pmatrix}, \tag{6.22}$$

so that (a) $\boldsymbol{r}_1$ is insensitive to faults in I_2 but sensitive to faults in I_1 while (b) $\boldsymbol{r}_2$ is insensitive to faults in I_1 but sensitive to faults in I_2, then, as shown below, it is possible to distinguish between the occurrence of a fault from the class I_1 or I_2. The logical expressions corresponding to these assumptions are

$$\begin{array}{ll} (a) & \left.\begin{array}{l} \forall i \in I_1 : \boldsymbol{f}_{I_1}(t) = \boldsymbol{f}_i(\eta_i, t) = 0 \\ \exists i \in I_2 : \boldsymbol{f}_{I_2}(t) = \boldsymbol{f}_i(\eta_i, t) \neq 0 \end{array}\right\} \Rightarrow \begin{array}{l} \boldsymbol{r}_1(\bar{y}^{(s)}(t), \bar{u}^{(s)}(t), \bar{f}^{(s)}(t)) = 0 \\ \boldsymbol{r}_2(\bar{y}^{(s)}(t), \bar{u}^{(s)}(t), \bar{f}^{(s)}(t)) \neq 0 \end{array} \\ (b) & \left.\begin{array}{l} \exists i \in I_1 : \boldsymbol{f}_{I_1}(t) = \boldsymbol{f}_i(\eta_i, t) \neq 0 \\ \forall i \in I_2 : \boldsymbol{f}_{I_2}(t) = \boldsymbol{f}_i(\eta_i, t) = 0 \end{array}\right\} \Rightarrow \begin{array}{l} \boldsymbol{r}_1(\bar{y}^{(s)}(t), \bar{u}^{(s)}(t), \bar{f}^{(s)}(t)) \neq 0 \\ \boldsymbol{r}_2(\bar{y}^{(s)}(t), \bar{u}^{(s)}(t), \bar{f}^{(s)}(t)) = 0. \end{array} \end{array}$$

There are four possible situations (logical 0 means $\boldsymbol{r} = 0$ while logical 1 means $\boldsymbol{r} \neq 0$) and the following conclusions are true.

$\boldsymbol{r}_1$	$\boldsymbol{r}_2$	Conclusion
0	0	$\mathcal{H}_0$ is not falsified (no fault is detected)
0	1	$\mathcal{H}_0$ is falsified by a fault $i \in I_2$
1	0	$\mathcal{H}_0$ is falsified by a fault $i \in I_1$
1	1	$\mathcal{H}_0$ is falsified by a fault $i \in I_1$ and a fault $j \in I_2$

Therefore, under (6.22), it is possible to isolate a fault within the subset I_1 or within the subset I_2. By designing several partitions of the set of faults into two classes it is obviously possible to isolate faults within smaller subsets that result from the intersections of all these partitions.

Remark 6.3 *Non isolable faults*

Only a limited number of partitions into two classes enjoying property (6.22) can be obtained for a given system. Therefore, it may happen that whatever the partition such that (6.22) holds, two given faults, say i and j are always in the same class. These faults always have the same effect on the analytical redundancy relations, and therefore they are not isolable from each other, which means that every FDI conclusion will contain "the fault is either i or j (or both)". □

Example 6.3 *Two-tank system*

In Example 5.40, the set of constraints associated with the two-tank system components wrote

Pump:	$q_P = u.f(h_1)$
Tank 1:	$\dot{h}_1 = \frac{1}{A}(q_P - q_L - q_{12})$
Tank 2:	$\dot{h}_2 = \frac{1}{A}(q_{12} - q_2)$
Pipe between tanks ($h_1 > h_2$):	$q_{12} = k_1\sqrt{h_1 - h_2}$
Output pipe:	$q_2 = k_2\sqrt{h_2}$
Outflow measurement:	$q_m = k_m.q_2.$

The state-space equations are

$$\begin{pmatrix} \dot{h}_1 \\ \dot{h}_2 \end{pmatrix} = \begin{pmatrix} -\frac{k_1}{A}\sqrt{h_1 - h_2} + \frac{f(h_1)}{A}.u - \frac{1}{A}.q_L \\ \frac{k_1}{A}\sqrt{h_1 - h_2} - \frac{k_2}{A}\sqrt{h_2} \end{pmatrix} \tag{6.23}$$

and the measurement equation is

$$q_m = k_m k_2 \sqrt{h_2}. \tag{6.24}$$

Derivating once the output gives

$$\begin{aligned} \dot{q}_m &= k_m k_2 \left(h_2\right)^{-1/2} \dot{h}_2 \qquad (6.25) \\ \dot{q}_m &= k_m k_2 \left(h_2\right)^{-1/2} \left(\frac{k_1}{A}\sqrt{h_1 - h_2} - \frac{k_2}{A}\sqrt{h_2}\right). \end{aligned}$$

From (6.24) and (6.25) the two states h_1 and h_2 can be computed

$$\begin{aligned} h_2 &= \left(\frac{q_m}{k_m k_2}\right)^2 \qquad (6.26) \\ h_1 &= q_m^2 \left(1 + \left(1 + \dot{q}_m\right)^2\right). \end{aligned}$$

Derivating once again gives

$$\ddot{q}_m = \frac{\left(h_1 - h_2\right)^{-1/2} \sqrt{h_2} \left(\dot{h}_1 - \dot{h}_2\right) - \left(h_2\right)^{-1/2} \dot{h}_2 \left(h_1 - h_2\right)^{1/2}}{h_2},$$

where replacing $h_1, h_2, \dot{h}_1, \dot{h}_2$ by their values taken from (6.26), (6.23) and (6.24) – (6.25) gives the redundancy relation

$$\begin{aligned} & r(q_m, \dot{q}_m, \ddot{q}_m, u, q_L) \\ &= \sqrt{h_2} \left(h_1 - h_2\right)^{1/2} \ddot{q}_m - \dot{h}_1 + \dot{h}_2 + \left(h_2\right)^{-1} \dot{h}_2 \left(h_1 - h_2\right) = 0 \qquad (6.27) \end{aligned}$$

and the leakage detection rule

$$r(q_m, \dot{q}_m, \ddot{q}_m, u, 0) \neq 0 \Rightarrow q_L \neq 0. \square$$

6.3 Analytical redundancy relations for linear deterministic systems – time domain

Let us consider the following continuous-time state-space model

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{E}_x\, \boldsymbol{d}(t) + \boldsymbol{F}_x\, \boldsymbol{f}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0, \qquad (6.28) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) + \boldsymbol{E}_y \boldsymbol{d}(t) + \boldsymbol{F}_y \boldsymbol{f}(t), \end{aligned}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ denotes the state vector, $\boldsymbol{u} \in \mathbb{R}^m$, is the vector of measured input signals, $\boldsymbol{y} \in \mathbb{R}^p$ is the vector of measured plant output signals, $\boldsymbol{d} \in \mathbb{R}^{n_d}$ and $\boldsymbol{f} \in \mathbb{R}^{n_f}$ are vectors of unknown input signals. $\boldsymbol{f}$ represents the faults one wishes to detect, while $\boldsymbol{d}$ are unknown disturbances that should not be detected.

The aim is to solve the following problem.

Problem 6.1 (Design of linear analytical redundancy relations)
Given a model of the supervised process of the form (6.28), determine, if possible, a set of linear relations between the measured inputs and outputs and their derivatives up to a certain order, say q, such that,

- *in the absence of fault,*

$$\sum_{i=1}^{q} \boldsymbol{W}_{y,i}\boldsymbol{y}^{(i)}(t) + \sum_{i=1}^{q} \boldsymbol{W}_{u,i}\boldsymbol{u}^{(i)}(t) = 0,$$

  where $\boldsymbol{z}^{(i)}(t)$ denotes the i^{th} derivative of $\boldsymbol{z}(t)$ and $\boldsymbol{W}_{y,i}$, $\boldsymbol{W}_{u,i}$ are $n_r \times p$ and $n_r \times m$ matrices of real elements, n_r being the number of relations (to be determined),
- *in the presence of a fault,*

$$\sum_{i=1}^{q} \boldsymbol{W}_{y,i}\boldsymbol{y}^{(i)}(t) + \sum_{i=1}^{q} \boldsymbol{W}_{u,i}\boldsymbol{u}^{(i)}(t) \neq 0.$$

  Such relations are a particular kind of analytical redundancy relations called parity relations.

In order to solve this problem, let us consider the successive time derivatives of $\boldsymbol{y}$ up to order q:

$$\begin{array}{rcl} \boldsymbol{y}(t) & = & \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) + \boldsymbol{E}_y\boldsymbol{d}(t) + \boldsymbol{F}_y\boldsymbol{f}(t) \\ \dot{\boldsymbol{y}}(t) & = & \boldsymbol{C}\dot{\boldsymbol{x}}(t) \\ & = & \boldsymbol{C}\boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{C}\boldsymbol{B}\boldsymbol{u}(t) + D\dot{\boldsymbol{u}}(t) + \boldsymbol{C}\boldsymbol{E}_x\,\boldsymbol{d}(t) \\ & & +\boldsymbol{E}_y\;\dot{\boldsymbol{d}}(t) + \boldsymbol{C}\boldsymbol{F}_x\,\boldsymbol{f}(t) + \boldsymbol{F}_y\,\dot{\boldsymbol{f}}(t), \end{array} \tag{6.29}$$

where the last equality is deduced by substitution of (6.28) for $\dot{\boldsymbol{x}}(t)$. By iterating this process, the following expression for the q^{th} derivative of $\boldsymbol{y}$ is obtained:

$$\begin{array}{rcl} \boldsymbol{y}^{(q)}(t) & = & \boldsymbol{C}\boldsymbol{A}^q\boldsymbol{x}(t) + \boldsymbol{C}\boldsymbol{A}^{(q-1)}\boldsymbol{B}\boldsymbol{u}(t) + \ldots + \boldsymbol{C}\boldsymbol{B}\boldsymbol{u}^{(q-1)}(t) + \boldsymbol{D}\boldsymbol{u}^{(q)}(t) + \\ & & + \boldsymbol{C}\boldsymbol{A}^{(q-1)}\boldsymbol{E}_x\boldsymbol{d}(t) + \ldots + \boldsymbol{C}\boldsymbol{E}_x\,\boldsymbol{d}^{(q-1)}(t) + \boldsymbol{E}_y\,\boldsymbol{d}^{(q)}(t) + \\ & & + \boldsymbol{C}\boldsymbol{A}^{(q-1)}\boldsymbol{F}_x\boldsymbol{f}(t) + \ldots + \boldsymbol{C}\boldsymbol{F}_x\,\boldsymbol{f}^{(q-1)}(t) + \boldsymbol{F}_y\,\boldsymbol{f}^{(q)}(t). \end{array} \tag{6.30}$$

The above set of equations can be concatenated into the expression

$$\bar{\boldsymbol{y}}^{(q)}(t) = \mathcal{O}\boldsymbol{x}(t) + \boldsymbol{T}_{u,q}\bar{\boldsymbol{u}}^{(q)}(t) + \boldsymbol{T}_{d,q}\bar{\boldsymbol{d}}^{(q)}(t) + \boldsymbol{T}_{f,q}\bar{\boldsymbol{f}}^{(q)}, \tag{6.31}$$

where $\bar{y}^{(q)}(t) = \left(\boldsymbol{y}(t)'\dot{\boldsymbol{y}}(t)'\ldots\boldsymbol{y}^{(q)}(t)'\right)'$, and $\bar{\boldsymbol{u}}^{(q)}(t)$, $\bar{\boldsymbol{d}}^{(q)}(t)$, $\bar{\boldsymbol{f}}^{(q)}(t)$ have a similar form with $\boldsymbol{u}(t)$, $\boldsymbol{d}(t)$ and $\boldsymbol{f}(t)$ substituted for $\boldsymbol{y}(t)$,

$$\mathcal{O} = \begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{C}\boldsymbol{A} \\ \vdots \\ \boldsymbol{C}\boldsymbol{A}^q \end{pmatrix}, \qquad \boldsymbol{T}_{u,q} = \begin{pmatrix} \boldsymbol{D} & 0 & 0 & \ldots & 0 \\ \boldsymbol{C}\boldsymbol{B} & \boldsymbol{D} & 0 & \ldots & 0 \\ \vdots & \ddots & \ddots & & \\ \boldsymbol{C}\boldsymbol{A}^{q-1}\boldsymbol{B} & \ldots & \ldots & \boldsymbol{C}\boldsymbol{B} & \boldsymbol{D} \end{pmatrix}$$

and a similar definition holds for the block Toeplitz matrices $\boldsymbol{T}_{d,q}, \boldsymbol{T}_{f,q}$ with respectively $\boldsymbol{E}_y$ and $\boldsymbol{E}_x$ or $\boldsymbol{F}_y$ and $\boldsymbol{F}_x$ substituted for $\boldsymbol{D}$ and $\boldsymbol{B}$.

If there exists a value of q such that

$$\operatorname{rank}\begin{pmatrix} \mathcal{O} & \boldsymbol{T}_{d,q} \end{pmatrix} < (q+1)p,$$

the left nullspace of $(\mathcal{O}\ \boldsymbol{T}_{d,q})$ is not empty. The dimension of this subspace, say n_r, is given as $n_r = (q+1)p - \operatorname{rank}\begin{pmatrix} \mathcal{O} & \boldsymbol{T}_{d,q} \end{pmatrix}$. Let $\boldsymbol{W}$ be a $n_r \times (q+1)p$ matrix of which each row is a basis vector for this subspace. Multiplying (6.31) on the left by $\boldsymbol{W}$ results in the following equality

$$\boldsymbol{W}\bar{\boldsymbol{y}}^{(q)}(t) - \boldsymbol{W}\boldsymbol{T}_{u,q}\bar{\boldsymbol{u}}^{(q)}(t) = \boldsymbol{W}\boldsymbol{T}_{f,q}\bar{\boldsymbol{f}}^{(q)}(t), \tag{6.32}$$

since $\boldsymbol{W}$ has been specifically computed to eliminate the terms in $\boldsymbol{x}(t)$ and $\bar{\boldsymbol{d}}^{(q)}(t)$. Equation (6.32) describes n_r analytical redundancy relations. Indeed, in the absence of fault, the right hand side is equal to zero, and it is normally different from zero in the presence of a fault.

In order to implement such relations, and thus to compute the quantity

$$\boldsymbol{r}(t) = \boldsymbol{W}\bar{\boldsymbol{y}}^{(q)}(t) - \boldsymbol{W}\boldsymbol{T}_{u,q}\bar{\boldsymbol{u}}^{(q)}(t), \tag{6.33}$$

it is necessary to evaluate the derivatives that appear in the above relation. Such signals are highly sensitive to noise, so that filtered estimates of the derivatives have to be used. One approach is to resort to a so-called state variable filter, which amounts to implementing the scheme of Fig. 6.2. Such a filter is used for each component of $\boldsymbol{y}(t)$ and $\boldsymbol{u}(t)$. Letting $z(t)$ denote the input of such a filter, the i^{th} integrator output provides the i^{th} filtered derivative of z, $z_f^{(i)}$. This filter corresponds to the analog simulation of the observability canonical state-space representation for the relation

$$z_f(s) = \frac{1}{s^q + a_1 s^{(q-1)} + \ldots + a_q} z(s).$$

By taking this filter into account, (6.33) can be rewritten in the frequency domain as

$$\boldsymbol{r}_f(s) = (\boldsymbol{W}_y(s)\boldsymbol{y}(s) + \boldsymbol{W}_u(s)\boldsymbol{u}(s))/p_f(s), \tag{6.34}$$

where

$$\boldsymbol{W}_y(s) = \sum_{i=0}^{q} \boldsymbol{W}_i s^i$$

with $\boldsymbol{W}_i$, the matrix made of columns $i\,p+1$ to $(i+1)p$ of $\boldsymbol{W}$, $\boldsymbol{W}_u(s)$ is defined similarly with $\boldsymbol{W}\boldsymbol{T}_{u,q}$ substituted for $\boldsymbol{W}$ and

$$p_f(s) = s^q + a_1 s^{(q-1)} + \ldots + a_q.$$

Vector $\boldsymbol{r}$ is called a parity vector. It has generally different directions and magnitudes in the presence of the different fault modes. The n_r dimensional space of all such vectors is called the parity space, and any linear combination of the rows of (6.33) is called a parity relation.

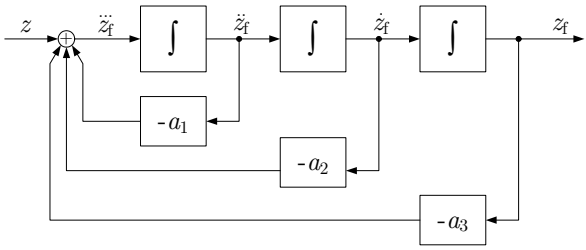


Fig. 6.2. Block diagram of a third-order state variable filter

The procedure for designing and implementing parity relations is now summarised.

Algorithm 6.1 *Parity relations for deterministic linear systems*

Given:	A linear state-space model of the form (6.28) and a suitable order of derivation q
Compute off-line: 1.	Matrices $\mathcal{O}, \boldsymbol{T}_{d,q}, \boldsymbol{T}_{u,q}$
2.	A basis $\boldsymbol{W}$ for the left null space of $(\mathcal{O} \;\; \boldsymbol{T}_{d,q})$
3.	State space filters for the estimation of the derivatives of $\boldsymbol{y}$ and $\boldsymbol{u}$ up to order q.
At each time instant:	
1.	Acquire the new data $\boldsymbol{y}(t)$, $\boldsymbol{u}(t)$.
2.	Compute $\boldsymbol{r}_f(t)$ from (6.34).
Result:	A residual vector $\boldsymbol{r}_f(t)$ for an increasing time horizon.

An alternative approach to determine analytical redundancy relations can be deduced from the input-output model of the supervised process, namely in the frequency domain. It directly results in relations involving the filtered derivatives of the measured signals. By extension this method is called the (generalised) parity space approach. It is the object of the next section. Fault isolation can be handled in the linear case in a similar way as for the nonlinear case. The detailed treatment of this issue is deferred to Section 6.4.3.

Example 6.4 *Parity relations for the ship*

A linearised model of the ship example can be written as

$$\begin{pmatrix} \dot{\omega}_3 \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} b\eta_1 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \omega_3 \\ \psi \end{pmatrix} + \begin{pmatrix} b \\ 0 \end{pmatrix} \delta + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \omega_w \tag{6.35}$$

$$\begin{pmatrix} \omega_{3m} \\ \psi_m \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \omega_3 \\ \psi \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} f_\omega \\ f_\psi \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \omega_w, \tag{6.36}$$

when linearisation around $\omega_3 = 0$ is considered. Here δ, the rudder angle, is a known input, while ω_w, the wave disturbance, is an unknown input.

Straightforward computations yield the following expression for (6.31) with $q = 1$:

$$\begin{pmatrix} \omega_{3m} \\ \psi_m \\ \dot{\omega}_{3m} \\ \dot{\psi}_m \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ b\eta_1 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \omega_3 \\ \psi \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ b & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \delta \\ \dot{\delta} \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \omega_w \\ \dot{\omega}_w \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f_\omega \\ f_\psi \\ \dot{f}_\omega \\ \dot{f}_\psi \end{pmatrix} \quad (6.37)$$

The block matrix $(\mathcal{O} \ \boldsymbol{T}_{d,1})$ takes the form:

$$(\mathcal{O} \ \boldsymbol{T}_{d,1}) = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ b\eta_1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}.$$

A basis vector for the one-dimensional left nullspace of this matrix can be written

$$\boldsymbol{W} = \begin{pmatrix} 1 & 0 & 0 & -1 \end{pmatrix}.$$

Expression (6.32) then yields

$$\omega_{3m} - \dot{\psi}_m = f_\omega - \dot{f}_\psi.$$

Hence a residual can be computed according as:

$$r_f(s) = (\omega_{3m}(s) - s\psi_m(s))/(s + a), \quad (6.38)$$

where a is a design parameter to be adjusted according to the noise level. This expression is a particulat case of the more general form (6.52) for a residual for the ship example. Further discussion of the proposed residual is provided in Section 6.4. □

6.4 Analytical redundancy relations for linear deterministic systems – frequency domain

In this section, the problems of fault detection, fault isolation and fault estimation are solved using the parity space approach to residual generation, from an input-output model of the supervised system. An alternative method would be to design observer-based residual generators, which yields similar filters, as indicated in the bibliographical notes. The observer-based approach will be used in the section on diagnosis systems design from a stochastic model, so that the reader will be acquainted with both methods.

6.4.1 Fault detection

Consider again a system described by a linear continuous-time state-space model of the form

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{E}_x\,\boldsymbol{d}(t) + \boldsymbol{F}_x\,\boldsymbol{f}(t), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) + \boldsymbol{E}_y\boldsymbol{d}(t) + \boldsymbol{F}_y\boldsymbol{f}(t), \end{aligned} \tag{6.39}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ denotes the state vector, $\boldsymbol{u} \in \mathbb{R}^m$, is the vector of measured input signals, $\boldsymbol{y} \in \mathbb{R}^p$ is the vector of measured plant output signals, $\boldsymbol{d} \in \mathbb{R}^{n_d}$ and $\boldsymbol{f} \in \mathbb{R}^{n_f}$ are vectors of unknown input signals. $\boldsymbol{f}$ represents the faults one wishes to detect, while $\boldsymbol{d}$ are unknown disturbances that should not be detected.

Such a model can also be written in terms of transfer functions:

$$\boldsymbol{y}(s) = \boldsymbol{H}_{yu}(s)\boldsymbol{u}(s) + \boldsymbol{H}_{yd}(s)\boldsymbol{d}(s) + \boldsymbol{H}_{yx}(s)\boldsymbol{x}(0) + \boldsymbol{H}_{yf}(s)\boldsymbol{f}(s), \tag{6.40}$$

where

$$\begin{aligned} \boldsymbol{H}_{yu}(s) &= \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} + \boldsymbol{D} \\ \boldsymbol{H}_{yx}(s) &= \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1} \\ \boldsymbol{H}_{yd}(s) &= \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{E}_x + \boldsymbol{E}_y \\ \boldsymbol{H}_{yf}(s) &= \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{F}_x + \boldsymbol{F}_y. \end{aligned}$$

As indicated in Fig. 6.1, a residual generator is a filter with input $\boldsymbol{u}$ and $\boldsymbol{y}$. As supervision of linear time-invariant systems is addressed here, the class of considered filters will be restricted to linear time-invariant systems of the following form

$$\begin{aligned} \dot{\boldsymbol{z}}(t) &= \boldsymbol{A}_z\boldsymbol{z}(t) + \boldsymbol{B}_{zu}\,\boldsymbol{u}(t) + \boldsymbol{B}_{zy}\,\boldsymbol{y}(t), \qquad \boldsymbol{z}(0) = \boldsymbol{z}_0 \\ \boldsymbol{r}(t) &= \boldsymbol{C}_{rz}\,\boldsymbol{z}(t) + \boldsymbol{D}_{ru}\,\boldsymbol{u}(t) + \boldsymbol{D}_{ry}\,\boldsymbol{y}(t) \end{aligned} \tag{6.41}$$

or, in transfer function form, assuming zero initial conditions:

$$\boldsymbol{r}(s) = \boldsymbol{V}_{ru}(s)\boldsymbol{u}(s) + \boldsymbol{V}_{ry}(s)\boldsymbol{y}(s) = \begin{pmatrix} \boldsymbol{V}_{ru}(s) & \boldsymbol{V}_{ry}(s) \end{pmatrix} \begin{pmatrix} \boldsymbol{u}(s) \\ \boldsymbol{y}(s) \end{pmatrix}. \tag{6.42}$$

The problem of residual generator design for fault detection based on a deterministic model can be stated as follows:

Problem 6.2 (Residual generator design for fault detection based on a deterministic model)
Given a model of the supervised process of the form (6.39) or (6.40) determine a stable linear time-invariant system (6.41) or (6.42) such that:

- *In the absence of fault ($\boldsymbol{f}(t) = 0$ for all t), the output signal $\boldsymbol{r}(t), t > 0$ asymptotically decays to zero for any input $\boldsymbol{u}(t), \boldsymbol{d}(t), t > 0$ and any initial conditions $\boldsymbol{x}(0), \boldsymbol{z}(0)$.*
- *$\boldsymbol{r}(t)$ is affected by $\boldsymbol{f}(t)$.*

The first condition assures that, after a transient due to the effect of initial conditions, the residual is almost equal to zero. The second condition is a fault detectability[3] condition. The output $\boldsymbol{r}(t)$ is affected by $\boldsymbol{f}(t)$ when the transfer matrix between $\boldsymbol{f}(s)$ and $\boldsymbol{r}(s)$ obtained by combining (6.40) and (6.42) is non-zero. A time domain definition of this notion is somewhat more cumbersome, hence we defer it to Section 6.8.2.

Quite often, each component of the vector $\boldsymbol{f}(t)$ corresponds to a different fault. The detectability condition is then defined component-wise. One distinguishes the following notions:

Definition 6.1 (Weak detectability)
The i^{th} fault ($f_i(t) \neq 0$ for all $t \geq t_0$) is weakly detectable if there exists a stable residual generator such that $\boldsymbol{r}(t)$ is affected by $f_i(t)$.

In the literature weak detectability is also referred to as detectability.

Definition 6.2 (Strong detectability)
A fault f_i is strongly detectable if there exists a stable residual generator such that $\boldsymbol{r}(t)$ reaches a non-zero steady-state value for a fault signal that has a bounded final value different from zero.

6.4.2 Solution by the parity space approach

In order to determine the conditions to be fulfilled by $\boldsymbol{V}_{ru}(s)$ and $\boldsymbol{V}_{ry}(s)$ for (6.42) to be a residual generator, (6.40) is substituted for $\boldsymbol{y}(s)$ in (6.42):

$$\begin{aligned}
\boldsymbol{r}(s) &= \boldsymbol{V}_{ru}(s)\boldsymbol{u}(s) + \boldsymbol{V}_{ry}(s)\left(\boldsymbol{H}_{yu}(s)\boldsymbol{u}(s) + \boldsymbol{H}_{yx}(s)\boldsymbol{x}(0)\right. \\
&\quad \left. + \boldsymbol{H}_{yd}(s)\boldsymbol{d}(s) + \boldsymbol{H}_{yf}(s)\boldsymbol{f}(s)\right) \\
&= \left(\boldsymbol{V}_{ru}(s) + \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yu}(s) \quad \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yd}(s)\right)\begin{pmatrix}\boldsymbol{u}(s)\\ \boldsymbol{d}(s)\end{pmatrix} \\
&\quad + \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yx}(s)\boldsymbol{x}(0) + \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yf}(s)\boldsymbol{f}(s)
\end{aligned} \tag{6.43}$$

Figure 6.3 illustrates this residual generator.

Fulfilment of the first condition in Problem 6.2 requires:

$$\left(\boldsymbol{V}_{ru}(s) + \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yu}(s) \quad \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yd}(s)\right) = \boldsymbol{O} \tag{6.44}$$

together with the asymptotic stability of $\boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yx}(s)$. Since, in healthy working mode, the plant is normally stabilised by an appropriate controller, the latter condition amounts to requiring the stability of the filter. This can be guaranteed by an

[3] This notion should not be confused with the detectability of a linear system or a pair $(C,\ A)$; indeed, the latter notion depends on the map from state to measured output, while the fault detectability is an input(i.e. fault)/output(i.e. residual) property.

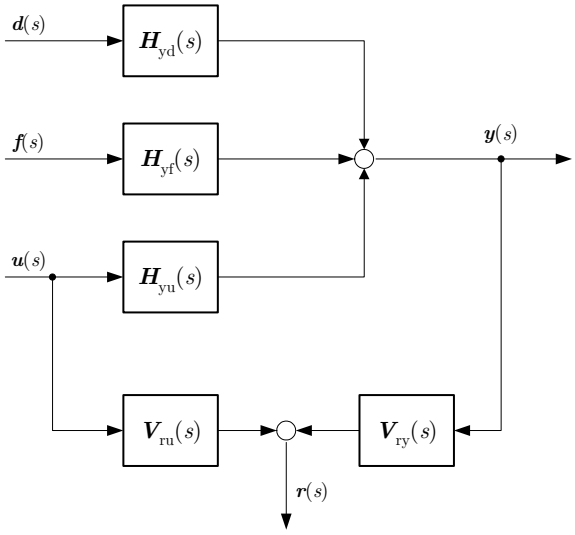


Fig. 6.3. Structure of residual generator in the parity space formulation

appropriate choice of the denominator of $\boldsymbol{V}_{ru}(s)$ and $\boldsymbol{V}_{ry}(s)$. Therefore, concentrate now on the way to achieve (6.44). The question of fault detectability will be addressed once the class of all filters that fulfil (6.44) is characterised.

Notice that (6.44) can be rewritten:

$$\begin{pmatrix} \boldsymbol{V}_{ry}(s) & \boldsymbol{V}_{ru}(s) \end{pmatrix} \begin{pmatrix} \boldsymbol{H}_{yu}(s) & \boldsymbol{H}_{yd}(s) \\ \boldsymbol{I} & \boldsymbol{O} \end{pmatrix} = 0. \tag{6.45}$$

For any filter, the least common multiple of the denominators of the entries of $\boldsymbol{V}_{ry}(s)$ and $\boldsymbol{V}_{ru}(s)$, $p(s)$ can be determined. Using $p(s)$, the left most matrix in (6.45) can be written:

$$\begin{pmatrix} \boldsymbol{V}_{ry}(s) & \boldsymbol{V}_{ru}(s) \end{pmatrix} = \frac{\begin{pmatrix} \bar{\boldsymbol{V}}_{ry}(s) & \bar{\boldsymbol{V}}_{ru}(s) \end{pmatrix}}{p(s)}, \tag{6.46}$$

where $\bar{\boldsymbol{V}}_{ry}(s)$ and $\bar{\boldsymbol{V}}_{ru}(s)$ are suitable polynomial matrices. Hence, the whole class of filters that meet (6.45) can be obtained by characterising the set of polynomial matrices $\begin{pmatrix} \bar{\boldsymbol{V}}_{ry}(s) & \bar{\boldsymbol{V}}_{ru}(s) \end{pmatrix}$ that fulfil:

$$\begin{pmatrix} \bar{\boldsymbol{V}}_{ry}(s) & \bar{\boldsymbol{V}}_{ru}(s) \end{pmatrix} \begin{pmatrix} \boldsymbol{H}_{yu}(s) & \boldsymbol{H}_{yd}(s) \\ \boldsymbol{I} & \boldsymbol{O} \end{pmatrix} = 0. \tag{6.47}$$

This is the set of polynomial matrices that lie in the left nullspace of

$$\boldsymbol{H}(s) = \begin{pmatrix} \boldsymbol{H}_{yu}(s) & \boldsymbol{H}_{yd}(s) \\ \boldsymbol{I} & \boldsymbol{O} \end{pmatrix}. \tag{6.48}$$

This space is denoted $\mathcal{N}_L(\boldsymbol{H}(s))$. Its dimension is equal to the difference between the number of rows of $\boldsymbol{H}(s)$ and its rank, namely

$$\dim(\mathcal{N}_L(\boldsymbol{H}(s))) = m + p - \text{ rank } \boldsymbol{H}(s) = m + p - (m + n_d) = p - n_d,$$

where m is the number of inputs, p the number of outputs, and n_d the number of unknown inputs (disturbances). It has been assumed that $\boldsymbol{H}_{yu}(s)$ and $\boldsymbol{H}_{yd}(s)$ have full column rank[4]. Notice that the number of plant output signals must be larger than the number of disturbances for the left nullspace to be non-zero.

One way to characterise the set of polynomial matrices $\begin{pmatrix}\bar{\boldsymbol{V}}_{ry}(s) & \bar{\boldsymbol{V}}_{ru}(s)\end{pmatrix}$ that meet (6.47) is to determine an irreducible polynomial basis, for the rational vector space $\mathcal{N}_L(\boldsymbol{H}(s))$. Further, let $\boldsymbol{F}(s)$ be a matrix of which the rows make such an irreducible polynomial basis, then any suitable matrix $\begin{pmatrix}\bar{\boldsymbol{V}}_{ry}(s) & \bar{\boldsymbol{V}}_{ru}(s)\end{pmatrix}$ can be obtained by combinations of the rows of $\boldsymbol{F}(s)$, namely

$$\begin{pmatrix}\bar{\boldsymbol{V}}_{ry}(s) & \bar{\boldsymbol{V}}_{ru}(s)\end{pmatrix} = \boldsymbol{Q}(s)\boldsymbol{F}(s), \tag{6.49}$$

where $\boldsymbol{Q}(s)$ is an arbitrary polynomial matrix with appropriate number of columns.

A general parametrisation of the family of residual generators is obtained from (6.49). Substitution of (6.49) into (6.46) yields

$$\begin{pmatrix}\boldsymbol{V}_{ry}(s) & \boldsymbol{V}_{ru}(s)\end{pmatrix} = \frac{\boldsymbol{Q}(s)\boldsymbol{F}(s)}{p(s)}.$$

Introducing this expression into (6.42) finally results in

$$r(s) = \frac{\boldsymbol{Q}(s)\boldsymbol{F}(s)}{p(s)} \begin{pmatrix}\boldsymbol{y}(s) \\ \boldsymbol{u}(s)\end{pmatrix}. \tag{6.50}$$

The choice of the matrix $\boldsymbol{Q}(s)$ and the polynomial $p(s)$ depends on the specification of the diagnosis problem. Typically the residual generator should ensure filtering of high frequency disturbances which always exist, even though they were not considered in the model, and adequate properties at low frequencies. Sometimes, precise information on the frequency range of the fault is available, and $\boldsymbol{Q}(s)/p(s)$ can be designed to perform appropriate filtering.

Remark 6.4 *Link with parity relations deduced from the state-space model*
Equation (6.34) clearly has the same form as (6.42) and, by construction, it fulfils the first condition of problem 6.2 provided $p_f(s)$ has all its roots in the open left-half plane. Hence there exist a matrix $\boldsymbol{Q}(s)$ and a polynomial $p(s)$ for which (6.34) and (6.50) are identical. □

Modelling uncertainty. Although modelling uncertainties have not been introduced here, they can be accounted for a posteriori when $\boldsymbol{F}(s)$ has several rows. $\boldsymbol{Q}(s)$ is then used to select appropriate rows in $\boldsymbol{F}(s)$. To explain the idea, let $\boldsymbol{F}_i(s), i = 1, \cdots, n_r$ denote the i^{th} row of $\boldsymbol{F}(s)$, and consider the scalar residuals

$$r_i(s) = \frac{\boldsymbol{F}_i(s)}{p(s)} \begin{pmatrix}\boldsymbol{y}(s) \\ \boldsymbol{u}(s)\end{pmatrix} \qquad i = 1, \cdots, n_r.$$

[4] The notion of rank considered here is the normal rank computed as $\max_s \text{ rank } H(s)$ where the maximum is taken over all complex values of s.

By performing a simulation of all these filters with actual plant measurements as input, one may compare how significantly the actual residuals $r_i(t)$, $i = 1, ..., n_r$ deviate from zero in the absence of fault, once the transient due to initial conditions has vanished. This reflects the effect of modelling errors on the residuals. Besides, by using faulty data obtained with a simulation or corresponding to actual plant measurements it is also possible to compare the actual sensitivities to faults. A kind of "signal to noise ratio" could be defined for each residual as

$$SNR_i = \frac{\int_{t_0}^{t_0+T} r_i^F(t)^2 dt}{\int_{t_1}^{t_1+T} r_i^{FF}(t)^2 dt}, \tag{6.51}$$

where $r_i^F(t)$ denotes the residual obtained with the measurement associated to the faulty mode, and r_i^{FF} corresponds to the fault free situation. T is a user defined horizon, t_0 and t_1 are time instants associated to faulty and fault free data sequences. Matrix $\boldsymbol{Q}(s)$ should then be chosen to select the components of $\boldsymbol{r}(s)$ for which the "signal-to-noise ratio" is significantly larger than 1.

Computational aspects. The problem of finding an irreducible polynomial basis for $\mathcal{N}_L(\boldsymbol{H}(s))$ can be transformed into the determination of a similar basis for a polynomial matrix instead of the rational matrix $\boldsymbol{H}(s)$. It suffices to notice that

$$\boldsymbol{H}(s) = \bar{\boldsymbol{H}}(s)/h(s),$$

where $h(s)$ is the least common multiple of all denominators. An irreducible polynomial basis for $\bar{\boldsymbol{H}}(s)$ is also an irreducible polynomial basis for $\boldsymbol{H}(s)$, and vice-versa. Numerically stable algorithms for the computation of an irreducible polynomial basis are available in the literature, and they have been programmed in the polynomial toolbox of MATLAB.

The symbolic tools Maple and Mathematica can calculate a basis for the left nullspace of $\boldsymbol{H}(s)$. The Maple command *nullspace basis* applied to the matrix $\boldsymbol{H}'(s)$ will provide the row basis given in analytical form. Calculation of a basis is not unique, so the result can be expanded or reduced by a polynomial fraction as desired. The result is not necessarily irreducible, either, but the reduction to an irreducible basis is usually straightforward once a factorisation is made of the entries in the nullspace basis[5].

Example 6.4 (cont.) *Parity relations for the ship*

A model of the form (6.40) can be easily deduced from the linear state-space model for the ship example. The following transfer matrices are obtained when sensor faults are considered, and when state and sensor noise are neglected:

$$\boldsymbol{H}_{yu}(s) = \begin{pmatrix} \frac{b}{s-b\eta_1} \\ \frac{b}{(s-b\eta_1)s} \end{pmatrix}, \quad \boldsymbol{H}_{yd}(s) = \begin{pmatrix} 1 \\ \frac{1}{s} \end{pmatrix}, \quad \boldsymbol{H}_{yx}(s) = \begin{pmatrix} \frac{1}{s-b\eta_1} & 0 \\ \frac{1}{s(s-b\eta_1)} & \frac{1}{s} \end{pmatrix}$$

[5] The Maple symbolic mathematics engine is a stand-alone product. It is also a part of the MATLAB Symbolic Toolbox. MATLAB®, Maple® and Mathematica® are registered trademarks of their respective owners.

and $\boldsymbol{H}_{yf} = \boldsymbol{I}_2$. In the above expressions,

$$\begin{array}{rcl} \boldsymbol{x}(t) & = & (\omega_3(t) \quad \psi(t))' \\ \boldsymbol{y}(t) & = & (\omega_{3m}(t) \quad \psi_m(t))' \\ d(t) & = & \omega_w(t) \\ u(t) & = & \delta(t) \\ \boldsymbol{f}(t) & = & (f_\omega(t) \quad f_\psi(t))' \end{array}$$

hold. It is assumed that η_1 is negative, so that the ship is stable. $\boldsymbol{H}_{yx}(s)$ is not asymptotically stable however, due to the integrator linking speed and position. We shall see below what slight modification must be introduced in the theory to handle the pole at the origin.

The matrix $\boldsymbol{H}(s)$ takes the form:

$$\boldsymbol{H}(s) = \begin{pmatrix} \frac{b}{s-b\eta_1} & 1 \\ \frac{b}{s(s-b\eta_1)} & \frac{1}{s} \\ 1 & 0 \end{pmatrix} = \frac{1}{s(s-b\eta_1)} \begin{pmatrix} bs & s(s-b\eta_1) \\ b & (s-b\eta_1) \\ s(s-b\eta_1) & 0 \end{pmatrix}.$$

The last matrix corresponds to $\bar{\boldsymbol{H}}(s)$. An irreducible basis for its left nullspace can be calculated, or found by inspection, to be

$$F(s) = (1 \quad -s \quad 0).$$

Thus, any vector of rational functions of the form

$$\begin{pmatrix} \frac{q(s)}{p(s)} & \frac{-sq(s)}{p(s)} & 0 \end{pmatrix},$$

where $p(s)$ is an arbitrary polynomial with roots in the left half plane and $q(s)$ is an arbitrary polynomial with degree less than $p(s)$, fulfils condition (6.45). Candidate residual generators have the form:

$$r(s) = \frac{q(s)}{p(s)}\omega_{3m}(s) - \frac{sq(s)}{p(s)}\psi_m(s). \tag{6.52}$$

Notice that, by setting $q(s) = 1$, one recovers (6.38) with $p(s) = s + a$.

Substituting the model equations for $\omega_3(s)$ and $\psi(s)$ yields

$$\begin{array}{rcl} r(s) & = & \frac{q(s)}{p(s)}\left(\frac{b}{s-b\eta_1}\delta(s) + \omega_w(s) + \frac{1}{s-b\eta_1}\omega_3(0) + f_\omega(s)\right) \\ & & -\frac{sq(s)}{p(s)}\left(\frac{b}{s(s-b\eta_1)}\delta(s) + \frac{\omega_w(s)}{s} + \frac{1}{s(s-b\eta_1)}\omega_3(0) + \frac{1}{s}\psi(0) + f_\psi(s)\right) \\ & = & -\frac{q(s)}{p(s)}\psi(0) + \frac{q(s)}{p(s)}f_\omega(s) - \frac{sq(s)}{p(s)}f_\psi(s). \end{array}$$

In order to assure that the residual asymptotically vanishes, two solutions are possible:

- Introduction of a derivative action in $q(s)$, so that $q(s) = s\bar{q}(s)$ and the term involving $\psi(0)$ in the above equation is null at steady state.
- Modification of (6.52) by adding a correction term associated with the initial position (supposed to be measured correctly). This yields

  $$r(s) = \frac{q(s)}{p(s)}\psi(0) + \frac{q(s)}{p(s)}\omega_{3m}(s) - \frac{sq(s)}{p(s)}\psi_m(s)$$

  or, after substitution of $\omega_{3m}(s)$ and $\psi_m(s)$ in terms of the model equations:

$$r(s) = \frac{q(s)}{p(s)} f_\omega(s) - \frac{sq(s)}{p(s)} f_\psi(s). \tag{6.53}$$

The first solution also introduces a derivative action in the transfer functions between $f_\omega(s)$ and $r(s)$, and between $f_\psi(s)$ and $r(s)$. Hence step like faults do not have any steady state effect on the residual. On the other hand, in (6.53), $q(s)$ can be chosen so that a step-like fault f_ω has a steady state effect on r, but a step like fault in f_ψ can only influence temporarily r due to the zero at the origin in $\frac{sq(s)}{p(s)}$. Application of the theory below will indicate that, indeed, fault f_ω is strongly detectable, but f_ψ is only weakly detectable. □

Example 6.5 *Parity relations – ship with three output measurements*

Some useful observations can be made later from the above example but using an additional instrument to measure the ship heading. This third instrument is taken to be independent of the other two. This is a realistic case since redundant heading instruments are required for most merchant ships.

With two independent heading angle measurements

$$y_2(s) = \psi_m^{(1)}(s)$$

and

$$y_3(s) = \psi_m^{(2)}(s),$$

the matrix $\boldsymbol{H}(s)$ takes the form:

$$\boldsymbol{H}(s) = \frac{1}{s(s - b\eta_1)} \begin{pmatrix} bs & s(s - b\eta_1) \\ b & (s - b\eta_1) \\ b & (s - b\eta_1) \\ s(s - b\eta_1) & 0 \end{pmatrix}.$$

The nullspace basis for $\boldsymbol{H}(s)$ is computed to be

$$\begin{pmatrix} (\frac{-1}{s} & 1 & 0 & 0) \\ (\frac{-1}{s} & 0 & 1 & 0) \end{pmatrix}.$$

This means a family of candidate residual generators exist, which have the form

$$\boldsymbol{r}(s) = \frac{q(s)}{p(s)} \begin{pmatrix} \frac{-1}{s} & 1 & 0 & 0 \\ \frac{-1}{s} & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \omega_{3m}(s) \\ \psi_m^{(1)}(s) \\ \psi_m^{(2)}(\delta) \\ \delta(s) \end{pmatrix}.$$

The relation between components of the residual vector $\boldsymbol{r}(s)$ to faults $\boldsymbol{f}(s)$ and wave disturbance ω_w is

$$\begin{aligned} r_1(s) &= \frac{q(s)}{p(s)} \left(-\frac{1}{s} f_\omega(s) + f_\psi^{(1)}(s) \right) \\ r_2(s) &= \frac{q(s)}{p(s)} \left(-\frac{1}{s} f_\omega(s) + f_\psi^{(2)}(s) \right). \end{aligned}$$

It is evident that all elements of the residual are decoupled from the wave disturbance, which was the intention.

Forming a third residual using the plain difference between heading angle measurements

$$r_3(s) = \psi_m^{(1)}(s) - \psi_m^{(2)}(s),$$

which would be a straightforward choice as an output parity equation, is indeed possible, but since this would be a linear relation between the two residuals already defined, this would not add to the information contained in the residual vector. □

Fault detectability. To deduce theoretical results on fault detectability, the expression of the residual in the presence of faults must be determined. Substituting (6.45) into (6.43) yields

$$\begin{aligned} \boldsymbol{r}(s) &= \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yx}(s)\boldsymbol{x}(0) + \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yf}(s)\boldsymbol{f}(s) \\ &= \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yx}(s)\boldsymbol{x}(0) + \sum_{i=1}^{n_f}\boldsymbol{V}_{ry}(s)\boldsymbol{H}^i_{yf}(s)f_i(s), \end{aligned} \tag{6.54}$$

where $\boldsymbol{H}^i_{yf}(s)$ denotes the i^{th} column of $\boldsymbol{H}_{yf}(s)$. It can be shown that a necessary and sufficient condition for detectability of the i^{th} fault is:

$$\boldsymbol{V}_{ry}(s)\boldsymbol{H}^i_{yf}(s) \neq 0, \tag{6.55}$$

where $\boldsymbol{V}_{ry}(s)$ also fulfils

$$\boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yd}(s) = 0. \tag{6.56}$$

The latter condition comes from the second entry in (6.44). For (6.55) and (6.56) to be simultaneously verified, one should not be able to express $\boldsymbol{H}^i_{yf}(s)$ as a linear combination of the columns of $\boldsymbol{H}_{yd}(s)$. In other words, there cannot exists any non-zero polynomial set $\alpha_0(s)$, $\alpha_1(s)$, $\cdots$, $\alpha_{n_d}(s)$ such that:

$$\alpha_0(s)\boldsymbol{H}^i_{yf}(s) + \alpha_1(s)\boldsymbol{H}^1_{yd}(s) + \cdots + \alpha_{n_d}(s)\boldsymbol{H}^{n_d}_{yd}(s) = 0$$

This condition is fulfilled when

$$\text{rank}\ \begin{pmatrix}\boldsymbol{H}_{yd}(s) & \boldsymbol{H}^i_{yf}(s)\end{pmatrix} > \text{rank}\ \boldsymbol{H}_{yd}(s), \tag{6.57}$$

where

$$\text{rank}\ \boldsymbol{A}(s) = \max_s\ \text{rank}\ \boldsymbol{A}(s)$$

denotes the normal rank of the rational matrix $\boldsymbol{A}(s)$. In the latter expression, the "rank"-operation in the right hand side acts on a matrix of complex numbers obtained for a specific value of s. It can thus be evaluated in the standard way. (6.57) is actually a necessary and sufficient condition for the i^{th} fault to be weakly detectable.

To determine a test for strong fault detectability, substitute the model (6.40) for $y(s)$ in the parametrisation of the class of residual generators (6.50)

$$\begin{aligned} \boldsymbol{r}(s) &= \frac{\boldsymbol{Q}(s)\boldsymbol{F}(s)}{p(s)}\begin{pmatrix}\boldsymbol{H}_{yu}(s) & \boldsymbol{H}_{yd}(s) & \boldsymbol{H}_{yf}(s)\\ \boldsymbol{I} & \boldsymbol{O} & \boldsymbol{O}\end{pmatrix}\begin{pmatrix}\boldsymbol{u}(s)\\ \boldsymbol{d}(s)\\ \boldsymbol{f}(s)\end{pmatrix} \\ &= \frac{\boldsymbol{Q}(s)\boldsymbol{F}(s)}{p(s)}\begin{pmatrix}\boldsymbol{H}_{yf}(s)\\ \boldsymbol{O}\end{pmatrix}\boldsymbol{f}(s), \end{aligned} \tag{6.58}$$

where the second equality accounts for the fact that $\boldsymbol{F}(s)$ is a basis for the left nullspace of $\boldsymbol{H}(s)$. The transient term due to $\boldsymbol{x}(0)$ was not considered as its effect vanishes when t tends to infinity. Strong detectability of the fault f_i is thus achieved if there exists some polynomial $p(s)$ and polynomial matrix $\boldsymbol{Q}(s)$ such that

$$\frac{\boldsymbol{Q}(s)\boldsymbol{F}(s)}{p(s)} \left(\begin{array}{c} \boldsymbol{H}_{yf}^{i}(s) \\ \boldsymbol{O} \end{array} \right) \Bigg|_{s=0} \neq 0. \tag{6.59}$$

As $p(0)$ is necessarily chosen non-zero to assure asymptotic stability of the filter, and $\boldsymbol{Q}(s)$ can be chosen arbitrarily, a necessary and sufficient condition for strong fault detectability is

$$\boldsymbol{F}(s) \left(\begin{array}{c} \boldsymbol{H}_{yf}^{i}(s) \\ \boldsymbol{O} \end{array} \right) \Bigg|_{s=0} \neq 0. \tag{6.60}$$

Notice that this expression may be different from $\boldsymbol{F}(0) \left(\begin{array}{c} \boldsymbol{H}_{yf}^{i}(0) \\ \boldsymbol{O} \end{array} \right)'$ and, hence, substitution by $s = 0$ must be performed after computation of the matrix product.

Example 6.6 *Detectability - ship with two output measurements*

To check that fault $f_{\psi}^{(1)}$ is detectable, (6.57) is applied as follows

$$\text{rank} \begin{pmatrix} 1 & 1 \\ \frac{1}{s} & 0 \end{pmatrix} > \text{rank} \begin{pmatrix} 1 \\ \frac{1}{s} \end{pmatrix}.$$

Similarly, the inequality

$$\text{rank} \begin{pmatrix} 1 & 0 \\ \frac{1}{s} & 1 \end{pmatrix} > \text{rank} \begin{pmatrix} 1 \\ \frac{1}{s} \end{pmatrix}$$

ensures that $f_{\psi}^{(2)}$ is detectable. Condition (6.60) is now used to check strong fault detectability. For fault $f_{\psi}^{(1)}$ it yields

$$\begin{pmatrix} 1 & -s & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \Bigg|_{s=0} = 1.$$

Thus fault $f_{\psi}^{(1)}$ is strongly detectable. For fault $f_{\psi}^{(2)}$ one gets

$$\begin{pmatrix} 1 & -s & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \Bigg|_{s=0} = 0,$$

which indicates that $f_{\psi}^{(2)}$ is not strongly detectable, as was expected. □

The procedure for residual generator design can be summarised as follows.

Algorithm 6.2 *Residual generator design with the parity space method*

Given: A model of the supervised system in the form (6.40).

Computation:

1. Compute matrix $\boldsymbol{H}(s)$ as defined by (6.48).
2. Determine an irreducible polynomial basis for $\mathcal{N}_L(\boldsymbol{H}(s))$, and let $\boldsymbol{F}(s)$ be the matrix whose rows make such a basis. If $\boldsymbol{F}(s) = \boldsymbol{O}$, the problem has no solution.
3. Design the filter $\frac{\boldsymbol{Q}(s)}{p(s)}$ as a low-pass or a band-pass filter which possibly selects appropriate rows in $\boldsymbol{F}(s)$ according to $SNR_i, i = 1, \cdots, \beta$ (cf. (6.51)).
4. Check for weak or strong fault detectability as needed.

Result: A residual generator in the form (6.50).

6.4.3 Fault isolation

For fault-tolerant control, faults should not only be detected, but also be isolated, namely the faulty components should be determined. The problem of residual generator design for fault detection and isolation based on a deterministic model can be stated as follows.

Consider a system described by a continuous-time linear state-space model of the form

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \sum_{j=1}^{n_f} \boldsymbol{F}_x^j \boldsymbol{f}_j(t), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) + \sum_{j=1}^{n_f} \boldsymbol{F}_y^j \boldsymbol{f}_j(t), \end{aligned} \tag{6.61}$$

where $\boldsymbol{f}_j \in \mathbb{R}^{n_{f_j}}, j = 1, \ldots, n_f$ represent the faults that must be detected and isolated. In terms of transfer functions, (6.61) can be written as

$$\boldsymbol{y}(s) = \boldsymbol{H}_{yu}(s)\boldsymbol{u}(s) + \boldsymbol{H}_{yx}(s)\boldsymbol{x}(0) + \sum_{j=1}^{n_f} \boldsymbol{H}_{yf_j}(s)\boldsymbol{f}_j(s), \tag{6.62}$$

where

$$\boldsymbol{H}_{yu}(s) = \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} + \boldsymbol{D},\ \boldsymbol{H}_{yx}(s) = \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}$$

and

$$\boldsymbol{H}_{yf_j}(s) = \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{F}_x^j + \boldsymbol{F}_y^j.$$

Problem 6.3 (Residual generator design for fault detection and isolation based on a deterministic model)
Given a model of the supervised process of the form (6.61) or (6.62), determine a set of n_f stable linear time-invariant filters described by

$$\begin{aligned} \dot{\boldsymbol{z}}_\ell(t) &= \boldsymbol{A}_{z,\ell}\boldsymbol{z}_\ell(t) + \boldsymbol{B}_{zu,\ell}\boldsymbol{u}(t) + \boldsymbol{B}_{zy,\ell}\boldsymbol{y}(t), \quad \boldsymbol{z}_\ell(0) = \boldsymbol{z}_{0,\ell} \\ \boldsymbol{r}_\ell(t) &= \boldsymbol{C}_{rz,\ell}\boldsymbol{z}_\ell(t) + \boldsymbol{D}_{ru,\ell}\boldsymbol{u}(t) + \boldsymbol{D}_{ry,\ell}\boldsymbol{y}(t), \quad \ell = 1, \ldots, n_f \end{aligned} \tag{6.63}$$

or, in transfer function form, assuming zero initial conditions,

$$\boldsymbol{r}_\ell(s) = \boldsymbol{V}_{ru,\ell}(s)\boldsymbol{u}(s) + \boldsymbol{V}_{ry,\ell}(s)\boldsymbol{y}(s), \qquad \ell = 1, \ldots, n_f, \tag{6.64}$$

such that the following conditions are met.

- *$\boldsymbol{r}_\ell(t)$ asymptotically decays to zero for any $\boldsymbol{u}(t)$ and any $\boldsymbol{f}_j(t),\ j = 1, \ldots, n_f,\ j \neq \ell,\ t > 0$.*
- *$r_\ell(t)$ is affected by $f_\ell(t)$.*

In this problem statement, the ℓ^{th} residual can only be affected by the ℓ^{th} fault, and not by the others. The table below represents this situation when $n_f = 3$.

Table 6.1 Effects of the faults on the residuals

$\nearrow$	$\boldsymbol{f}_1$	$\boldsymbol{f}_2$	$\boldsymbol{f}_3$
$\boldsymbol{r}_1$	$\times$	0	0
$\boldsymbol{r}_2$	0	$\times$	0
$\boldsymbol{r}_3$	0	0	$\times$

A "$\times$" in Table 6.1 indicates that the fault in the corresponding column affects the residual of the corresponding row.

The faults that do not affect the ℓ^{th} residual can be seen as unknown inputs to which this residual should not be sensitive. Hence, to design a residual generator that output r_ℓ, it suffices to use the solution of the problem of residual generation for fault detection in which vector $\boldsymbol{d}$ is replaced by $(\boldsymbol{f}_1' \ldots \boldsymbol{f}_{\ell-1}'\ \boldsymbol{f}_{\ell+1}' \ldots \boldsymbol{f}_{n_f}')'$. n_f such problems should be solved for $\ell = 1, \ldots, n_f$ in order to obtain the n_f filters that make a solution to the fault isolation problem.

From the conditions for fault detectability, the following conditions can be deduced for the above scheme to work:

$$\begin{aligned} &\text{rank } \left(\boldsymbol{H}_{yf_\ell}(s)\ \ \boldsymbol{H}_{yf_j}(s)\right) > \text{ rank } \boldsymbol{H}_{yf_j}(s) \\ &\text{for all } \ell,\ j = 1, \ldots, n_f,\ \ell \neq j. \end{aligned} \tag{6.65}$$

A suficient condition for (6.65) to hold is

$$\sum_{\substack{j=1,n_f \\ j \neq \ell}} n_{f_j} < p, \tag{6.66}$$

where p is the number of measured output signals (dimension of $\boldsymbol{y}$).

When condition, (6.65) is not met, the diagonal structure of Table 6.1 cannot be obtained, and one should attempt to group the fault vectors in different classes and to generate residuals that are affected by a specific fault class and not by the others. The table below illustrates one way to perform such a grouping, in a situation where $n_f = 3$ and two residual generators are designed.

Table 6.2 Effects of the faults on the residuals – non-diagonal structure

$\nearrow$	$\boldsymbol{f}_1$	$\boldsymbol{f}_2$	$\boldsymbol{f}_3$
r_1	$\times$	$\times$	0
r_2	$\times$	0	$\times$

In the situation of Table 6.2, all three faults can be distinguished as the combination of r_1 and r_2 reacts differently to each fault. However, simultaneous faults cannot be isolated because they affect both residuals in all cases.

Example 6.7 *Isolability - ship with three output measurements*

For the ship with one rate measurement and two heading measurements (Example 6.5), a residual generator is achieved, which was decoupled from the disturbance,

$$\begin{pmatrix} r_1(s) \\ r_2(s) \end{pmatrix} = \begin{pmatrix} -\frac{1}{s} & 1 & 0 \\ -\frac{1}{s} & 0 & 1 \end{pmatrix} \begin{pmatrix} f_{\omega 3}(s) \\ f_{\psi}^{(1)}(s) \\ f_{\psi}^{(2)}(s) \end{pmatrix}. \tag{6.67}$$

This residual generator has the properties shown in Table 6.2. □

Sensor fault isolation in a fault-tolerant control setting. If it has been detected that one out of a set of faults is present, but it has not been possible to isolate which fault is actually present, and this was due to the design of the residual generator specification, alternatives are available on the fault-tolerant setting because the supervisory system has control of the input signals to the plant. Similar to system identification, where a dedicated test signal is applied to obtain the optimal information about a particular parameter, a dedicated test signal can be applied on the control input to help confirm particular hypotheses. This procedure can help reduce time to diagnose and therefore time to reconfigure a controller.

Example 6.8 *Dedicated test signal for isolation – ship steering*

If two identical rate sensors are available in the ship steering example, and the residual generator was designed to be insensitive to the wave disturbance, it is not possible to isolate faults

f_ω^1 and f_ω^2. In a fault-tolerant control setting, employ active test signal generation to isolate the fault once it has been detected that one of the rate sensor units is defect. Define a dedicated test signal

$$\delta(t) = \tilde{\delta}(t),\ t \subset [0,\ T],$$

which is applied immediately after the hypothesis of

$$\left\{ \hat{f}_\omega^1(t) \vee \hat{f}_\omega^2(t) \right\} \neq 0$$

is confirmed. Observe a-priori the response in the non-faulty condition

$$\omega_3^{rec}(t) = g_\omega\left(\tilde{\delta}(t),\ U(t)\right),\ t \subset [0, T]$$

note that the function g_{ω_3} is not calculated, the angular rate is merely recorded and stored. Calculate the correlations

$$\begin{aligned} cor_{31}(t) &= \frac{1}{t}\int_0^t \omega_3^{rec}(\tau)\omega_{3m}^1(\tau)d\tau \\ cor_{21}(t) &= \frac{1}{t}\int_0^t \omega_{3m}^2(\tau)\omega_{3m}^1(\tau)d\tau \\ cor_{32}(t) &= \frac{1}{t}\int_0^t \omega_3^{rec}(\tau)\omega_{3m}^2(\tau)d\tau. \end{aligned}$$

These correlation signals with appropriate normalisation make it straightforward to determine which hypothesis is the most likely. □

6.4.4 Fault estimation

The isolation schemes signify which fault is present but do not assess the magnitude of the fault. Fault estimates are needed in certain fault accommodation approaches as was indicated in Section 6.1. This notion is defined as follows.

Definition 6.3 (Fault estimation)
Fault estimation is the ability to estimate the magnitude of a fault $\boldsymbol{f}_i(t)$ *and its time history.*

Combining (6.42) and (6.58), the link between the fault vector $\boldsymbol{f}(s)$ and the residual $\boldsymbol{r}(s)$ is seen to be

$$\boldsymbol{r}(s) = \boldsymbol{V}_{ru}(s)\boldsymbol{u}(s) + \boldsymbol{V}_{ry}(s)\boldsymbol{y}(s) = \frac{\boldsymbol{Q}(s)\boldsymbol{F}(s)}{p(s)} \begin{pmatrix} \boldsymbol{H}_{yf}(s) \\ \boldsymbol{O} \end{pmatrix} \boldsymbol{f}(s), \quad (6.68)$$

where it is assumed that initial conditions have vanished. Letting

$$\boldsymbol{F}(s) = \left(\boldsymbol{F}_1(s) \quad \boldsymbol{F}_2(s)\right),$$

where $\boldsymbol{F}_1(s)$ has p columns and $\boldsymbol{F}_2(s)$ has m columns, Eq. (6.68) can be written

$$\boldsymbol{r}(s) = \boldsymbol{V}_{ru}(s)\boldsymbol{u}(s) + \boldsymbol{V}_{ry}(s)\boldsymbol{y}(s) = \frac{\boldsymbol{Q}(s)\boldsymbol{F}_1(s)}{p(s)} \boldsymbol{H}_{yf}(s)\boldsymbol{f}(s). \quad (6.69)$$

On the other hand, Eq. (6.54) yields the following relation when the transient due to the inital conditions is neglected

$$\boldsymbol{r}(s) = \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yf}(s)\boldsymbol{f}(s),$$

hence

$$\boldsymbol{V}_{ry}(s) := \frac{\boldsymbol{Q}(s)\boldsymbol{F}_1(s)}{p(s)}. \tag{6.70}$$

As a compact notation, introduce $\boldsymbol{H}_{rf}(s)$ by

$$\boldsymbol{H}_{rf}(s) := \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yf}(s) = \frac{\boldsymbol{Q}(s)\boldsymbol{F}_1(s)}{p(s)}\boldsymbol{H}_{yf}(s).$$

If it is possible to determine a suitable left inverse to $\boldsymbol{H}_{rf}(s)$, say $\boldsymbol{G}(s)$, an estimate of $\boldsymbol{f}(s)$ would be

$$\hat{\boldsymbol{f}}(s) = \boldsymbol{G}(s)\boldsymbol{r}(s) = \boldsymbol{G}(s)(\boldsymbol{V}_{ru}(s)\boldsymbol{u}(s) + \boldsymbol{V}_{ry}(s)\boldsymbol{y}(s)). \tag{6.71}$$

Left inverse transformation. If the polynomial matrix $\boldsymbol{H}_{rf}(s)$ is square, then the estimate $\hat{\boldsymbol{f}}(s) = \boldsymbol{H}_{rf}^{-1}(s)$ where the ij-th element of $\boldsymbol{H}_{rf}^{-1}$, call it h_{ij} is the usual inverse

$$h_{ij}(s) = \frac{1}{\det(\boldsymbol{H}_{rf}(s))}(-1)^{i+j}(\boldsymbol{M}_{ji}(s)),$$

where $\boldsymbol{M}_{ji}(s)$ is the determinant of the matrix formed by $\boldsymbol{H}_{rf}(s)$ after deleting row j and column i.

If $\boldsymbol{H}_{rf}(s)$ is non-square, with l rows and n_f columns, then, there exists a left pseudo-inverse $\boldsymbol{G}(s)$ of $\boldsymbol{H}_{rf}(s)$ if and only if

$$\text{rank}\,(\boldsymbol{H}_{rf}(s)) = n_f,$$

where the normal rank is considered. $\boldsymbol{G}(s)$ is given as

$$\boldsymbol{G}(s) = (\boldsymbol{H}'_{rf}(s)\boldsymbol{H}_{rf}(s))^{-1}\boldsymbol{H}'_{rf}(s). \tag{6.72}$$

The pseudo-inverse has the property

$$\boldsymbol{G}(s)\boldsymbol{H}_{rf}(s) = \boldsymbol{I}_{n_f}$$

with $\boldsymbol{I}_{n_f}$ being the unity matrix of dimension n_f.

Remark 6.5 *Causality of solution*
To be able to implement the filter Eq. (6.71), $\boldsymbol{G}(s)\,\boldsymbol{V}_{ru}(s)$ and $\boldsymbol{G}(s)\,\boldsymbol{V}_{ry}(s)$ must be proper and stable transfer functions. This may not be true when $\boldsymbol{G}(s)$ is computed as above. A modified procedure can be found in the literature (see the bibliographical notes for this chapter). □

Fault estimation after isolation. A necessary condition to be able to compute the above rational estimate, based on a pseudo inverse transformation, is that the rank

of the $\boldsymbol{H}_{rf}$ matrix is equal to the number of faults to be estimated. As the number of faults is often larger than the number of independent residuals, it is necessary to take advantage of the results of the fault isolation to limit estimation of faults to those that the isolation algorithm found to be present in the system.

Assume the subset of the fault vector $\boldsymbol{f}_i$, $i \in [j, ..., k]$ has been determined necessary to estimate by the isolation algorithm. The above general expressions then hold for the entries of the transfer function matrices that relate to f_i, $i \in [j, ..., k]$.

Assume a single fault has been determined present. Then, a single column in $\boldsymbol{H}_{rf}(s)$ needs to be considered. The result for this simplest case can be formulated as follows.

Given the stable residual generator

$$\boldsymbol{r}(s) = \boldsymbol{V}_{ru}(s)\boldsymbol{u}(s) + \boldsymbol{V}_{ry}(s)\boldsymbol{y}(s))$$

and a transfer function model relating this residual to faults

$$\boldsymbol{r}(s) = \boldsymbol{H}_{rf}(s)\boldsymbol{f}(s).$$

Assume that the isolation procedure indicates that fault number i is present, and let the i^{th} column of $\boldsymbol{H}_{rf}(s)$ be

$$\boldsymbol{h}_i(s) = \frac{\bar{\boldsymbol{h}}_i(s)}{\boldsymbol{\eta}(s)},$$

where $\bar{\boldsymbol{h}}_i(s)$ is a polynomial vector with entries $\bar{\boldsymbol{h}}_{ji}(s)$ and $\boldsymbol{\eta}(s)$ is the least common denomination of the entries of $\boldsymbol{h}_i(s)$.

Theorem 6.1 (Single fault estimation)

On the condition that $\boldsymbol{\eta}(s)$ *and* $\tilde{\boldsymbol{h}}_i(s) = \sum_{j=1}^{l} \bar{\boldsymbol{h}}_{ji}(s)^2$ *are stable polynomials, an estimate of* $\boldsymbol{f}_i$, $\hat{\boldsymbol{f}}_i$ *is given by:*

$$\hat{\boldsymbol{f}}_i(s) = \left(\boldsymbol{h}_i'(s)\,\boldsymbol{h}_i(s)\right)^{-1}\ \boldsymbol{h}_i(s)'\boldsymbol{r}(s). \tag{6.73}$$

This estimator is causal when $\deg \boldsymbol{\eta}(s) = \max\ \deg\ \bar{\boldsymbol{h}}_{ji}(s)$. This is easily proved by direct computation of the pseudo inverse in Eq. (6.73):

$$\left(\boldsymbol{h}_i(s)'\,\boldsymbol{h}_i(s)\right)^{-1}\ \boldsymbol{h}_i(s)' = \frac{\eta(s)}{\sum_{i=1}^{l} \bar{\boldsymbol{h}}_{ji}^2(s)}\ \left(\bar{\boldsymbol{h}}_{1i}(s),\ ...,\ \bar{\boldsymbol{h}}_{li}(s)\right). \tag{6.74}$$

If the above condition on the degree is not met, a low pass approximation for the fault estimate can be obtained by multiplying the denomination of Eq. (6.74) by $(\boldsymbol{s}+\alpha)^{\beta}$, where $\alpha \in \mathbb{R}^+$and β is chosen so that all entries in Eq. (6.74) are causal.

Example 6.9 *Fault estimation - ship with three output measurements*

Fault estimation following isolation for the ship with three output measurements results from the residual generator obtained in Example 6.7

$$\boldsymbol{H}_{rf}(s) = \begin{pmatrix} -\frac{1}{s} & 1 & 0 \\ -\frac{1}{s} & 0 & 1 \end{pmatrix} = \frac{1}{s}\begin{pmatrix} -1 & s & 0 \\ -1 & 0 & s \end{pmatrix}.$$

Fault 1 isolated: The estimate of fault number 1 is

$$\hat{f}_1 = \frac{s}{2}\left(\boldsymbol{r}_1(s) + \boldsymbol{r}_2(s)\right).$$

Since this filter is not causal, a low-pass filtered approximation for the rate gyro fault is needed, where $\alpha \in \mathbb{R}^+$

$$\hat{f}_1 = \frac{s}{2(s+\alpha)}\left(\boldsymbol{r}_1(s) + \boldsymbol{r}_2(s)\right). \tag{6.75}$$

Fault 2 isolated: The estimate of fault number 2 is

$$\hat{\boldsymbol{f}}_2 = \boldsymbol{r}_1(s).$$

Fault 2 isolated: The estimate of fault number 3 is

$$\hat{\boldsymbol{f}}_3 = \boldsymbol{r}_2(s).$$

It should be noted that an erroneous isolation will give gross errors in the fault estimate.

In an implementation, the above fault estimators would run in parallel. Once a particular fault is isolated, the estimate can be rapidly provided. □

Alternative methods to fault estimation. In cases, where the above algebraic approach to fault estimation fails, asymptotic estimation of faults may be achievable using an observer on an augmented system, where the state is augmented by the fault(s) to be estimated (modelling faults to be constant):

$$\begin{aligned}
\frac{d}{dt}\begin{pmatrix}\boldsymbol{x}\\ \boldsymbol{f}\end{pmatrix} &= \begin{pmatrix}\boldsymbol{A} & \boldsymbol{F}_x\\ \boldsymbol{O} & \boldsymbol{O}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}\\ \boldsymbol{f}\end{pmatrix} + \begin{pmatrix}\boldsymbol{B}\\ 0\end{pmatrix}\boldsymbol{u}(t)\\
\boldsymbol{y}(t) &= \begin{pmatrix}\boldsymbol{C} & \boldsymbol{F}_y\end{pmatrix}\begin{pmatrix}\boldsymbol{x}\\ \boldsymbol{f}\end{pmatrix}.
\end{aligned}$$

A necessary condition for an asymptotically stable estimator to exist is that the pair

$$\left(\begin{pmatrix}\boldsymbol{A} & \boldsymbol{F}_x\\ \boldsymbol{O} & \boldsymbol{O}\end{pmatrix}, \begin{pmatrix}\boldsymbol{C} & \boldsymbol{F}_y\end{pmatrix}\right)$$

is observable. Observer-based methods are covered extensively in the literature (see the bibliographical notes for references).

In summary the procedure for estimating the magnitude of a fault is as follows:

Algorithm 6.3 *Fault estimation*

Given: A model of the supervised process of the form (6.40) and a residual generator of the form (6.42)

Compute:

1. The transfer matrix $\boldsymbol{H}_{rf}(s)$ relating residuals to faults
2. A left inverse to $\boldsymbol{H}_{rf}(s)$
3. An estimator of the form (6.71), possibly after appropriate filtering of the left inverse in order to obtain a causal and stable estimator for all faults.

Result: A causal and stable fault estimator based on the measurements of $\boldsymbol{u}(s)$ and $\boldsymbol{y}(s)$.

6.5 Deterministic model – optimisation-based approach

6.5.1 Problem statement

The above methods were based on algebraic or polynomial manipulations, and relied on the ability to achieve exact decoupling from disturbances and from input to the residual. When this is not possible, the influence $\boldsymbol{d}(t)$ and $\boldsymbol{u}(t)$ have on the residual competes with that generated by faults $\boldsymbol{f}(t)$. If the effects of input and disturbance on the residual are non-zero, we do not obtain

$$(\boldsymbol{V}_{ru}(s) + \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yu}(s) \quad \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yd}(s)) \begin{pmatrix} \boldsymbol{u}(s) \\ \boldsymbol{d}(s) \end{pmatrix} = 0 \tag{6.76}$$

for all $\boldsymbol{u}(s)$ and $\boldsymbol{d}(s)$ and

$$\begin{aligned} \boldsymbol{r}(s) \quad = \quad & (\boldsymbol{V}_{ru}(s) + \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yu}(s) \quad \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yd}(s)) \begin{pmatrix} \boldsymbol{u}(s) \\ \boldsymbol{d}(s) \end{pmatrix} \\ & + \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yf}(s)\boldsymbol{f}(s) + \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yx}(s)\boldsymbol{x}(0) \end{aligned} \tag{6.77}$$

is strictly speaking not a residual generator according to the definition.

The purpose of this section is to find ways to relax the requirement on exact decoupling for the residual generator. Instead, some optimal approximation should be obtained in the sense that the design shall satisfy certain criteria.

The design objectives should be to

1. Provide a sufficient suppression of disturbances $\boldsymbol{d}$ seen from the residual,

2. Maximise the sensitivity $\boldsymbol{r}$ of the residual with respect to all or a to a selected set of faults in $\boldsymbol{f}$.
3. Make the residual signal sufficiently insensitive to variations in the input signal $\boldsymbol{u}$.
4. Provide the designer with tools to enter a specification of the desired performance.

Formulating the design objectives as performance indices will enable a rigorous treatment. From the condition Eq. (6.76), perfect decoupling of disturbance requires

$$\boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yd}(s) = 0.$$

Insensitivity to input requires the model to fulfil

$$\boldsymbol{V}_{ru}(s) + \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yu}(s) = 0$$

and both are subject to the constraint that sensitivity to faults is not vanishing

$$\boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yf}(s) \neq 0$$

Norms and gains. In order to treat the relaxed condition, it is not required that the right-hand sides are exactly zero, but we wish to obtain minimal values subject to constraints like stable systems and causal realisation of filters. In order to formulate adequate optimisation problems, recall the definitions of the vector norm and the matrix norm induced by a vector norm: Let $\boldsymbol{x} \in \mathbb{R}^n$. Then the vector p-norm of $\boldsymbol{x}$ is

$$|\boldsymbol{x}|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

In particular, when $p = 2, \infty$,

$$|\boldsymbol{x}|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$$

and

$$|\boldsymbol{x}|_\infty = \max_{1 \leq i \leq n} |x_i| .$$

Further, let $\boldsymbol{A} = (a_{ij}) \in \mathbb{R}^{m \times n}$ and $\boldsymbol{x} \in \mathbb{R}^n$. The matrix norm induced by a vector p-norm is defined as

$$|\boldsymbol{A}|_p = \sup_{x \neq 0} \frac{|\boldsymbol{A}\boldsymbol{x}|_p}{|\boldsymbol{x}|_p}$$

In particular, when $p = 2, \infty$,

$$|\boldsymbol{A}|_2 = \sqrt{\lambda_{\max}(\boldsymbol{A}'\boldsymbol{A})} = \bar{\sigma}(A)$$

and

$$|\boldsymbol{A}|_\infty = \max_{1\le i\le m} \sum_{j=1}^{n} |a_{ij}| \qquad \textit{(maximum absolute row sum)},$$

where $\lambda_{\max}$ is the largest eigenvalue, and $\bar{\sigma}$ is the largest singular value.

It is noted that an induced norm can be viewed as a mapping from a vector space $\mathbf{C}^n$ equipped with a norm $|\cdot|_p$ to a vector space $\mathbf{C}^m$ with a norm $|\cdot|_p$. The induced norms have the interpretation of input/output amplification gains.

Let $\boldsymbol{H}(j\omega) \in \mathbf{C}^{m\times n}$ be a stable transfer function, i.e. with all poles strictly in the left half plane. Then the 2-norm is

$$|\boldsymbol{H}|_2 = trace\left(\frac{1}{2\pi}\int_{-\infty}^{\infty} \boldsymbol{H}(j\omega)\,\boldsymbol{H}'(-j\omega)\,d\omega\right)^{\frac{1}{2}}$$

and the ∞ norm

$$|\boldsymbol{H}|_\infty = \max_{\omega} \bar{\sigma}\ (\boldsymbol{H}(j\omega))\,.$$

An important result is that

$$|\boldsymbol{H}\boldsymbol{f}|_2^2 \le |\boldsymbol{H}|_\infty^2\,|\boldsymbol{f}|_2^2 = \max_{\omega} \bar{\sigma}(\boldsymbol{H}(j\omega))^2\,|\boldsymbol{f}|_2^2$$

since

$$\begin{aligned} |\boldsymbol{H}\boldsymbol{f}|_2^2 &= \frac{1}{2\pi}\int_{-\infty}^{\infty} \boldsymbol{f}'(-j\omega)\boldsymbol{H}'(-j\omega)\boldsymbol{H}(j\omega)\boldsymbol{f}(j\omega)d\omega \\ &\le |\boldsymbol{H}|_\infty^2\,\frac{1}{2\pi}\int_{-\infty}^{\infty} \boldsymbol{f}'(-j\omega)\boldsymbol{f}(j\omega)d\omega \\ &= |\boldsymbol{H}|_\infty^2\,|\boldsymbol{f}|_2^2\,, \end{aligned}$$

which shows $|\boldsymbol{H}|_\infty^2$ is the upper bound for the signal power transmitted from input to output of the transfer function $\boldsymbol{H}(s)$.

Formulation as an optimisation problem. The first property of a relaxed residual generator should be minimisation of the effect of disturbances in the residual.

A direct minimisation of the effect the disturbance has on the residual is expressed in the induced norm

$$\min_{\boldsymbol{V}_{ry}} \boldsymbol{J}_{id} = \min_{\boldsymbol{V}_{ry}} \frac{|\boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yd}(s)\boldsymbol{d}(s)|_2^2}{|\boldsymbol{d}(s)|_2^2} = \min_{\boldsymbol{V}_{ry}} |\boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yd}(s)|_\infty^2$$

subject to

$$\boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yf}(s) \neq 0.$$

The constraint prevents the trivial solution $\boldsymbol{V}_{ry}(s) = 0$.

The signal power comprised in the residual caused by the disturbance over the power generated by faults should be minimised, hence a feasible index could be

$$\max_{\boldsymbol{V}_{ry}} \boldsymbol{J}_2 = \max_{\boldsymbol{V}_{ry}} \left(\frac{|\boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yf}(s)\boldsymbol{f}(s)|_2^2}{|\boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yd}(s)\boldsymbol{d}(s)|_2^2} \right)_{|\boldsymbol{d}|\neq 0}.$$

The interpretation of this index is to maximise the signal over noise ratio in the residual, using the total power, i.e. over all frequencies. This index cannot be easily optimised. If we, however, make a slight modification to the optimisation criterion, standard tools are available.

As a general tool for optimisation, the standard setup formulation is widely used in robust and optimal control theory and is widely supported by computer aided design tools. Hence it is advantageous to describe the optimisation problem in the standard setup formulation.

Application of the standard methods require a specific formulation of the problem, which is first illustrated using manipulation on the block diagram in Fig. 6.4 for the case, where the objective is to find a polynomial matrix $\boldsymbol{F}(s)$ such that a signal $\boldsymbol{e}(s)$ is insensitive to a disturbance $\boldsymbol{d}(s)$

$$\boldsymbol{e}(s) = (\boldsymbol{H}_{zd}(s) - \boldsymbol{F}(s)\boldsymbol{H}_{yd}(s))\ d(s).$$

6.5.2 Solution using the standard setup formulation

We introduce first the basic notion of the standard estimation setup and the standard estimation problem, which have a direct bearing on design of residual generators.

Definition 6.4 (Standard estimation setup)
Let a system be given by input vector (known and unknown input) $\boldsymbol{d} \in \mathbb{R}^{n_d}$, state vector $\boldsymbol{x} \in \mathbb{R}^n$, an auxiliary output $\boldsymbol{z} \in \mathbb{R}^l$ and measured output vector $\boldsymbol{y} \in \mathbb{R}^p$ with state-space equation

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{E}_x\boldsymbol{d}(t) \\ \boldsymbol{z}(t) &= \boldsymbol{C}_z\boldsymbol{x}(t) + \boldsymbol{E}_z\boldsymbol{d}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}_y\boldsymbol{x}(t) + \boldsymbol{E}_y\boldsymbol{d}(t) \end{aligned} \tag{6.78}$$

and, ignoring initial conditions, represented in the Laplace domain by

$$\begin{aligned} \boldsymbol{z}(s) &= \boldsymbol{H}_{zd}(s)\boldsymbol{d}(s) \\ \boldsymbol{y}(s) &= \boldsymbol{H}_{yd}(s)\boldsymbol{d}(s), \end{aligned} \tag{6.79}$$

where

$$\boldsymbol{H}_{zd}(s) = \left(\begin{array}{cc} \boldsymbol{A} & \boldsymbol{E}_x \\ \boldsymbol{C}_z & \boldsymbol{E}_z \end{array} \right) = \boldsymbol{C}_z(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{E}_x + \boldsymbol{E}_z \tag{6.80}$$

$$\boldsymbol{H}_{yd}(s) = \left(\begin{array}{cc} \boldsymbol{A} & \boldsymbol{E}_x \\ \boldsymbol{C}_y & \boldsymbol{E}_y \end{array} \right) = \boldsymbol{C}_y(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{E}_x + \boldsymbol{E}_y. \tag{6.81}$$

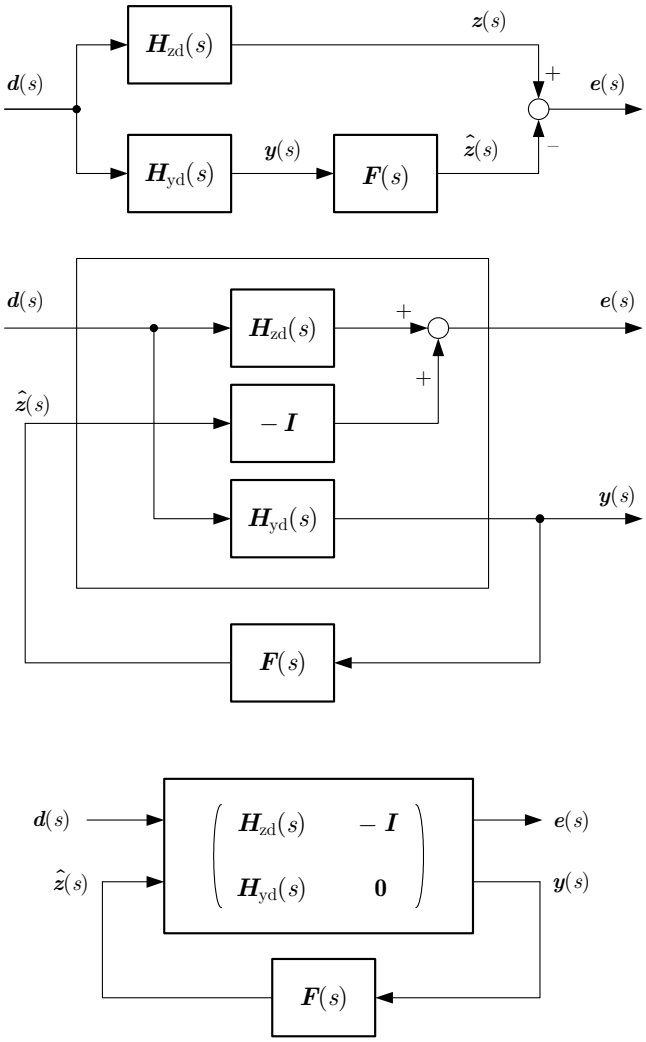


Fig. 6.4. Manipulation of the block diagram to arrive at a standard problem formulation. The upper two diagrams are equivalent, the lower is the representation used to determine $\boldsymbol{F}(s)$ by standard methods.

Problem 6.4 (Standard estimation problem)
Let $\hat{\boldsymbol{z}}(s)$ be an estimate of $\boldsymbol{z}(s)$. Denote the difference by $\boldsymbol{e}_z(s) = \boldsymbol{z}(s) - \hat{\boldsymbol{z}}(s)$. For the system defined by the standard problem setup (6.79), determine a stable transfer function matrix $\boldsymbol{F}(s)$ to provide an estimate of the auxiliary output given the measured output,

$$\hat{\boldsymbol{z}}(s) = \boldsymbol{F}(s)\boldsymbol{y}(s) \tag{6.82}$$

subject to a suitable norm of the estimation error, $|\boldsymbol{e}_z(s)|$ being less than a chosen gain factor

$$\sup_{\boldsymbol{F}(s)} |\boldsymbol{e}_z(s)| < \gamma \Leftrightarrow \sup_{\boldsymbol{F}(s)} |\boldsymbol{H}_{zd}(s) - \boldsymbol{F}(s)\boldsymbol{H}_{yd}(s)| < \gamma, \tag{6.83}$$

where the norm can be of types $\mathcal{H}_2$ or $\mathcal{H}_\infty$ for instance.

Equation (6.83) follows from expanding the estimation error:

$$\begin{aligned} \boldsymbol{e}_z(s) &= \boldsymbol{z}(s) - \hat{\boldsymbol{z}}(s) \\ &= \boldsymbol{z}(s) - \boldsymbol{F}(s)\boldsymbol{y}(s) \\ &= (\boldsymbol{H}_{zd}(s) - \boldsymbol{F}(s)\boldsymbol{H}_{yd}(s))\boldsymbol{d}(s). \end{aligned} \tag{6.84}$$

Remark 6.6
Different filtering and estimation problems can be easily formulated within this general estimation framework. The state can be estimated using $\boldsymbol{C}_z = \boldsymbol{I}_{n,n}$ and $\boldsymbol{E}_z = \boldsymbol{O}$. The input can be estimated using $\boldsymbol{C}_z = \boldsymbol{O}$ and $E_z = I_{l,l}$. The estimation setup will be used later for residual generation. □

Standard $\mathcal{H}_2$ and $\mathcal{H}_\infty$ methods that find the minimum of function according to the selected norm can also be applied to find a suitable $\boldsymbol{F}(s)$ transfer function matrix for the estimation problem. Use of widely available software for this purpose (for example the MATLAB μ toolbox) requires formulation in what is referred to as the *standard system setup and standard problem* in robust control.

Definition 6.5 (Standard system setup)
Let a system be described in the Laplace domain by the transfer function matrix $P(s)$, and four vectors, input $\boldsymbol{u}(s) \in \mathbf{C}^m$, auxiliary input $\boldsymbol{d}(s) \in \mathbf{C}^{n_d}$, auxiliary output $\boldsymbol{e}(s) \in \mathbf{C}^{m_e}$ and measured output $\boldsymbol{y}(s) \in \mathbf{C}^p$. Input and output are related through $\boldsymbol{P}(s) \in \mathbf{C}^{(p+m_e)\times(m+n_d)}$ as

$$\begin{pmatrix} \boldsymbol{e}(s) \\ \boldsymbol{y}(s) \end{pmatrix} = \boldsymbol{P}(s) \begin{pmatrix} \boldsymbol{d}(s) \\ \boldsymbol{u}(s) \end{pmatrix} = \begin{pmatrix} \boldsymbol{P}_{ed}(s) & \boldsymbol{P}_{eu}(s) \\ \boldsymbol{P}_{yd}(s) & \boldsymbol{P}_{yu}(s) \end{pmatrix} \begin{pmatrix} \boldsymbol{d}(s) \\ \boldsymbol{u}(s) \end{pmatrix}$$

Let the transfer function matrix $\boldsymbol{F}(s) \in \mathbf{C}^{m\times p}$ be a feedback controller for the system, between $\boldsymbol{y}$ and $\boldsymbol{u}$,

$$\boldsymbol{u}(s) = \boldsymbol{F}(s)\boldsymbol{y}(s)$$

Using this setup and utilising solutions for two fundamental optimisation problems in the design of residual generators, the $\mathcal{H}_\infty$ sub-optimal control problem and the $\mathcal{H}_2$ sub-optimal control problem.

Problem 6.5 ($\mathcal{H}_\infty$ suboptimal control)
Given a system in form of the standard system setup of Definition 6.5. Design a stabilising controller $\boldsymbol{F}(s)$ such that the norm of the closed-loop transfer function $\boldsymbol{T}_{ed}(s)$ from auxiliary input $\boldsymbol{d}(s)$ to auxiliary output $\boldsymbol{e}(s)$ is lower than a specified bound γ:

$$\sup_{\boldsymbol{F}} |T_{ed}|_\infty < \gamma \Leftrightarrow \sup_{\boldsymbol{F}(j\omega)} \bar{\sigma}(T_{ed}(j\omega)) < \gamma,$$

where $\bar{\sigma}$ denotes the largest singular value.

The $\mathcal{H}_\infty$ norm gives the maximum sinusoidal gain of the system (energy gain or induced L_2 system gain).

Problem 6.6 ($\mathcal{H}_2$ sub-optimal control problem)
Given a system in form of the standard system setup in Definition 6.5. Design a stabilizing controller $\boldsymbol{F}(s)$ such that the $\mathcal{H}_2$ norm of the closed-loop transfer function $\boldsymbol{T}_{ed}(s)$ from auxiliary input $\boldsymbol{d}(s)$ to auxiliary output $\boldsymbol{e}(s)$ is minimised.

The standard estimation problem of Fig. 6.4 can be formulated in the standard setup. The generalised system $\boldsymbol{P}(s)$ then takes the form

$$\boldsymbol{P}(s) = \begin{pmatrix} \boldsymbol{H}_{zd}(s) & -\boldsymbol{I} \\ \boldsymbol{H}_{yd}(s) & \boldsymbol{O} \end{pmatrix}.$$

Note that there is no feedback through the system since $\boldsymbol{P}_{yu}(s) = \boldsymbol{O}$.

6.5.3 Residual generation

The above result can be applied in connection with detection, isolation and estimation of faults. We aim at making $\hat{\boldsymbol{z}}(s)$ a residual signal. We investigate two problems. The first is to suppress disturbances as well as possible. The second is to make a balanced optimisation, where the fault signature is preserved in the residual while disturbances are suppressed to the extent possible. Both results follow from appropriate formulation of the standard problem. The strategy is to select an auxiliary output $\boldsymbol{z}(s)$ and give it the properties that the residual should have. This means the formulation of $\boldsymbol{z}(s)$ is directly a specification of the residual. In designing the estimate $\hat{\boldsymbol{z}}(s)$ to track $\boldsymbol{z}(s)$ as closely as possible, according to a given criterion, a sub-optimal estimator is obtained for the ideal residual. The accuracy with which the specification is met is seen in the choice of the optimisation coefficient γ.

The basic residual generator will have the form

$$\boldsymbol{r}(s) = \boldsymbol{F}(s)(\boldsymbol{y}(s) - \boldsymbol{H}_{yu}(s)\boldsymbol{u}(s)). \tag{6.85}$$

The design problem is to determine the operator $\boldsymbol{F}(s)$.

Remark 6.7 *Relation to parity space formulation*
The residual generator Eq. (6.43) had as prerequisite, following from Problem 6.2, that

$$\boldsymbol{V}_{ru} + \boldsymbol{V}_{ry}\boldsymbol{H}_{yu} = 0$$

hence

$$\begin{aligned}\boldsymbol{r}(s) &= \boldsymbol{V}_{ry}(s)\boldsymbol{y}(s) + \boldsymbol{V}_{ru}\boldsymbol{u}(s)\\ &= \boldsymbol{V}_{ry}(s)\boldsymbol{y}(s) - \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yu}\boldsymbol{u}(s)\\ &= \boldsymbol{V}_{ry}(s)(\boldsymbol{y}(s) - \boldsymbol{H}_{yu}(s)\boldsymbol{u}(s)).\end{aligned}$$

Comparison with Eq. (6.85) shows that finding the solution $\boldsymbol{F}(s)$ in the standard setup is equivalent to determining the operator $\boldsymbol{V}_{ry}(s)$. □

In the design, two requirements have to be combined.

Residual generation with specification on fault sensitivity and disturbance suppression. The goal is now to have the residual replicating a fault through a specified dynamical relation while the disturbance should be suppressed as far as possible. Therefore, we include the fault vector $\boldsymbol{f}(s)$ in the system description and define the auxiliary output $\boldsymbol{z}(s)$ to be dependent only of the fault vector:

$$\begin{aligned}\boldsymbol{y}(s) &= \boldsymbol{H}_{yd}(s)\boldsymbol{d}(s) + \boldsymbol{H}_{yf}(s)\boldsymbol{f}(s)\\ \boldsymbol{z}(s) &= \boldsymbol{H}_{zf}(s)\boldsymbol{f}(s)\\ \hat{\boldsymbol{z}}(s) &= \boldsymbol{V}_{ry}(s)\boldsymbol{y}(s)\\ \boldsymbol{e}_z(s) &= \boldsymbol{z}(s) - \hat{\boldsymbol{z}}(s)\end{aligned}$$

The selection of the auxiliary output reflects directly the properties that the residual should have. $\boldsymbol{H}_{zd} = \boldsymbol{O}$ is chosen because we wish to interpret $\boldsymbol{d}(s)$ as a disturbance and decouple it from the residual. The specification of $\boldsymbol{H}_{zf}(s)$ is a design choice. There may not exist a solution $\boldsymbol{V}_{ry}(s)$ for all arbitrary specifications, so $\boldsymbol{H}_{zf}(s)$ is the key design parameter.

The performance that should be achieved is that the residual follows $\boldsymbol{z}(s)$ as close as possible, hence the relation

$$|\boldsymbol{H}_{zf}(s) - \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yf}(s)|_\infty < \gamma_s$$

should hold, where γ_s characterises the desired fault sensitivity (or tracking) performance. Simultaneously, the effect of the disturbance should be below a certain level, hence

$$|\boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yd}(s)|_\infty < \gamma_r,$$

where γ_r is a measure of robustness with respect to input effects. Combining the two, the physically motivated optimisation problem yields:

$$|(-\boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yd}(s)) \quad (\boldsymbol{H}_{zf}(s) - \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yf}(s))|_\infty < \gamma. \tag{6.86}$$

Since

$$\begin{aligned}&\boldsymbol{z}(s) - \boldsymbol{V}_{ry}(s)\boldsymbol{y}(s)\\ &= (\boldsymbol{H}_{zf}(s) - \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yf}(s))\,\boldsymbol{f}(s) - \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yd}(s)\boldsymbol{d}(s)\\ &= ((-\boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yd}(s)) \quad (\boldsymbol{H}_{zf}(s) - \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yf}(s)))\begin{pmatrix}\boldsymbol{d}(s)\\ \boldsymbol{f}(s)\end{pmatrix}\end{aligned}$$

Eq. (6.86) is equivalent to

$$\sup_{\begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{p} \end{pmatrix} \neq \boldsymbol{0}} \frac{|\boldsymbol{z}(j\omega) - \boldsymbol{V}_{ry}(j\omega)\boldsymbol{y}(j\omega)|_2}{\left| \begin{pmatrix} \boldsymbol{d}(j\omega) \\ \boldsymbol{f}(j\omega) \end{pmatrix} \right|_2} < \gamma \Leftrightarrow \sup_{\begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{p} \end{pmatrix} \neq \boldsymbol{0}} \frac{|\boldsymbol{e}_z(j\omega)|_2}{\left| \begin{pmatrix} \boldsymbol{d}(j\omega) \\ \boldsymbol{f}(j\omega) \end{pmatrix} \right|_2} < \gamma.$$

The residual generation design problem is illustrated in Fig. 6.5. The upper diagram in the Figure depicts the residual generator with both specifications $\boldsymbol{H}_{zd}$ and $\boldsymbol{H}_{zf}$ given. In formulating the requirement that disturbance feed-through to the residual should be minimal, $\boldsymbol{H}_{zd} = \boldsymbol{O}$ is specified in the setup shown in the lower part of Fig. 6.5.

In the standard setup, the residual generator design has the following form:

Problem 6.7 (Residual generation with specification on fault sensitivity and disturbance suppression)
Given an LTI system with input $\boldsymbol{u}(s)$, unknown input (disturbances) $\boldsymbol{d}(s)$ and faults $\boldsymbol{f}(s)$ and let input-output relations of the system be described by $(\boldsymbol{H}_{yu}, \boldsymbol{H}_{yd}, \boldsymbol{H}yf)$. Introduce an auxiliary variable $\boldsymbol{z}(s)$ and specify a transfer function matrix $\boldsymbol{H}_{zf}$ and a real number γ_s. Let $\boldsymbol{z}(s) = \boldsymbol{H}_{zf}\boldsymbol{f}(s)$. Determine $\boldsymbol{V}_{ry}$ such that the maximal deviation between $\hat{\boldsymbol{z}}(s) = \boldsymbol{V}_{ry}\boldsymbol{y}(s)$ and $\boldsymbol{z}(s)$ is bounded by γ_s:

$$|\boldsymbol{z}(s) - \hat{\boldsymbol{z}(s)}| \quad < \quad \gamma_s \tag{6.87}$$

The solution to this problem of residual generation design has the following form:

1. Define the standard problem setup:

$$\begin{aligned}
\text{aux.input}: \quad & \boldsymbol{d}(s) \leftarrow \begin{pmatrix} \boldsymbol{d}(s) \\ \boldsymbol{f}(s) \end{pmatrix} \\
\text{input}: \quad & \boldsymbol{u}(s) \leftarrow \boldsymbol{r}(s) \\
\text{aux.output}: \quad & \boldsymbol{e}(s) \leftarrow \boldsymbol{e}_z(s) = \boldsymbol{z}(s) - \boldsymbol{r}(s) \\
\text{output}: \quad & \boldsymbol{y}(s) \leftarrow \boldsymbol{y}(s) \\
& \boldsymbol{P}(s) \leftarrow \begin{pmatrix} \begin{pmatrix} \boldsymbol{O} & \boldsymbol{H}_{zf}(s) \end{pmatrix} & -\boldsymbol{I} \\ \begin{pmatrix} \boldsymbol{H}_{yd}(s) & \boldsymbol{H}_{yf}(s) \end{pmatrix} & \boldsymbol{O} \end{pmatrix} \\
& \boldsymbol{F}(s) \leftarrow \boldsymbol{V}_{ry}(s)
\end{aligned}$$

2. Use software that solves the standard problem to determine a solution in form of a stable transfer function $\boldsymbol{V}_{ry}(s)$ that satisfies the inequality

$$\sup_{\begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{f} \end{pmatrix} \neq \boldsymbol{0}} \frac{|\boldsymbol{e}_z|_2}{\left| \begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{f} \end{pmatrix} \right|_2} < \gamma$$

which is equivalent to finding a solution to

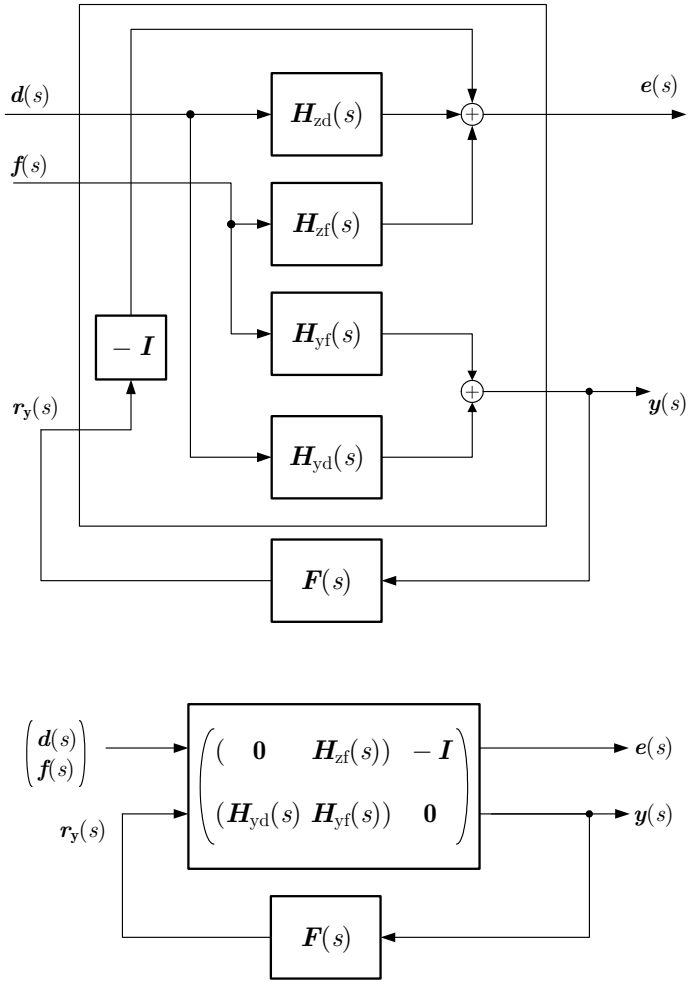


Fig. 6.5. Residual generator depicted in a standard setup formulation with specifications $\boldsymbol{H}_{zf}$ and $\boldsymbol{H}_{zd}$ in the upper part of the figure. $\boldsymbol{H}_{zd}$ is specified as zero in the design problem shown in the lower part of the figure.

$$\left\| \begin{pmatrix} -\boldsymbol{V}_{ry}\boldsymbol{H}_{yd} & \boldsymbol{H}_{zf} - \boldsymbol{V}_{ry}\boldsymbol{H}_{yf} \end{pmatrix} \right\|_{\infty} < \gamma.$$

If a result exists, which is not guaranteed, the result is strong in the sense it provides the residual generator with optimal weighting between suppression of disturbances and specified sensitivity to faults.

In practice it is worthwhile to start a design with investigating the extent to which disturbances can be suppressed using the disturbance suppression problem. When insight in the problem has been gained, continue with supplying a specification to

the problem and iterate until a suitable compromise has been found between disturbance suppression and fault tracking.

Fault detection. When the purpose is to design a pure fault detection filter, a sensible way to specify $\boldsymbol{H}_{zf}(s)$ is to require that it is a row vector with nonzero causal and stable entries. When a residual vector is sought the specification becomes:

$$\forall \omega; j\omega \neq z_k : \begin{cases} \text{rank } (\boldsymbol{H}_{zf}(j\omega)) \leq 1 \\ \forall i : h_i(j\omega) \neq 0, \end{cases}$$

where $h_i(j\omega)$ stands for the i^{th} column of $\boldsymbol{H}_{zf}(j\omega)$ and z_k are the zeros of $\boldsymbol{H}_{zf}(s)$.

Fault isolation. If the number of faults to be isolated is $\boldsymbol{n}_f$ and simultaneous faults can occur, $\boldsymbol{H}_{zf}(s)$ has to fulfil the requirement:

$$\text{rank } \boldsymbol{H}_{zf}(s) = \boldsymbol{n}_f,$$

where, as usual, the normal rank is considered.

When simultaneous faults are not considered, a vector $\boldsymbol{z}$ of size l is sufficient to isolate $2^l - 1$ faults by considering suitable coding sets. This translates into the following specification for matrix $\boldsymbol{H}_{zf}(s)$. To isolate $\boldsymbol{n}_f$ faults, choose a matrix $\boldsymbol{H}_{zf}(s)$ such that:

$$\begin{aligned} \text{rank } \boldsymbol{H}_{zf}(s) &\geq \log_2(\boldsymbol{n}_f + 1) \\ \text{rank } (\boldsymbol{h}_i(s)\, \boldsymbol{h}_j(s)) &= 2 \quad \text{with} \quad i = 1, ..., \boldsymbol{n}_f i \neq j \quad j = 1, ..., \boldsymbol{n}_f. \end{aligned}$$

Fault estimation. Fault estimation can be obtained by specifying $\boldsymbol{H}_{zf}(s) = \boldsymbol{I}$. In this case,

$$\boldsymbol{z}(s) = \boldsymbol{I} f(s)$$

and

$$\boldsymbol{e}(s) = \boldsymbol{z}(s) - \hat{\boldsymbol{z}}(s).$$

In the ideal situation, where no disturbance exists, this specification aims at assuring that $\hat{\boldsymbol{z}}$ tracks the fault f by guaranteeing that

$$\sup \frac{|\hat{\boldsymbol{z}} - \boldsymbol{z}|_2}{|f|_2} < \gamma.$$

When disturbances do exist, a trade-off is made between fault tracking and insensitivity of $\hat{\boldsymbol{z}}$ to the disturbance. The block diagram to specify fault estimation from the solution to a standard problem is shown in Fig. 6.6.

Design considerations. In connection with using $\mathcal{H}_2$ or $\mathcal{H}_\infty$ optimisation to design the residual generator, a weight function can further be included in the setup to some

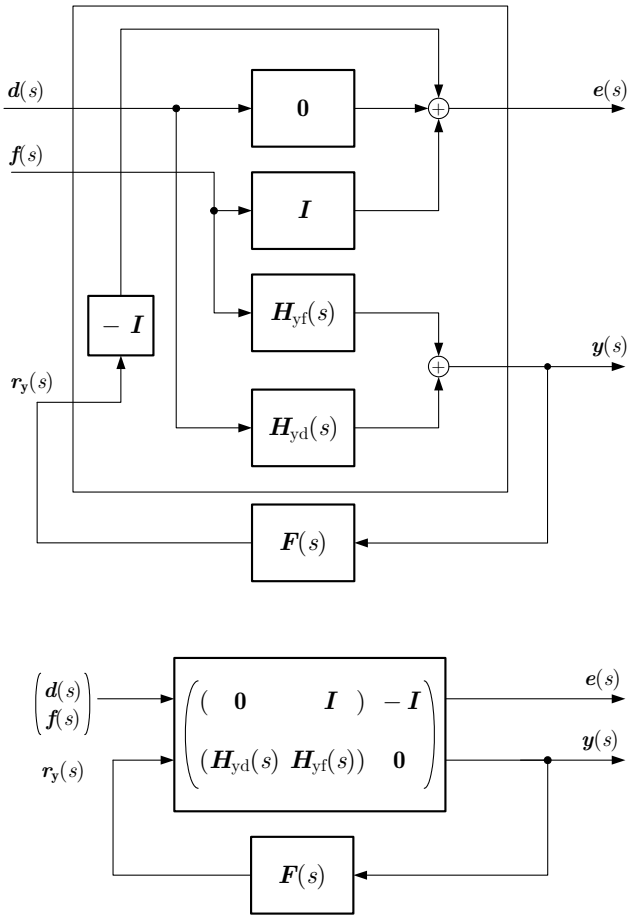


Fig. 6.6. If a solution $\boldsymbol{F}(s)$ exists, fault estimation is obtained by solving the standard problem using the specification $\boldsymbol{H}_{zd} = \boldsymbol{O}$, $\boldsymbol{H}_{zf} = \boldsymbol{I}$.

advantage of the designer. The weight function $\boldsymbol{W}(s)$ can be applied to specify the frequency range(s), where detection, isolation or estimation should be obtained most effectively. The way to include a weight matrix in the design is to modify the $\boldsymbol{P}(s)$ system matrix to

$$\boldsymbol{P}(s) = \begin{pmatrix} \begin{pmatrix} \boldsymbol{O} & \boldsymbol{W}\boldsymbol{H}_{zf} \\ \boldsymbol{H}_{yd} & \boldsymbol{H}_{yf} \end{pmatrix} & -\boldsymbol{W} \\ \boldsymbol{O} & \end{pmatrix}.$$

The weighting matrix specifies in which frequency ranges a designer emphasises to meet the bound γ and where it can be relaxed.

The main issue in using the standard setup to obtain sub-optimal residual generators of the different classes described above is the selection of a proper specification

$\boldsymbol{H}_{zf}(s)$. Whereas the optimisation itself is left to the software tools available, good results are only obtained if a good specification is provided. An iterative design method has proved useful in practice.

Algorithm 6.4 *Residual generator design*

1. **Formulate problem:** Formulate the relevant version of the standard setup for the problem.
2. **Design specification:** Specify an initial qualified guess on the specification $\boldsymbol{H}_{zf}(s)$. The specification needs to be bounded from below, otherwise, the optimal solution will be $\boldsymbol{F} = \boldsymbol{O}$ and $\boldsymbol{H}_{zf} = \boldsymbol{O}$.
3. **Solve problem:** Find the function $\boldsymbol{F}(s)$ in the residual generator
$$\boldsymbol{r}(s) = \boldsymbol{F}(s)(\boldsymbol{y}(s) - \boldsymbol{H}_{yu}(s)\boldsymbol{u}(s)),$$
where $\boldsymbol{F}(s)$ is the best obtainable solution to the problem given the specification $\boldsymbol{H}_{zf}(s)$.
4. **Iterate until converged:** Continue until the value of γ obtained has converged.
5. **Iterate in specification:** Based on this residual generator, specify a new $\boldsymbol{H}_{zf}(s)$ and repeat the design.

The procedure usually requires very few iterations.

6.6 Residual evaluation

Given a residual generator for the deterministic case, i.e. there are only insignificant random disturbances or measurement noise, the purpose of this section is to find a method for residual evaluation that will determine whether a fault is present.

Residual - general case. Let a set of residuals obtained from structural analysis have the form $\boldsymbol{r} = (r_1, r_2, ..., r_n)'$. Consider one of these residuals

$$r_j(t) = p_j(k_i, c_i, t) \qquad k_i \in K^{(j)}, c_i \in C^{(j)}, \quad j = 1, ..., n, \tag{6.88}$$

where p_j is of the form in which the constraints in $C^{(j)}$ were formulated: linear, nonlinear, tabular, quantised, logical or hybrid. As it is useful to categorise known variables into the natural categories input u, measured y, and parameters θ, the parity relation is written as

$$r_j(t) = p_j(u_i, y_i, \theta_i, c_i, t) \quad u_i, y_i, \theta_i \in K^{(j)}, c_i \in C^{(j)}, \quad j = 1, ..., n. \tag{6.89}$$

The parity relations implemented for residual generation would not be the true system constraints c_i nor the true parameters θ_i but would be estimates of those, $\hat{c}_i, \hat{\theta}_i$, respectively.

In order to shape the signatures of faults in the residuals or suppress noise, filtering of the raw parity relation Eq. (6.89) will usually take place, and also the filtered version is a residual,

$$r_j(t) = \int_0^t w_j(t-\tau) p_j(u_i, y_i, \hat{\theta}_i, \hat{c}_i, \tau) d\tau, \quad j = 1, ..., n, \tag{6.90}$$

where $w_j(t-\tau)$ is the impulse response of the filter applied to parity relation j.

Further, the vector of residuals could be constructed as a linear combination of the elements from the above residuals, Eq. (6.90),

$$\boldsymbol{r}(t) = \boldsymbol{W} \begin{pmatrix} r_1(t) \\ \vdots \\ r_n(t) \end{pmatrix}, \tag{6.91}$$

where $\mathbf{W} \in \mathbb{R}^{n \times n}, \det(\mathbf{W}) \neq 0$.

Uncertainty. In real life, $\hat{c}_i \neq c_i$, and $\hat{\theta}_i \neq \theta_i$, hence $r_j(t)$ could be nonzero even though there was no violation of a constraint in relation j, $\forall c_i \in C^{(j)} : c_i = 0$. In particular, actuator demand and disturbances could drive the residual away from zero when parameters and constraints are not exactly equal to those of the real object. In order to make residual evaluation under such uncertainty, it is necessary to accept that a residual can have some deviation from zero even in the no-fault case. However, the effect on $r_j(t)$ has to be bounded, hence $p_j(u_i, y_i, \hat{\theta}_i, \hat{c}_i, t)$ is bounded,

$$\left\| p_j(u_i, y_i, \hat{\theta}_i, \hat{c}_i, t) \right\| \leq \alpha_j(u_i, y_i, t) \wedge 0 < \alpha(u_i, y_i, t) < \infty. \tag{6.92}$$

LTI case. If the object for diagnosis is linear and time-invariant (LTI), the residual generator could be LTI with a frequency representation

$$\mathbf{r}(s) = \mathbf{H}_{ru}(s)\boldsymbol{u}(s) + \mathbf{H}_{rd}(s)\mathbf{d}(s) + \mathbf{H}_{rf}(s)\mathbf{f}(s) \tag{6.93}$$

being an explicit function of input, disturbances and faults.

In an ideal case, residual generation is perfect and we have $\mathbf{H}_{ru}(s) = \boldsymbol{O}$ and $\mathbf{H}_{rd}(s) = \boldsymbol{O}$. Residual evaluation then reduces to investigating the properties of

$$\mathbf{r}(s) = \mathbf{H}_{rf}(s)\mathbf{f}(s). \tag{6.94}$$

In the general case, still with an LTI system, model uncertainty and unmodelled dynamics will give rise to $\mathbf{H}_{ru}(s) \neq \boldsymbol{O}$ and $\mathbf{H}_{rd}(s) \neq \boldsymbol{O}$. Residual evaluation need then be made such that false alarms are avoided from control input $\boldsymbol{u}(t)$ and disturbances $\mathbf{d}(t)$ within the normal range.

6.6.1 Evaluation against a threshold

Validating that no fault is present is equivalent with checking that the residual vector is zero. Validating the presence of a fault means checking whether the residual is or

has been different from zero. The two hypotheses and the associated condition on the residual vector are

$$\begin{array}{lll} \mathcal{H}_0(0,t): & \text{no fault is present} & \|\mathbf{r}\| = 0 \\ \mathcal{H}_1(f_j,t_j): & \text{fault } f_j \text{ was present since time } t_j & \|\mathbf{r}(t)\| \neq 0, t \geq t_j, \end{array} \tag{6.95}$$

where $\|\mathbf{r}\|$ is an appropriate norm of the residual.

Test function. For generality, introduce a test function $\varphi(r(t))$, which provides a measure (norm) of the residual's deviation from zero. Some common test functions are the following

- Absolute value

$$\varphi(r_j(t)) = |r_j(t)|\,. \tag{6.96}$$

- An approximation to the two-norm of the residual vector

$$\varphi(r_j(t)) = \left(\frac{1}{T} \int_{t-T}^{t} |r_j(\tau)|^2 \, d\tau \right)^{\frac{1}{2}}. \tag{6.97}$$

- Square root of filtered absolute value, squared,

$$\varphi(r_j(t)) = \left(\int_0^t w_\varphi^{(j)}(t-\tau)\, |r_j(\tau)|\, d\tau \right)^{\frac{1}{2}}. \tag{6.98}$$

- Filtered mean square value of signal

$$\varphi(r_j(t)) = \int_0^t w_\varphi^{(j)}(t-\tau) \left(r_j(\tau) - \frac{1}{T} \int_{\tau-T}^{\tau} r_j(\tau_2) d\tau_2 \right)^2 d\tau, \tag{6.99}$$

where $w_\varphi^{(j)}(t)$ is the impulse response of a filter used particularly for evaluaton of residual j. In this context, the test function given in Eq. (6.96) is considered further.

Threshold function. The next step in residual evaluation is to determine a threshold function $\Phi(t)$ for evaluation of the test function $\varphi(t)$. $\Phi(t)$ should have the properties

$$\begin{array}{lll} \text{no fault:} & \forall t \geq 0,\ f(t) = 0: & \varphi(r(t)) \leq \Phi(t) \\ \text{weakly detectable fault:} & \exists t \geq t_0 : f(t) \neq 0: & \varphi(r(t)) > \Phi(t) \\ \text{strongly detectable fault:} & \forall t \geq t_1 \geq t_0 : f(t) \neq 0,\ t \geq t_0: & \varphi(r(t)) > \Phi(t) \end{array}$$

LTI case. In the ideal LTI case, Eq. (6.94), $\Phi(t)$ could be chosen constant and as close to zero as allowed by practical values of bias and noise in the residual.

In the non-ideal case, Eq. (6.93) applies and input and disturbances have some feed-through to the residual. With the test function $\varphi(t) = \|r_j(t)\|_2$, the threshold need be determined such that

$$\Phi_j(t) \geq \sup_{f=0,\|u,d\|<\varepsilon} (\varphi(r_j(t)))$$

is achieved in the time domain. The fact that total power calculated in the time domain and in the frequency domain are equal is used to determine the theshold function,

$$||r(j\omega)||^2 = \frac{1}{2\pi}\int_0^\infty r(j\omega)r(-j\omega)d\omega = \lim_{T\to\infty}\int_0^{'} |r(t)|^2 dt = ||r(t)||^2.$$

From Eq. (6.93), the residual is given in the frequency domain. Component j of the residual is

$$r_j(s) = (\mathbf{H}_{ru}(s)\boldsymbol{u}(s))^{(j)} + (\mathbf{H}_{rd}(s)\mathbf{d}(s))^{(j)} + (\mathbf{H}_{rf}(s)\mathbf{f}(s))^{(j)}.$$

With k control inputs

$$\begin{aligned} \|r_j(j\omega)\|_2 &\leq \|\mathbf{H}_{ru}(j\omega)\boldsymbol{u}(j\omega)\|_2^{(j)} + \|\mathbf{H}_{rd}(j\omega)\mathbf{d}(j\omega)\|_2^{(j)} \\ &\leq \sum_{i=1}^{k} \|\mathbf{H}_{ru}(j\omega)\|_\infty^{(ji)} \|u_i(j\omega)\|_2 + \|\mathbf{H}_{rd}(j\omega)\mathbf{d}(j\omega)\|_\infty^{(j)} \end{aligned} \tag{6.100}$$

for all admissible $\boldsymbol{u}$ and $\mathbf{d}$. The first term in the right hand side is a gain times input power. The second is the maximal contribution to the residual from disturbances.

Let the effect of disturbances on the residual be bounded by

$$\|\mathbf{H}_{rd}(j\omega)\mathbf{d}(j\omega)\|_\infty^{(j)} < \beta_d^{(j)}, \tag{6.101}$$

then $\Phi(t)$ should be chosen as the time-varying function

$$\begin{aligned} \Phi_j(t) &= \sum_{i=1}^{k} \beta_i \|u_i(t)\|_2 + \beta_d^{(j)} \\ \beta_i &= \|\mathbf{H}_{ru}(j\omega)\|_\infty^{(ji)}. \end{aligned} \tag{6.102}$$

This threshold is a function of maximal gains from control inputs to residual and of the maximum gain from disturbances to residual. It is often referred to as a time-varying threshold in the literature. The term adaptive threshold has also been used.

If the time-varying threshold Eq. (6.102) is too conservative, a dynamic bound could be specified as

$$\Phi_j(t) = \sum_{i=1}^{k} \left(\int_0^t \hat{h}_{ru}^{(ji)}(t-\tau)u_i(\tau)d\tau \right) + \varepsilon_d, \tag{6.103}$$

where $\hat{h}_{ru}^{(ji)}$ is an estimate of the maximum (envelope) of impulse response functions from input i to residual j for a given model uncertainty.

Example 6.10 *Ship example (LTI case)*

Assume the ship was LTI,

$$\begin{aligned} y_1(s) &= \omega_3(s) + \omega_w(s) + f_\omega(s) = \frac{b}{s-b\eta_1}\delta(s) + \omega_w(s) + f_\omega(s) \\ y_2(s) &= \psi(s) + f_\psi(s) = \frac{1}{s}(\omega_3(s) + \omega_w(s)) + f_\psi(s) \end{aligned} \tag{6.104}$$

the design model was

$$\begin{aligned}\hat{\omega}_3(s) &= \frac{\hat{b}}{s-\hat{b}\hat{\eta}_1}\delta \\ \hat{\psi}(s) &= \frac{1}{s}\hat{\omega}_3(s)\end{aligned}$$

and a residual generator is chosen as

$$\begin{aligned}r_1(s) &= y_1(s)-\hat{\omega}_3(s) = \left(\frac{b}{s-b\eta_1}-\frac{\hat{b}}{s-\hat{b}\hat{\eta}_1}\right)\delta(s)+\omega_w(s)+f_\omega(s) \\ r_2(s) &= \frac{\tau}{1+s\tau}\left(sy_2(s)-y_1(s)\right) = \frac{s\tau}{1+s\tau}f_\psi(s)-\frac{\tau}{1+s\tau}f_\omega(s).\end{aligned}$$

With no faults

$$\|r_1(j\omega)\|_2 \le \left\|\frac{b}{j\omega-b\eta_1}-\frac{\hat{b}}{j\omega-\hat{b}\hat{\eta}_1}\right\|_\infty \|\delta(j\omega)\|_2+\|\omega_w(j\omega)\|_\infty$$

and

$$\|r_1(j\omega)\|_2 \le \left\|\frac{b}{j\omega+b\eta_1}-\frac{\hat{b}}{j\omega+\hat{b}\hat{\eta}_1}\right\|_\infty \|\delta(t)\|_2+\|\omega_w(j\omega)\|_\infty \tag{6.105}$$

$$\Phi_1(t) = \left\|\frac{b}{j\omega-b\eta_1}-\frac{\hat{j\omega}}{j\omega-\hat{b}\hat{\eta}_1}\right\|_\infty \|\delta(t)\|_2+\|\omega_w(j\omega)\|_\infty \tag{6.106}$$

with $\|\omega_w(j\omega)\|_\infty \le \beta_d$,

$$\Phi_1(t) = \left|\frac{\hat{\eta}_1-\eta_1}{\eta_1\hat{\eta}_1}\right| \|\delta(t)\|_2+\beta_d = \beta_u\|\delta(t)\|_2+\beta_d. \tag{6.107}$$

In real time, we evaluate

$$\|r(t)\|_2 \le \beta_u\|\delta(t)\|_2+\beta_d \tag{6.108}$$

using Eq. (6.97) as an approximation to the two norm. □

General case. In the general case, if the parity relation is bounded by

$$\varphi(p_j(u_i,y_i,\hat{\theta}_i,\hat{c}_i,t)) \le \alpha_j(u_i,y_i,t) \wedge 0 < \alpha(u_i,y_i,t) < \infty. \tag{6.109}$$

The threshold function can obviously be chosen as

$$\Phi_j(t) \ge \alpha_j(u_i,y_i,t) \tag{6.110}$$

If more detailed information is available, e.g.

$$\alpha_j(u_i,y_i,t) \le \beta_0+\sum_{i=1}^{k}\beta_{ji}|u_i| \tag{6.111}$$

such information should be utilised when specifying the threshold, in this case as

$$\Phi_j(t) = \beta_0+\sum_{i=1}^{k}\beta_{ji}|u_i|. \tag{6.112}$$

Return to normal. The above procedure tested for the change H_0 to H_1. When a fault has been detected, $\varphi(r_j(t)) \geq \Phi_j(t) \Longrightarrow H^{(j)} = H_1$, change to normal is usually made with a hysteresis, $\gamma : \varphi(r_j(t)) < \gamma\Phi_j(t) \Longrightarrow H^{(j)} = H_0$. A common choice of hysteresis is $\gamma \subset [0.5, 0.8]$.

If a fault is only weakly detectable in residual j, but strongly detectable in other residuals, $\forall j : \varphi(r_j(t)) < \gamma\Phi_j(t) \Longrightarrow H^{(j)} = H_0$ should be used.

It is obvious that simulation and tests in the real environment some engineering judgement need be employed before good choices can be made of the timevarying threshold function $\Phi_j(t)$ and of the hysteresis γ.

This leads to algorithms for deterministic change detection,

Algorithm 6.5 *Test against time-varying threshold*

Given: A residual $r_j = p_j(u_i, y_i, \hat{\theta}_i, \hat{c}_i, t)$ and the object for diagnosis assumed in the no-fault condition.

1. Determine a test function $\varphi(r(t))$ according to Eqs. (6.96) to (6.99).

2. Determine a threshold function $\Phi_j(t)$:for the LTI case according to Eq. (6.102), for the general case according to Eq. (6.109) or Eq. (6.102) when scecific information is available.

Iniitialise: $H^{(j)} = H_0$.

Do:

1. Calculate $\varphi(r_j(t))$ and $\Phi_j(t)$.

2. If $H^{(j)} = H_0, \forall j$:.

If $\varphi(r_j(t)) \geq \Phi_j(t)$ set hypothesis to $H^{(j)} = H_1$.

Else:

If $\varphi(r_j(t)) < \gamma\ \Phi_j(t)$ for $\forall j$ set hypothesis to $H^{(j)} = H_1$.

Example 6.11 *Time-varying threshold for ship*

Let the ship's true constraints be:

$$\begin{array}{lrcl} c_1: & \dot{\omega}_3 = & b\eta_1\omega_3 + b\eta_3\omega_3^3 + b\delta \\ c_2: & \dot{\psi} = & \omega_3 + \omega_w \\ m_1: & y_1 = & \dot{\psi} \\ m_2: & y_2 = & \psi \end{array} \tag{6.113}$$

And let a model used for design be

$$\begin{array}{lrcl} \hat{c}_1: & \dot{\omega}_3 = & \hat{b}\hat{\eta}_1\omega_3 + \hat{b}\delta \\ \hat{c}_2: & \dot{\psi} = & \omega_3 \\ m_1: & y_1 = & \dot{\psi} \\ m_2: & y_2 = & \psi \end{array} \tag{6.114}$$

Using the model for design, a residual generator is suggested as

$$\begin{array}{ll} r_1 = & \frac{d}{dt}y_1 - \frac{d}{dt}\hat{y}_1 \\ r_2 = & \frac{d}{dt}y_2 - y_1 \end{array} \tag{6.115}$$

then, the real residual will vary with input and

$$\begin{array}{ll} r_1(t) = & (b\eta_1 - \hat{b}\hat{\eta}_1)\omega_3 + b\eta_3\omega_3^3 + (b-\hat{b})\delta(t) + \frac{d}{dt}\omega_w(t) \\ r_2(t) = & 0 \end{array}$$

$$\begin{aligned} |r_1(t)| \quad & \leq \left|b\eta_1 - \hat{b}\hat{\eta}_1\right| |y_1| + |b\eta_3| \left|y_1^3\right| + \left|b-\hat{b}\right| |\delta| + \left|\frac{d}{dt}\omega_w(t)\right|_{\sup} \\ & \leq \beta_1 |y_1| + \beta_3 \left|y_1^3\right| + \alpha_1 |\delta| + \beta_d \leq \alpha_2 |\delta| + \beta_d. \ \square \end{aligned}$$

6.7 Stochastic model – change detection algorithms

6.7.1 Introduction

To be able to solve the problem of fault detection, isolation and/or estimation that will be stated precisely in a stochastic framework below, a prerequisite is needed on sequential change detection algorithms. It is known that fault detection (and isolation or estimation) systems are made of two parts, a residual generator and a decision system. When the residual generator is designed on the basis of a linear stochastic model, residual evaluation reduces to the problem of detecting a change in the mean of a normally distributed random sequence, which can be achieved by sequential change detection algorithms. Therefore, this topic is considered before addressing successively fault detection, isolation and estimation in the case of additive faults.

6.7.2 Sequential change detection: the scalar case

Introduction. The sequential change detection algorithms are first derived in the simple case of processing a sequence of independent random variables with probability density function depending on a scalar parameter θ. The situation where θ is the mean of a Gaussian distribution is used to illustrate the theory, since this is the problem encountered for residual evaluation. As the sequential algorithms will be used to process residuals, the above theory has to be generalised to be able to detect changes in the mean of sequences of Gaussian vectors, which is done in a subsequent paragraph.

Problem statement. Consider a sequence of independent random variables $z(i)$, $i = 1, 2, \ldots,$ with probability density function $p_\theta(z)$ depending upon one scalar parameter θ. Before an unknown change time, k_0, θ is equal to θ_0. At time k_0, it changes to $\theta = \theta_1 \neq \theta_0$. The problem is to detect the change time, and possibly

estimate the value of the change in the parameter. No a priori knowledge of the distribution of the change time is assumed. θ_0 is known by hypothesis, and two situations are considered for θ_1, namely θ_1 known and θ_1 unknown. The first case yields the so-called cumulative sum (CUSUM) algorithm, the second the so-called generalised likelihood ratio (GLR) algorithm.

Both algorithms rely on a fundamental concept, namely the log-likelihood ratio of an observation z, which is defined as:

$$s(z) = \ln \frac{p_{\theta_1}(z)}{p_{\theta_0}(z)}. \tag{6.116}$$

The name comes from the fact that the likelihood function of the observation z is by definition equal to the probability density $p_\theta(z)$ of the underlying random variable evaluated at z. The likelihood function is thus a deterministic function of θ.

The log-likelihood ratio has the following fundamental statistical property:

$$E_{\theta_0}(s) = \int_{-\infty}^{\infty} s(z)\, p_{\theta_0}(z) dz < 0, \tag{6.117}$$

$$E_{\theta_1}(s) = \int_{-\infty}^{\infty} s(z)\, p_{\theta_1}(z) dz > 0. \tag{6.118}$$

E_{θ_0} (E_{θ_1}) denotes expectation of $s(z)$ under the distribution associated to $p_{\theta_0}(z)$ ($p_{\theta_1}(z)$). This property can be easily understood from the following example. Assume that $p_\theta(z)$ is a Gaussian distribution and that the parameter θ is the mean of this distribution, which will be denoted μ.

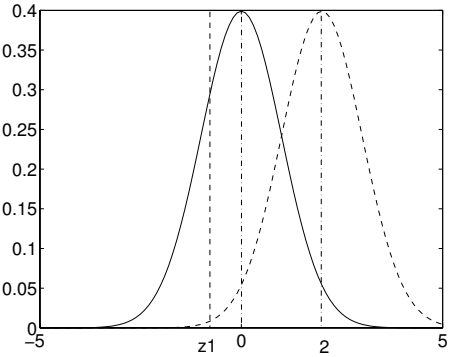


Fig. 6.7. Two Gaussian probability density functions with mean $\mu_0 = 0$ and $\mu_1 = 2$, and with the same variance $\sigma^2 = 1$

Consider Fig. 6.7. When the random variable z has $p_{\mu_0}(z)$ ($p_{\mu_1}(z)$) as probability density function, its realisations are most often in the "neighbourhood" of μ_0 (μ_1). Take the realisation z_1 for instance. Clearly $\frac{p_{\mu_1}(z_1)}{p_{\mu_0}(z_1)} < 1$. As z_1 is most probably obtained when the random variable z has $p_{\mu_0}(z)$ as probability density function, this illustrates that the logarithm of $\frac{p_{\mu_1}(z)}{p_{\mu_0}(z)}$ is on the average negative when z has

$p_{\mu_0}(z)$ as probability density function. The property described by (6.117), (6.118) is exploited in the next section to provide an intuitive derivation of the CUSUM algorithm.

Derivation of the CUSUM algorithm. The problem stated in the previous section, with θ_1 known, is addressed. Consider the cumulative sum:

$$\text{Cumulative sum:} \quad S(k) = \sum_{i=1}^{k} s(z(i)) = \sum_{i=1}^{k} \ln \frac{p_{\theta_1}(z(i))}{p_{\theta_0}(z(i))}. \tag{6.119}$$

In this expression, and in the remaining part of this section, k denotes the present time instant. From (6.117) and (6.118), $S(k)$ is expected to exhibit a negative drift before change, and a positive drift after change. This is illustrated in Fig. 6.8 and 6.9.

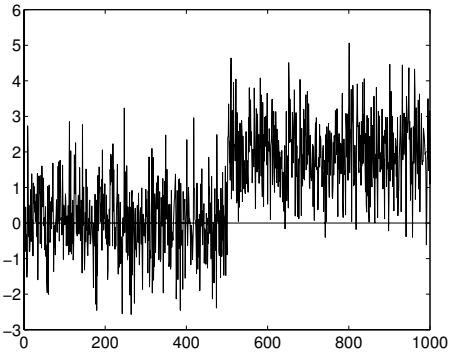


Fig. 6.8. Realisation of a sequence of independent random variables with distributions depicted in Fig. 6.7. Time on the x-axis is expressed in number of samples.

In Fig. 6.8, a realisation of a sequence of independent random variables with distribution $p_{\theta_0}(z)$ before $k = 500$ and $p_{\theta_1}(z)$ after $k = 500$ is depicted. Here $p_{\theta_0}(z)$ and $p_{\theta_1}(z)$ correspond to the Gaussian distributions $p_{\mu_0}(z)$ and $p_{\mu_1}(z)$ of Fig. 6.7. Figure 6.9 gives the evolution of $S(k)$, which behaves as expected. The difference between $S(k)$ and the minimum value of $S(j)$, $1 \leq j \leq k$ yields relevant information on the change. Hence the following decision function $g(k)$ is considered

$$g(k) = S(k) - m(k) \tag{6.120}$$

with $m(k) = \min_{1 \leq j \leq k} S(j)$. The stopping time (also called alarm time), k_a is the time instant at which $g(k)$ crosses a user defined positive threshold h. The fault occurrence time, k_0, can be estimated as the time instant $\hat{k}_0$ at which $S(k)$ has changed from negative to positive slope. It is formally expressed by

$$\hat{k}_0 = \operatorname{argmin}_{1 \leq j \leq k_a} S(j).$$

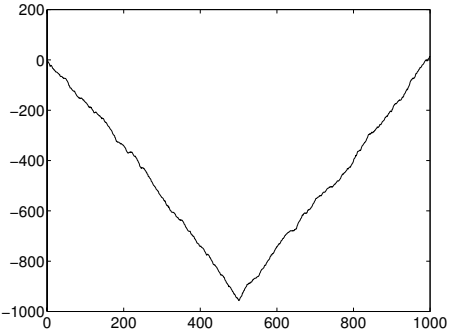


Fig. 6.9. Evolution of $S(k)$ for the sequence of Fig. 6.8, as a function of time in number of samples

The expression of the cumulative sum (6.119) can easily be computed for the distributions considered in Fig. 6.7, as shown in the example below.

Example 6.12 *Change in the mean of a Gaussian sequence*

Remember that the Gaussian probability density function for a random variable with mean μ and variance σ is

$$p_\mu(z) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right). \tag{6.121}$$

The resulting likelihood ratio for detecting a change in the mean from μ_0 to μ_1 is

$$\frac{p_{\mu_1}(z)}{p_{\mu_0}(z)} = \exp\left(-\frac{(z-\mu_1)^2}{2\sigma^2} + \frac{(z-\mu_0)^2}{2\sigma^2}\right).$$

Hence straightforward computations yield the following expression for the log-likelihood ratio $s(z)$:

$$s(z) = \frac{2(\mu_1-\mu_0)z + (\mu_0^2-\mu_1^2)}{2\sigma^2} = \frac{\mu_1-\mu_0}{\sigma^2}\left(z - \frac{\mu_0+\mu_1}{2}\right). \tag{6.122}$$

Figure 6.9 has been obtained by substituting (6.122) (with $z = z(i)$) for $s(z(i))$ in (6.119), which yields the algorithm depicted in the block diagram of Fig. 6.10. Notice that the signal to noise ratio $\frac{\mu_1-\mu_0}{\sigma^2}$ appears in (6.122), and it is thus automatically accounted for in the testing procedure. □

Example 6.13 *Change in the mean and variance*

If both mean and variance change after a fault, the following relation

$$\frac{p_{\mu_1}(z)}{p_{\mu_0}(z)} = \frac{\sigma_0}{\sigma_1} \exp\left(-\frac{(z-\mu_1)^2}{2\sigma_1^2} + \frac{(z-\mu_0)^2}{2\sigma_0^2}\right)$$

holds and the log-likelihood ratio is

$$s(z) = \ln\frac{\sigma_0}{\sigma_1} + \frac{(z-\mu_0)^2}{2\sigma_0^2} - \frac{(z-\mu_1)^2}{2\sigma_1^2}.$$

This general case is shown in the literature but not considered further in this context. □

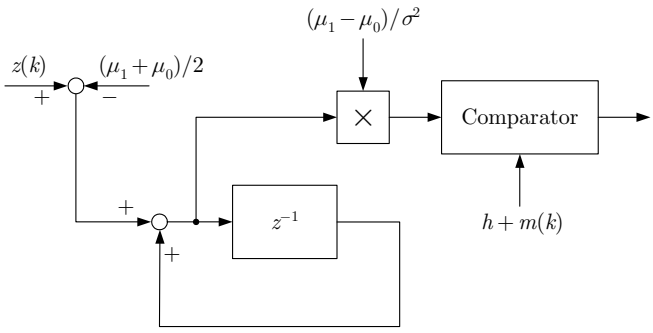


Fig. 6.10. Block diagram for the CUSUM test (6.119), (6.120), (6.122)

Remark 6.8 *Mean and variance of cumulative sum increments*
The mean μ_s, and the variance σ_s^2 of the cumulative sum increments (6.122) will be needed at a latter stage. They can be computed in a straightforward way from (6.122). Indeed,

$$\text{when} \quad \mu=\mu_0,\ E_{\mu_0}(z)=\mu_0 \qquad \text{and} \qquad \mu_s=E_{\mu_0}\left(s(z)\right)=-\frac{(\mu_1-\mu_0)^2}{2\sigma^2}\,,$$

$$\text{when} \quad \mu=\mu_1,\ E_{\mu_1}(z)=\mu_1 \qquad \text{and} \qquad \mu_s=E_{\mu_1}\left(s(z)\right)=\frac{(\mu_1-\mu_0)^2}{2\sigma^2}\,,$$

$$\sigma_s^2=\frac{(\mu_1-\mu_0)^2}{\sigma^2}. \quad \square$$

A more formal derivation of the CUSUM algorithm which is helpful for the subsequent description of the GLR algorithm is now presented. It is called the off-line statistical derivation, and it is based on the following re-formulation of the problem.

Problem 6.8 (Off-line statistical formulation) *Consider the sequence of independent random variables $z(1), \ldots, z(k)$ with probability density function $p_\theta(z)$ depending on one scalar parameter θ. Choose at time instant k between the hypotheses:*

$\mathcal{H}_0$*:* $\theta=\theta_0$ *for* $1\le i\le k$.
$\mathcal{H}_1$*:* $\theta=\theta_0$ *for* $1\le i\le k_0-1$ *and* $\theta=\theta_1$ *for* $k_0\le i\le k$, *where the time instant* k_0 *is unknown.*

From classical results in hypothesis testing due to Neyman and Pearson, it is known that tests to decide between $\mathcal{H}_0$ and $\mathcal{H}_1$ that are optimal in some sense are based on the log-likelihood ratio between both hypotheses. As k_0 is unknown, let j

be an hypothetical change time. The log-likelihood ratio between $\mathcal{H}_0$ and $\mathcal{H}_1$ with $k_0 = j$ is given as

$$\Lambda_1^k(j) = \frac{\prod_{i=1}^{j-1} p_{\theta_0}(z(i)) \prod_{i=j}^{k} p_{\theta_1}(z(i))}{\prod_{i=1}^{k} p_{\theta_0}(z(i))}. \tag{6.123}$$

The independence between the random variables $z(i)$, $i = 1, \ldots k$ was used to express $\Lambda_1^k(j)$ in terms of the marginal probability density function $p_\theta\,(z(i))$. From (6.123), the following cumulative sum of log-likelihood ratios is deduced:

$$S_j^k = \sum_{i=j}^{k} \ln \frac{p_{\theta_1}(z(i))}{p_{\theta_0}(z(i))}. \tag{6.124}$$

As the change time is unknown, the standard statistical approach consists in replacing it by its maximum likelihood estimate, namely, in looking for the value of j that maximises the numerator in (6.123). This is also the value of j that maximises (6.124). The log-likelihood ratio between $\mathcal{H}_0$ and $\mathcal{H}_1$ is thus estimated by $\max_{1\leq j\leq k} S_j^k$. The result due to Neyman and Pearson invoked above actually states that the optimal decision function for Problem 6.8 is

$$g(k) = \max_{1\leq j\leq k} \sum_{i=j}^{k} \ln \frac{p_{\theta_1}(z(i))}{p_{\theta_0}(z(i))} = \max_{1\leq j\leq k} \sum_{i=j}^{k} s(i) \tag{6.125}$$

and the optimal test consists of the following decisions.

$$\begin{array}{llll} \text{if} & g(k) \leq h & \text{accept} & \mathcal{H}_0 \\ \text{if} & g(k) > h & \text{accept} & \mathcal{H}_1. \end{array} \tag{6.126}$$

The way optimality is understood here involves several concepts. The reader should consult the reference section for precisions on this topic.

When $\mathcal{H}_1$ is accepted, an estimate of the change time is provided by:

$$\hat{k}_0 = \operatorname{argmax}_{\ 1\leq j\leq k_a} S_j^k,$$

where k_a is the alarm time, namely the value of k for which $g(k)$ crosses the threshold h.

The decision functions (6.120) and (6.125) are identical. Indeed, with reference to Fig. 6.9, $\sum_{i=j}^{k} \ln \frac{p_{\theta_1}(z(i))}{p_{\theta_0}(z(i)}$ is maximum when all the successive likelihood ratios which correspond to a positive slope on average are considered. This is precisely the way (6.120) was obtained.

An efficient way to implement the CUSUM algorithm is to use its recursive form. From (6.120) and Fig. 6.9 or from (6.125), and from the fact that the threshold h is positive, it is seen that only the contributions to the cumulative sum that add up to a positive number must be taken into account in order to determine the decision function. It justifies the following recursive computation of this function:

$$g(k) = \max(0,\, g(k-1) + s(z(k))). \tag{6.127}$$

To obtain an estimate of the fault occurrence time, the number of successive observations for which the decision function remains strictly positive is computed as:

$$N(k) = N(k-1)\, 1_{\{g(k-1)>0\}} + 1, \tag{6.128}$$

where $1_{\{x\}}$ is the indicator of event x, namely $1_{\{x\}} = 1$ when x is true, and $1_{\{x\}} = 0$ otherwise. An estimate for the fault occurrence time is then given as

$$\hat{k}_0 = k_a - N(k_a), \tag{6.129}$$

where k_a is the stopping or alarm time.

Example 6.12 (cont.) *Change in the mean of a Gaussian sequence*

Considering again the detection of a change in the mean of a Gaussian sequence, (6.127) together with (6.122) yields:

$$g(k) = \max(0,\, g(k-1) + \frac{\mu_1 - \mu_0}{\sigma^2}\, \left((z(k) - \frac{\mu_0 + \mu_1}{2})\right)) \tag{6.130}$$

which must be introduced in the decision logic (6.126). □

Remark 6.9 *Two-sided CUSUM algorithm*

Quite often, both positive and negative changes in the mean of a Gaussian sequence with mean μ_0 and variance σ^2 have to be detected. Letting β denote the magnitude of this change, the following two-sided CUSUM algorithm can be used for this purpose.

$$g^+(k) = \max\left(0,\, g^+(k-1) + z(k) - \mu_0 - \frac{\beta}{2}\right) \tag{6.131}$$

$$g^-(k) = \max\left(0,\, g^-(k-1) - z(k) + \mu_0 - \frac{\beta}{2}\right). \tag{6.132}$$

An alarm is generated when either $g^+(k)$ or $g^-(k)$ reaches the threshold $\bar{h} = h\sigma^2/\beta$. Notice that the factor $\frac{\mu_1-\mu_0}{\sigma^2}$ that appears in (6.130) has been cancelled from the decision functions $g^+(k)$ and $g^-(k)$ in (6.131) and 6.132). Equivalently, it is now included in the threshold $\bar{h}$. The expression for $g^-(k)$ is deduced from (6.130) by looking for a positive change in the mean of the sequence $-z(i), i = 1, 2, \ldots$. □

Parameter tuning for the CUSUM algorithm. In this section, the focus is on the case of a change in the mean, μ, of a Gaussian sequence.

Normally, the data associated to hypothesis $\mathcal{H}_0$ correspond to a fault free working mode. Hence parameter μ_0 can be estimated from a set of experimental data obtained in the absence of fault by taking the empirical mean of these data. The variance σ^2 can also be estimated in this way. The estimates are denoted $\hat{\mu}_0$ or $\hat{\sigma}^2$, respectively.

There are thus two design parameter left in the CUSUM algorithm, h and μ_1. Indeed, although the algorithm was derived under the hypothesis that μ_1 is known, this is seldom the case in practice. Nevertheless, the algorithm proved to be useful even when μ_1 is replaced by an approximate value.

The choice of the threshold h results from a compromise between the mean delay for detection and the mean time between false alarms. Exact computation of these quantities is involved, but approximate expressions and bounds are available. Both quantities can be determined from the so-called average run length (ARL) function defined as

$$L(\mu) = E_\mu\left(k_a\right)$$

for the example of the detection of a change in the mean of a Gaussian sequence with variance σ^2. The ARL function is thus the expected value of the alarm time instant when the data are distributed according to the normal probability density function with mean μ and variance σ^2. It is a function of the mean μ. When $\mu = \mu_0$ (data recorded in healthy conditions), the value of the ARL function $L(\mu_0)$ is equal to the mean time between false alarms, $\bar{T}$. On the other hand, $L(\mu_1)$ gives the mean delay for detection. An approximation for the ARL function in the situation were changes in the mean of a Gaussian sequence have to be detected is given by the following expression ([5], page 219)

$$\hat{L}(\mu_s) = \left(\exp\left[-2\left(\frac{\mu_s\, h}{\sigma_s^2} + 1.166\frac{\mu_s}{\sigma_s}\right)\right] - 1 + 2\left(\frac{\mu_s\, h}{\sigma_s^2} + 1.166\frac{\mu_s}{\sigma_s}\right)\right)\left(\frac{\sigma_s^2}{2\mu_s^2}\right) \tag{6.133}$$

for $\mu_s \neq 0$, where μ_s and σ_s are the mean or the standard deviation of the increments of the cumulative sum, respectively, as computed in Remark 6.8.

The mean time for detection, $\bar{\tau}$ can be estimated as

$$\hat{\bar{\tau}} = \hat{L}\left(\frac{(\mu_1 - \mu_0)^2}{2\sigma^2}\right) = \hat{L}\left(\frac{\beta^2}{2\sigma^2}\right), \tag{6.134}$$

where $\beta = \mu_1 - \mu_0$ and the estimated mean time between false alarms is obtained as

$$\hat{\bar{T}} = \hat{L}\left(-\frac{(\mu_1 - \mu_0)^2}{2\sigma^2}\right) = \hat{L}\left(-\frac{\beta^2}{2\sigma^2}\right). \tag{6.135}$$

When μ_1 (or equivalently β) is not known, it can be replaced by a user specified value such as the most likely magnitude of the change. A simple way to determine the test threshold from (6.134) or (6.135) is to plot $\hat{\bar{\tau}}$ and $\hat{\bar{T}}$ as a function of h. To this end, considering (6.134) for instance, $\frac{(\mu_1 - \hat{\mu}_0)^2}{2\hat{\sigma}^2}$ is substituted for μ_s and $\frac{(\mu_1 - \hat{\mu}_0)^2}{\hat{\sigma}^2}$ is substituted for σ_s in (6.133). Knowing the desired value for $\hat{\bar{\tau}}$ or $\hat{\bar{T}}$, one then determines from the plot an appropriate value for h. Alternatively, standard approaches such as the secant method can be used to solve the nonlinear equations (6.134) and (6.135).

Quite often, one can provide a minimum value of the change for which one wishes the algorithm to generate an alarm. Let $\beta_{\min}$ denote this value. It is then advisable to choose $\mu_1 = \hat{\mu}_0 + 2\beta_{\min}$. Indeed, let $p_{\hat{\mu}_0}(z)$ and $p_{\hat{\mu}_0 + 2\beta_{\min}}(z)$ denote the Gaussian probability density functions with respective mean $\hat{\mu}_0$ and $\hat{\mu}_0 + 2\beta_{\min}$ and with variance σ^2. It is easy to check that $p_{\hat{\mu}_0}(z) = p_{\hat{\mu}_0 + 2\beta_{\min}}(z)$ is achieved for $z =$

$\hat{\mu}_0 + \beta_{\min}$. Thus any sequence of values of z greater than $\hat{\mu}_0 + \beta_{\min}$ on average will yield a sequence of positive log-likelihood ratio $\ln \frac{p_{\hat{\mu}_0 + 2\beta_{\min}}(z)}{p_{\hat{\mu}_0}(z)}$ on average, and an alarm will be triggered after some time for such a sequence.

Remark 6.10 *Effect of an error on μ_1*

The objective of this remark is to illustrate that the CUSUM algorithm for detection of a change in the mean of a Gaussian sequence can detect changes even when μ_1 is overestimated. To this end, let us consider Fig. 6.11.

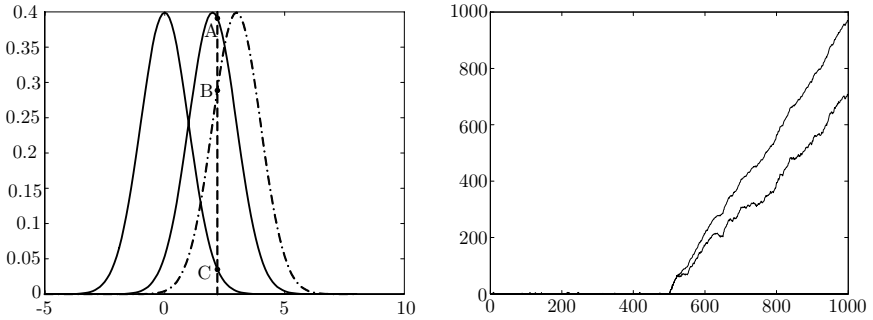


Fig. 6.11. (a) Gaussian probability density functions with actual (continuous line) and overestimated means (dash-dotted line), (b) evolution of the recursive CUSUM decision functions computed with the exact (continuous line) and approximated likelihood ratios (dash-dotted line) for the data sequence of Fig. 6.8

In the left hand figure, the density functions represented by continuous lines correspond to the actual data, which have mean $\mu_0 = 0$ before the change and mean $\mu_1 = 2$ after the change. Fig. 6.8 represents a data sequence which was generated from these density functions. The evolution of the recursive CUSUM decision function tuned with $\mu_0 = 0$ and $\mu_1 = 2$ obtained by processing the data of Fig. 6.8 is plotted as a continuous line in Fig. 6.11(b). Let us now process the same data with a CUSUM algorithm tuned with a mean value after change equal to 3 (instead of 2). The resulting decision function is represented by the dash-dotted line in Fig. 6.11(b). One notices that, when the value of μ_1 in the function $g(k)$ is higher than the real μ, the decision function still increases on average upon occurrence of a change, however the slope of the decision function is lower than with the correct value of μ_1. To understand this phenomenon, let us look again at Fig. 6.11(a), where the Gaussian density function with mean equal to 3 is plotted with a dash-dotted line. Let $p_0(z)$, $p_2(z)$ and $p_3(z)$ denote the density functions with mean 0, 2 or 3, respectively. After the change in the mean, a typical data sample from the actual data sequence, says $\tilde{z}$ will have a value in the neighbourhood of 2. The associated values of the density functions are represented by the points A ($p_2(\tilde{z})$), B($p_3(\tilde{z})$) or C ($p_0(\tilde{z})$), respectively. The contribution to the CUSUM decision function associated to $\tilde{z}$ is equal to $\frac{p_2(\tilde{z})}{p_0(\tilde{z})}$ when the correct tuning is used, and to $\frac{p_3(\tilde{z})}{p_0(\tilde{z})}$ when the μ_1 is overestimated. Both values are clearly larger than one, but $\frac{p_3(\tilde{z})}{p_0(\tilde{z})} < \frac{p_2(\tilde{z})}{p_0(\tilde{z})}$ which explains the lower slope of the CUSUM decision function when μ_1 is overestimated. □

The configuration and the implementation of the CUSUM algorithm to detect changes in the mean of a Gaussian sequence can be summarised as follows:

Algorithm 6.6 *CUSUM algorithm for detection of a change in the mean of a Gaussian sequence*

Given:

1. A set of experimental data $\{z(1), \ldots, z(N)\}$ obtained in fault free working mode.
2. β, corresponding to twice the minimum magnitude of the change to be detected or to the most likely magnitude of this change.
3. A specified mean time for detection or a specified mean time between false alarms.

Initialisation:

1. Determine $\hat{\mu}_0$ and $\hat{\sigma}^2$ from $\{z(1), \ldots, z(N)\}$.
2. Choose h to meet either the specified mean time for detection or the specified mean time between false alarms from (6.134) or (6.135).

At each sample time:

1. Acquire the new data $z(k)$.
2. Compute $g(k)$ by (6.130) and $N(k)$ by (6.128).
3. If $g(k) > h$, issue an alarm, provide an estimate of the change occurrence time $\hat{k}_0$ by (6.129) and reinitialise the decision function to 0.

Result: A sequence of alarm time instants k_a and estimated change occurrence times $\hat{k}_0$, for increasing time horizon k.

The reinitialisation after an alarm allows one to check whether the change in the mean persists as time elapses. More on this issue will be said when the algorithm will be used for fault detection applications.

Example 6.12 (cont.) *Change in the mean of a Gaussian sequence*

From the first 500 data samples plotted in Fig. 6.8, the following estimates were obtained

$$\hat{\mu}_0 = 0.0445 \qquad \hat{\sigma}^2 = 0.946.$$

Letting $\beta = 2$ yields $\hat{\mu}_1 = 2.0445$. Figure 6.12 gives the mean detection delay and the mean time between false alarms as a function of the threshold h, computed from (6.134) and (6.135) were the estimated values are substituted for μ_0, μ_1 and σ^2. The threshold $h = 10$ gives an estimated mean time between false alarms larger than 10^5 while assuring an estimated mean detection delay lower than 6 samples. Figure 6.13 gives a zoom of the decision function in the

vicinity of the change time (namely time 500). The alarm will be issued at time 503 which corresponds to a detection delay of three samples (of the order of magnitude of the estimated one). □

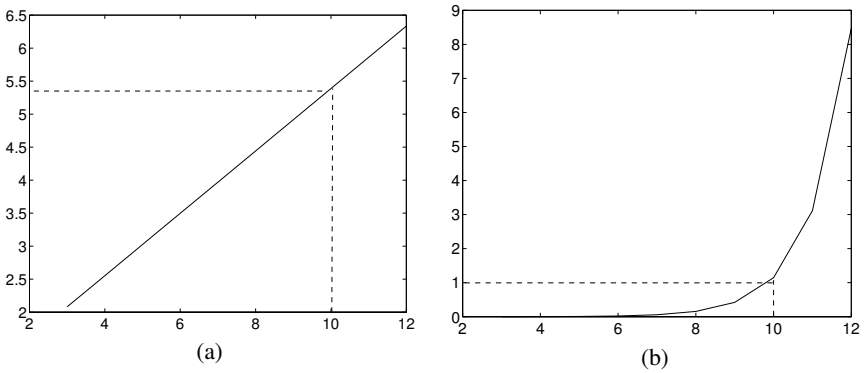


Fig. 6.12. Estimated mean detection delay in number of samples, as a function of h (a) and mean time between false alarms expressed in multiples of 10^5 samples as a function of h (b)

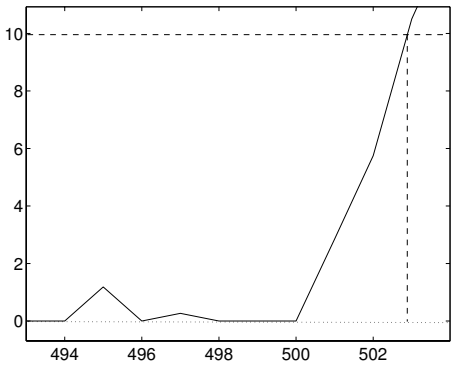


Fig. 6.13. Zoom on the decision function resulting from the recursive algorithm for the data of Fig. 6.8

Another option regarding the choice of θ_1 (the value of the parameter after change) consists in replacing it by the most likely value computed a posteriori from experimental data. This leads to the generalised likelihood ratio algorithm described in the next section.

Deriviation of the generalised likelihood ratio algorithm. The problem can be stated in a similar way as for the off-line derivation of the CUSUM algorithm (cf.

Problem 6.8), except that θ_1 is unknown. By the same reasoning as above, the log-likelihood between hypotheses $\mathcal{H}_0$ and $\mathcal{H}_1$, with an hypothetical change time j, can be computed as

$$S_j^k(\theta_1) = \sum_{i=j}^{k} \ln \frac{p_{\theta_1}(z(i))}{p_{\theta_0}(z(i))}. \tag{6.136}$$

For a given time instant k, it is a function of both j, the change time, and θ_1, the value of the parameter θ after the change. The standard statistical approach to estimate (6.136) is to replace j and θ_1 by their maximum likelihood estimates. The latter are obtained by solving the following double maximisation problem:

$$(\hat{k}_0, \hat{\theta}_1) = \arg \left\{ \max_{1 \le j \le k} \max_{\theta_1} S_j^k(\theta_1) \right\}, \tag{6.137}$$

where $\hat{k}_0$ denotes the estimate of the change time. The GLR decision function takes the form:

$$g(k) = \max_{1 \le j \le k} \max_{\theta_1} S_j^k(\theta_1). \tag{6.138}$$

The configuration and the implementation of the algorithm can be summarised as follows:

Algorithm 6.7 *GLR algorithm, scalar parameter*

Given:

1. A sequence of data $z(1), \ldots, z(k)$ with probability density function $p_\theta(z)$ depending on the scalar parameter θ.
2. A threshold h.

Compute: $g(k)$ using (6.136), (6.138).

Decide to

1. accept $\mathcal{H}_0$ if $g(k) \le h$.
2. accept $\mathcal{H}_1$ if $g(k) > h$.

Remark 6.11 *Maximum and supremum*

Rigorously, the symbol $\max_{\theta_1} S_j^k(\theta_1)$ should be replaced by $\sup_{\theta_1} S_j^k(\theta_1)$ (which gives the least upper bound of $S_j^k(\theta_1)$ with respect to θ_1). The reason is that the maximum may only be reached when θ_1 tends to infinity in pathological cases. However, for engineering purpose, there is no difference between "max" and "sup"; hence only the "max" operation will be used here. □

The maximisation in (6.138) is performed over all possible past time instants. As time elapses, the considered time span increases which induces an increasing search

duration for finding the optimum. To avoid that problem, the fault occurrence time is restricted to the last M time instants in practice. This amounts to assuming that the delay for detection is lower than M so that faults can be detected in M sampling periods at most. The actual decision function obtained from (6.138) is thus:

$$g(k) = \max_{k-M+1 \leq j \leq k} \max_{\theta_1} S_j^k(\theta_1). \tag{6.139}$$

Example 6.12 (cont.) *Change in the mean of a Gaussian sequence*

In this particular case, it is possible to find an explicit expression for $\hat{\mu}_1(k,\, j)$, the maximum likelihood estimate of μ_1 at the present time instant k, assuming that the fault occurred at time instant j. Indeed, from (6.122), $S_j^k(\mu_1)$ takes the following form:

$$S_j^k(\mu_1) = \frac{\mu_1 - \mu_0}{\sigma^2} \sum_{i=j}^{k} \left(z(i) - \frac{\mu_0 + \mu_1}{2} \right) \tag{6.140}$$

In order to maximise this expression with respect to μ_1, one has to take the derivative of $S_j^k(\mu_1)$ with respect to μ_1 and equate that expression to zero:

$$\frac{\partial S_j^k(\mu_1)}{\partial \mu_1} = \frac{1}{\sigma^2} \sum_{i=j}^{k} \left(z(i) - \frac{\mu_0 + \mu_1}{2} \right) - \frac{k-j+1}{2} \frac{(\mu_1 - \mu_0)}{\sigma^2} = 0. \tag{6.141}$$

Equation (6.141) yields:

$$\hat{\mu}_1(k,\, j) = \frac{1}{k-j+1} \sum_{i=j}^{k} z(i). \tag{6.142}$$

Substituting this expression for μ_1 in (6.140) results, after straightforward computations, in:

$$S_j^k(\hat{\mu}_1(k,\, j)) = \frac{1}{2\sigma^2} \frac{1}{k-j+1} \left[\sum_{i=j}^{k} (z(i) - \mu_0) \right]^2. \tag{6.143}$$

Hence the GLR decision function can be written:

$$g(k) = \frac{1}{2\sigma^2} \max_{k-M+1 \leq j \leq k} \frac{1}{k-j+1} \left[\sum_{i=j}^{k} (z(i) - \mu_0) \right]^2. \tag{6.144}$$

If $\mathcal{H}_1$ is accepted in the above GLR algorithm, at the alarm time k_a, the estimated change occurrence time is given as:

$$\hat{k}_0 = \arg\{ \frac{1}{2\sigma^2} \max_{k_a-M+1 \leq j \leq k_a} \frac{1}{k_a-j+1} \left[\sum_{i=j}^{k_a} (z(i) - \mu_0) \right]^2. \quad \square \tag{6.145}$$

Parameter tuning for the generlised likelihood ratio algorithm. An experimental approach will be considered here to adjust these parameters. Although the method is described with reference to Example 6.12, it can be generalised easily to other types of changes than jumps in the mean of a Gaussian sequence. First

M should be chosen larger or equal to the acceptable detection delay. Next, the threshold should be determined on the basis of healthy and faulty process data. A computation of the decision function based on a set of healthy data allows one to determine the typical range of values of this function in the absence of fault, and to set the threshold in such a way that false alarms are avoided (or the time between false alarms is acceptable) . This choice can then be validated by processing data obtained in faulty working mode, and checking that detection is achieved. Should experimental data corresponding to a faulty behaviour not be available, a simulator could possibly be used to obtain data that could be used as a substitute. An iterative adjustment of the horizon M and the threshold h may be needed to obtain the right compromise between false alarm rate and detection delay. Indeed, the lower M, the lower h has to be chosen in order to achieve detection in the window $[k-M+1,\ k]$. Decreasing the detection delay thus increases the false alarm rate. For the evaluation of the GLR decision function (6.144) required in the above procedure, empirical estimates $\hat{\mu}_0$ and $\hat{\sigma}^2$ should be substituted for μ_0 and σ^2.

Reinitilization. Again the particular case of a change in the mean of a Gaussian sequence is considered here. The reinitialisation allows one to detect a new change in the mean. The mean value of the data after change is thus considered as the new value of μ_0. This reinitialisation could use $\hat{\mu}_1(k_a, \hat{k}_0)$ as an estimate of the mean after the change occurred. However, if the delay for detection is short (one or a few samples), very few data are used to compute $\hat{\mu}_1(k_a, \hat{k}_0)$, and this estimate of the mean might be poor when the noise on the data is significant. It is the reason why the reinitialisation is based on a data set of fixed length obtained by collecting additional data. Here the length of the data set is chosen equal to M, but an additional parameter different from M might be introduced. It should be determined in such a way that a reliable estimate of the mean after change is obtained. More on this can be found in the appendix on random variables and stochastic processes, where the statistics of the empirical mean is studied.

With this in mind, the global GLR algorithm to detect a change in the mean of a Gaussian sequence can be summarised as follows.

Algorithm 6.8 *GLR algorithm to detect and estimate changes in the mean of a Gaussian sequence*

Given:

1. A set of data $\{z_0(1), \ldots, z_0(N_0)\}$ under hypothesis $\mathcal{H}_0$ and a set $\{z_1(1), \ldots, z_1(N_1)\}$ under hypothesis $\mathcal{H}_1$.
2. An acceptable maximum detection delay.

Initialisation:

1. Choose M larger than or equal to the maximum detection delay.
2. Determine $\hat{\mu}_0$ and $\hat{\sigma}^2$ from $\{z_0(1), \ldots, z_0(N_0)\}$.
3. Compute the decision function $g(i), i = M+1, \ldots, N_0$ for the data set $\{z_0(1), \ldots, z_0(N_0)\}$ by using (6.144) and choose the threshold h so that $g(i) < h,\ i = M+1, \ldots, N_0$.
4. Compute the decision function $g(i),\ i = M+1, \ldots, N_1$ for the data set $\{z_1(1), \ldots, z_1(N_1)\}$ by (6.144) and the estimated change magnitude by (6.142). Check that the fault is detected and that the delay for detection is acceptable for the estimated change magnitude.
5. Possibly iterate on the choice of M and h.
6. Acquire $M-1$ data samples.

At each sampling time:

R1. Acquire the new data $z(k)$.

R2. Compute $g(k)$ from (6.144).

R3. If $g(k) > h$, generate an alarm; provide the alarm time instant $k_a = k$, the estimate of the change occurrence time $\hat{k}_0$ by (6.145) ; and compute $\hat{\mu}_1(k_a, \hat{k}_0)$ by (6.142).

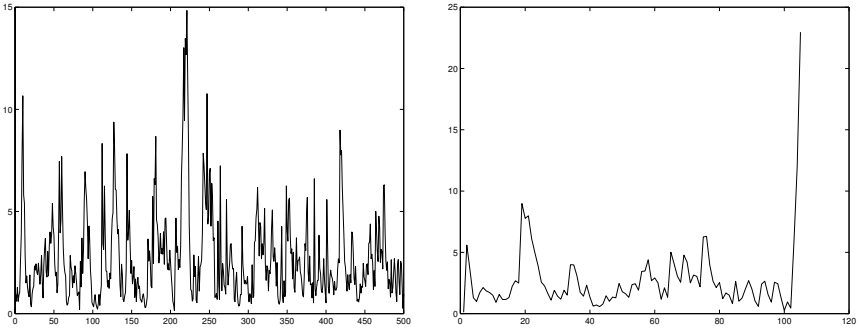
Reinitialisation:

1. Collect a set of M data from time $\hat{k}_0$ to $\hat{k}_0 + M - 1$.
2. Compute the new value of $\hat{\mu}_0$ from these data.
3. Restart the on-line algorithm from $k = \hat{k}_0 + M$ onwards (step R1).

Result: A sequence of alarm time instants k_a, estimated change occurrence times $\hat{k}_0$ and mean signal values $\hat{\mu}_1(k_a, \hat{k}_0)$, for increasing time horizon k.

Example 6.12 (cont.) *Change in the mean of a Gaussian sequence*

Consider again the data of Fig. 6.8. Let the set $z(1), \ldots, z(N_0)$ be made of the first 500 samples while $z(1), \ldots, z(N_1)$ consists of samples 400 to 1000. One gets, as before, $\hat{\mu}_0 = 0.0445$, $\hat{\sigma}^2 = 0.9455$. Figure 6.14 (left) gives the value of the GLR decision function obtained by processing the sequence $z(1), \ldots, z(N_0)$ with a window M of length 10 samples. A threshold above 15 appears to be suitable in this case. Hence, h is set to 20. Running the algorithm on the set $z(1), \ldots, z(N_1)$, one observes that an alarm is generated at time 105 (Fig. 6.14 (right)). The estimate of the change magnitude is 2.69, and the estimate of the change occurrence time is 103, while the actual change occurred at the 101^{st} sample in that set. Due to the noise on the signal, the estimate of the change magnitude is in error by 30%. This could be partly alleviated by increasing the threshold, so that more data are used to estimate the fault magnitude; this would increase the detection delay however. □



(a) (b)

Fig. 6.14. Two GLR decision functions

6.7.3 Sequential change detection: the vector case

The previous discussion dealt with detection of changes in a scalar signal. In the fault detection applications, the signal to be processed is issued by a residual generator, and it is generally a vector signal. The combined information comprised in this vector should be considered in our algorithm. Since fault detection will be reduced to the detection of changes in the mean of a Gaussian vector sequence, the solution to the following problem will be needed.

Problem 6.9 (Detection of a change in the mean of a Gaussian vector sequence) *Consider a sequence of n_z-dimensional random vectors $\boldsymbol{z}(1), \ldots, \boldsymbol{z}(k)$ that are independent and distributed as $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{Q})$, where $\boldsymbol{Q}$ is known, as well as the nominal value for $\boldsymbol{\mu}$, $\boldsymbol{\mu}_0$. Choose between the following two hypotheses:*

$\mathcal{H}_0 : \mathcal{L}(z(i)) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{Q}), \qquad i = 1, \ldots, k$

$\mathcal{H}_1$: *From time instant* 1 *up to an unknown time instant* k_0, $z(i)$, $i = 1,\ldots,k_0 - 1$ *is distributed as:*

$$\mathcal{L}(\boldsymbol{z}(i)) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{Q}) \tag{6.146}$$

while for time instants $i \geq k_0$*:*

$$\mathcal{L}(\boldsymbol{z}(i)) = \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{Q}) \tag{6.147}$$

with $\boldsymbol{\mu}_1 \neq \boldsymbol{\mu}_0$.

Beside detecting the possible change in the mean, one should also estimate its time of occurrence, and possibly its magnitude.

When $\boldsymbol{\mu}_1$ is known, the detection algorithm is a direct generalisation of the CUSUM algorithm. In the second situation which is considered in this paragraph, the change in $\boldsymbol{\mu}$ has known direction but unknown magnitude. This yields a GLR algorithm. Finally, the situation, where $\boldsymbol{\mu}_1$ is replaced by a dynamical profile of change will be considered, as this is a result needed at a later stage.

$\boldsymbol{\mu}_1$ known - CUSUM algorithm. By using the expression of the probability density function of a n-dimensional Gaussian vector $\boldsymbol{z}$ with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{Q}$

$$p_{\boldsymbol{\mu}}(\boldsymbol{z}) = \frac{1}{\sqrt{(2\pi)^n det\boldsymbol{Q}}} \exp\left(-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{\mu})' \boldsymbol{Q}^{-1}(\boldsymbol{z}-\boldsymbol{\mu})\right), \tag{6.148}$$

the following expression is obtained for the log-likelihood ratio associated to the above problem.

$$\begin{aligned} s(\boldsymbol{z}(k)) &= \ln \frac{p_{\boldsymbol{\mu}_1}(\boldsymbol{z}(k))}{p_{\boldsymbol{\mu}_0}(\boldsymbol{z}(k))} \\ &= -\frac{1}{2}(\boldsymbol{z}(k)-\boldsymbol{\mu}_1)' \boldsymbol{Q}^{-1}(\boldsymbol{z}(k)-\boldsymbol{\mu}_1) + \frac{1}{2}(\boldsymbol{z}(k)-\boldsymbol{\mu}_0)' \boldsymbol{Q}^{-1}(\boldsymbol{z}(k)-\boldsymbol{\mu}_0) \\ &= (\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)' \boldsymbol{Q}^{-1}\left(\boldsymbol{z}(k) - \frac{1}{2}(\boldsymbol{\mu}_0+\boldsymbol{\mu}_1)\right) \end{aligned} \tag{6.149}$$

This loglikelihood ratio is scalar, so the recursive computation of the CUSUM decision function can be performed in a similar way as for the scalar case (cf. (6.127))

$$g(k) = \max(0,\, g(k-1) + s(\boldsymbol{z}(k))). \tag{6.150}$$

The alarm or stopping time, k_a, is the smallest time instant at which $g(k)$ crosses a given threshold.

Known direction of change - GLR algorithm. Let $\boldsymbol{\mu}_1$ be of the form

$$\boldsymbol{\mu}_1 = \boldsymbol{\mu}_0 + \boldsymbol{\Gamma}\nu,$$

where $\boldsymbol{\Gamma}$ is a known vector, and ν is an unknown scalar change magnitude. Substituting this expression for $\boldsymbol{\mu}_1$ in (6.149) allows one to deduce the following expression of the cumulative sum $S_j^k(\nu)$, where j denotes an hypothetical value of the change time k_0,

$$
\begin{aligned}
S_j^k(\nu) &= \sum_{i=j}^{k} \ln \frac{p_{\boldsymbol{\mu}_0 + \boldsymbol{\Gamma}\nu}(\boldsymbol{z}(i))}{p_{\boldsymbol{\mu}_0}(\boldsymbol{z}(i))} \qquad (6.151)\\
&= \sum_{i=j}^{k} \left(\nu \boldsymbol{\Gamma}' \boldsymbol{Q}^{-1} (\boldsymbol{z}(i) - \boldsymbol{\mu}_0) - \frac{1}{2} \nu^2 \boldsymbol{\Gamma}' \boldsymbol{Q}^{-1} \boldsymbol{\Gamma} \right).
\end{aligned}
$$

Equating $\frac{\partial S_j^k(\nu)}{\partial \nu}$ to zero yields

$$
\begin{aligned}
\frac{\partial S_j^k(\nu)}{\partial \nu} &= \sum_{i=j}^{k} \boldsymbol{\Gamma}' \boldsymbol{Q}^{-1} (\boldsymbol{z}(i) - \boldsymbol{\mu}_0) - (k-j+1)\, \boldsymbol{\Gamma}' \boldsymbol{Q}^{-1} \boldsymbol{\Gamma} \nu \\
&= (k-j+1)\, \boldsymbol{\Gamma}' \boldsymbol{Q}^{-1} \left(\bar{\boldsymbol{Z}}_j^k - \boldsymbol{\mu}_0 \right) - (k-j+1)\, \boldsymbol{\Gamma}' \boldsymbol{Q}^{-1} \boldsymbol{\Gamma} \nu \qquad (6.152)\\
&= 0
\end{aligned}
$$

with $\bar{\boldsymbol{Z}}_j^k = \frac{1}{k-j+1} \sum_{i=j}^{k} \boldsymbol{z}(i)$.

Hence, the maximum likelihood estimate of ν at time k, assuming the fault occurred at time j is obtained from (6.152) as

$$
\hat{\nu}(k,\, j) = \frac{\boldsymbol{\Gamma}' \boldsymbol{Q}^{-1} (\bar{\boldsymbol{Z}}_j^k - \boldsymbol{\mu}_0)}{\boldsymbol{\Gamma}' \boldsymbol{Q}^{-1} \boldsymbol{\Gamma}}. \qquad (6.153)
$$

Substituting (6.153) for ν into (6.151) finally yields the GLR decision function in a similar way as (6.144) was deduced from (6.142) and (6.143)

$$
\begin{aligned}
g(k) &= \max_{k-M+1 \le j \le k} S_j^k(\hat{\nu}(k,\, j)) = \max_{k-M+1 \le j \le k} (k-j+1) \cdot \\
&\cdot \left(\hat{\nu}(k,\, j)\, \boldsymbol{\Gamma}' \boldsymbol{Q}^{-1} \left(\bar{\boldsymbol{Z}}_j^k - \boldsymbol{\mu}_0 \right) - \frac{1}{2} \hat{\nu}(k,\, j)^2\, \boldsymbol{\Gamma}' \boldsymbol{Q}^{-1} \boldsymbol{\Gamma} \right).
\end{aligned}
$$

The estimated fault occurrence time upon acceptance of hypothesis $\mathcal{H}_1$ at time instant k_a is given as

$$
\begin{aligned}
\hat{k}_0 &= \arg\{ \max_{k_a - M+1 \le j \le k_a} (k_a - j + 1) \cdot \\
&\cdot \left(\hat{\nu}(k_a,\, j) \boldsymbol{\Gamma}' \boldsymbol{Q}^{-1} \left(\bar{\boldsymbol{Z}}_j^k - \boldsymbol{\mu}_0 \right) - \frac{1}{2} \hat{\nu}(k_a, j)^2\, \boldsymbol{\Gamma}' \boldsymbol{Q}^{-1} \boldsymbol{\Gamma} \right) \}.
\end{aligned}
$$

Known dynamical profile of change - CUSUM algorithm. There is a need to generalise the previous result by replacing $\boldsymbol{\Gamma}\nu$ by a time-varying change direction, as this is precisely the situation which is encountered when detecting additive faults in linear systems. This leads to the following modified version of Problem 6.9.

Problem 6.10 (Change detection, known dynamical profile of change)
Consider a sequence of n_z-dimensional random vectors $\boldsymbol{z}(1), \ldots, \boldsymbol{z}(k)$ that are independent and distributed as $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{Q})$, where $\boldsymbol{Q}$ is known, as well as the nominal value for $\boldsymbol{\mu}$, $\boldsymbol{\mu}_0$. Choose between the following two hypotheses:

$\mathcal{H}_0 : \mathcal{L}(z(i)) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{Q}) \qquad i = 1, \ldots, k$

$\mathcal{H}_1$: *From time instant* 1 *up to an unknown time instant* k_0, $z(i)$, $i = 1, \ldots, k_0$–1 *is distributed as*

$$\mathcal{L}(\boldsymbol{z}(i)) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{Q}) \tag{6.154}$$

while for time instants $i \geq k_0$*:*

$$\mathcal{L}(\boldsymbol{z}(i)) = \mathcal{N}(\boldsymbol{\mu}_0 + \boldsymbol{\rho}(i - k_0), \boldsymbol{Q}), \tag{6.155}$$

where $\boldsymbol{\rho}(i - k_0)$ *is a known vector profile which is non-zero only for* $i \geq k_0$*.*

Beside detecting the possible change in the mean, one should also estimate its time of occurrence.

The cumulative sum for this problem setting, with j as an hypothetical value for k_0, is given as

$$\begin{aligned} S_j^k &= \sum_{i=j}^{k} \ln \frac{p_{\boldsymbol{\mu}_0 + \boldsymbol{\rho}(i-j)}(\boldsymbol{z}(i))}{p_{\boldsymbol{\mu}_0}(\boldsymbol{z}(i))} \\ &= \sum_{i=j}^{k} \left(-\frac{1}{2}(\boldsymbol{z}(i) - \boldsymbol{\mu}_0 - \boldsymbol{\rho}(i-j))' \boldsymbol{Q}^{-1} (\boldsymbol{z}(i) - \boldsymbol{\mu}_0 - \boldsymbol{\rho}(i-j)) \right) + \\ &\quad + \frac{1}{2} \sum_{i=j}^{k} (\boldsymbol{z}(i) - \boldsymbol{\mu}_0)' \boldsymbol{Q}^{-1} (\boldsymbol{z}(i) - \boldsymbol{\mu}_0) \\ &= \sum_{i=j}^{k} \boldsymbol{\rho}(i-j)' \boldsymbol{Q}^{-1} (\boldsymbol{z}(i) - \boldsymbol{\mu}_0) - \frac{1}{2} \sum_{i=j}^{k} \boldsymbol{\rho}(i-j)' \boldsymbol{Q}^{-1} \boldsymbol{\rho}(i-j) \end{aligned} \tag{6.156}$$

and the decision function, obtained in a similar way as for the scalar case, is given as (cf. (6.125))

$$g(k) = \max_{1 \leq j \leq k} S_j^k$$

with S_j^k as in (6.156).

This algorithm can be written in a recursive form ([5], pp. 283-284):

$$g(k) = \max(0, S(k)) \tag{6.157}$$

$$N(k) = N(k-1)\, 1_{\{g(k-1)>0\}} + 1 \tag{6.158}$$

$$\begin{aligned} S(k) &= S(k-1)\, 1_{\{g(k-1)>0\}} + \boldsymbol{\rho}(N(k)-1)' \boldsymbol{Q}^{-1} (\boldsymbol{z}(k) - \boldsymbol{\mu}_0) - \\ &\quad -0.5 \boldsymbol{\rho}(N(k)-1)' \boldsymbol{Q}^{-1} \boldsymbol{\rho}\, (N(k)-1). \end{aligned} \tag{6.159}$$

$N(k)$ is thus the number of observations after the last time instant for which the decision function g was null. Notice that this algorithm requires the hypothesis $\boldsymbol{\rho}(0) \neq 0$, which is included in the problem statement, otherwise the decision function would always remain equal to zero.

Example 6.14 *Data exhibiting a dynamical profile of change*

The aim of this example is to illustrate the type of vector signal on which the above algorithm can be applied. Consider a vector signal made of two components, z_1 and z_2. Suppose that the dynamical profile of the change in z_1 (z_2) can be modelled as the step response to a first order system with transfer function $\frac{0.5}{z-0.5}$ ($\frac{1.4}{z-0.3}$). In other words, the sequence $z_j(i)$, $i = 1, 2, \ldots,$ $j = 1, 2,$ takes the form

$$z_j(i) = z_j^0(i) + \rho_{s,j}(i - k_0)\, 1_{\{i \geq k_0\}}, \tag{6.160}$$

where

$$\mathcal{L}(z_1^0\,(i)) = \mathcal{L}(z_2^0\,(i)) = \mathcal{N}(0, 0.025),$$

hold and $\rho_{s,j}(\ell)$, which is tabulated below for $\ell = 0, \ldots, 9$, $j = 1, 2$, are the step responses (hence the index s) associated to $\frac{0.5}{z-0.5}$ and $\frac{1.4}{z-0.3}$. Figure 6.15 gives a realisation of the sequence (6.160) for $k_0 = 20$. □

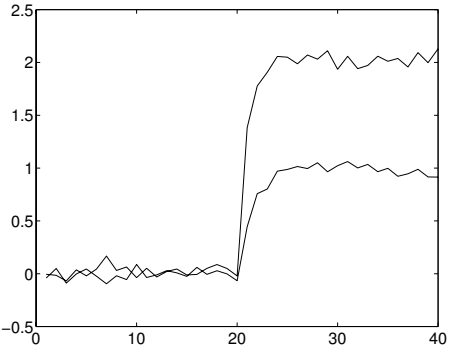


Fig. 6.15. Realisation of the vector sequence (6.160)

The superimposition of the deterministic step response and the stochastic sequence is clearly visible. To apply algorithm (6.157)-(6.159) to detect the change in the sequence, one should take $\rho_j(\ell) = \rho_{s,j}(\ell + 1)$, $\ell = 0, 1, 2 \ldots$, $j = 1, 2$ in order to assure that $\boldsymbol{\rho}(0)$ be nonzero.

Table 6.3 First 10 values of the dynamical profile of the change

ℓ	$\rho_{s,1}$	$\rho_{s,2}$
0	0	0
1	0.5000	1.4000
2	0.7500	1.8200
3	0.8750	1.9460
4	0.9375	1.9838
5	0.9688	1.9951
6	0.9844	1.9985
7	0.9922	1.9996
8	0.9961	1.9999
9	0.9980	2.0000

Known dynamical profile of change up to an unknown constant - GLR algorithm. Yet a more general situation occurs when the form of the dynamical profile of change is known, but its magnitude is not known.

Problem 6.11 (Change detection, known dynamical profile of change up to an unknown constant)
Consider a sequence of n_z-dimensional random vectors $\boldsymbol{z}(1), \ldots, \boldsymbol{z}(k)$ that are independent and distributed as $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{Q})$, where $\boldsymbol{Q}$ is known, as well as the nominal value for $\boldsymbol{\mu}$, $\boldsymbol{\mu}_0$. Choose between the following two hypotheses:

$\mathcal{H}_0 : \mathcal{L}(z(i)) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{Q}) \qquad i = 1, \ldots, k$

$\mathcal{H}_1$: *From time instant* 1 *up to an unknown time instant* k_0, $z(i)$, $i = 1, \ldots, k_0 - 1$ *is distributed as:*

$$\mathcal{L}(\boldsymbol{z}(i)) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{Q}) \tag{6.161}$$

while for time instants $i \geq k_0$

$$\mathcal{L}(\boldsymbol{z}(i)) = \mathcal{N}(\boldsymbol{\mu}_0 + \tilde{\boldsymbol{\rho}}\,(i - k_0)\nu, \boldsymbol{Q}), \tag{6.162}$$

where $\tilde{\boldsymbol{\rho}}\,(i - k_0)$ is a known vector profile which is non-zero only for $i \geq k_0$, k_0 is an unknown time instant, and ν is an unknown scalar.

Beside detecting the possible change in the mean, one should also estimate its time of occurrence and its magnitude ν.

The cumulative sum for this problem setting, with j as an hypothetical value for k_0, is given as

$$S_j^k(\nu) \quad = \quad \sum_{i=j}^{k} \ln \frac{p_{\boldsymbol{\mu}_0 + \tilde{\boldsymbol{\rho}}\,(i-j)\nu}(\boldsymbol{z}(i))}{p_{\boldsymbol{\mu}_0}(\boldsymbol{z}(i))}$$

$$= \quad \nu \sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)'\,\boldsymbol{Q}^{-1}\,(\boldsymbol{z}(i)-\boldsymbol{\mu}_0) - \frac{\nu^2}{2}\sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)'\,\boldsymbol{Q}^{-1}\tilde{\boldsymbol{\rho}}\,(i-j). \tag{6.163}$$

Similar computations as for the case of a constant direction of the parameter change yield the following maximum likelihood estimate of ν at time k, assuming the fault occurred at time j,

$$\hat{\nu}(k,\,j) = \frac{\displaystyle\sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)'\,\boldsymbol{Q}^{-1}\,(\boldsymbol{z}(i)-\boldsymbol{\mu}_0)}{\displaystyle\sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)'\,\boldsymbol{Q}^{-1}\tilde{\boldsymbol{\rho}}\,(i-j)} \tag{6.164}$$

and

$$\begin{aligned} g(k) \quad &= \quad \max_{k-M+1\le j\le k}\ \max_{\nu}\ S_j^k(\nu) \\ &= \quad \max_{k-M+1\le j\le k} \{\hat{\nu}(k,\,j)\sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)'\,\boldsymbol{Q}^{-1}\,(\boldsymbol{z}(i)-\boldsymbol{\mu}_0) \\ &\qquad -\frac{\hat{\nu}(k,\,j)^2}{2}\sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)'\,\boldsymbol{Q}^{-1}\tilde{\boldsymbol{\rho}}\,(i-j)\}. \end{aligned} \tag{6.165}$$

The estimated fault occurrence time upon acceptance of hypothesis $\mathcal{H}_1$ at time instant k_a is given as

$$\begin{aligned} \hat{k}_0 \quad &= \quad \arg\left\{ \max_{k_a-M+1\le j\le k_a}\ \hat{\nu}(k_a,\,j)\sum_{i=j}^{k_a} \tilde{\boldsymbol{\rho}}\,(i-j)'\,\boldsymbol{Q}^{-1}\,(\boldsymbol{z}(i)-\boldsymbol{\mu}_0) - \right. \\ &\qquad \left. -\frac{\hat{\nu}(k_a,\,j)^2}{2}\sum_{i=j}^{k_a} \tilde{\boldsymbol{\rho}}\,(i-j)'\,\boldsymbol{Q}^{-1}\tilde{\boldsymbol{\rho}}\,(i-j) \right\}. \end{aligned} \tag{6.166}$$

Parameter setting for the CUSUM algorithm. For the CUSUM algorithm associated to a known vector $\boldsymbol{\mu}_1$, the expressions of the ARL function (6.133) remains valid in the vector case. It suffices to replace μ_s and σ_s by:

$$\begin{aligned} \mu_s \quad &= \quad \pm\frac{(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)'\,\boldsymbol{Q}^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)}{2} \\ \sigma_s^2 \quad &= \quad (\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)'\,\boldsymbol{Q}^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0), \end{aligned}$$

where the plus or the minus sign are chosen according to the value of $\boldsymbol{\mu}$, the expected value of $\boldsymbol{z}(i)$ (cf. Remark 6.8). The mean time for detection and the mean time between false alarms can respectively be estimated as

$$\hat{\bar{\tau}} = \hat{L}\left(\frac{(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)' \boldsymbol{Q}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)}{2}\right) = \hat{L}\left(\frac{1}{2}\boldsymbol{\beta}' \boldsymbol{Q}^{-1}\boldsymbol{\beta}\right), \tag{6.167}$$

where $\boldsymbol{\beta} = \boldsymbol{\mu}_1 - \boldsymbol{\mu}_0$

$$\hat{\bar{T}} = \hat{L}\left(-\frac{(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)' \boldsymbol{Q}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)}{2}\right). \tag{6.168}$$

In order to evaluate the above expressions, given a set of experimental data recorded in fault free condition and a change magnitude $\boldsymbol{\beta}$, the empirical mean and variances $\hat{\boldsymbol{\mu}}_0$ and $\hat{\boldsymbol{Q}}$ should be substituted for $\boldsymbol{\mu}_0$ and $\boldsymbol{Q}$. These empirical estimates should also be used for the implementation of (6.149)-(6.150).

The algorithm for the vector parameter case is thus similar to the scalar case, namely

Algorithm 6.9 *CUSUM algorithm for Gaussian vector sequence, step-like change (μ_1 known)*

Given:

1. A set of experimental data $\{\boldsymbol{z}(1), \ldots, \boldsymbol{z}(N)\}$ obtained in fault free working mode.
2. $\boldsymbol{\beta}$, corresponding to twice the minimum magnitude of the change to be detected or to the most likely magnitude of this change.
3. A specified mean time for detection or a specified mean time between false alarms.

Initialisation:

1. Determine $\hat{\boldsymbol{\mu}}_0$ and $\hat{\boldsymbol{Q}}$ from $\{\boldsymbol{z}(1), \ldots, \boldsymbol{z}(N)\}$.
2. Choose h to meet either the specified mean time for detection or the specified mean time between false alarms from (6.167) or (6.168).

At each sample time:

1. Acquire the new data vector $\boldsymbol{z}(k)$.
2. Compute $g(k)$ by (6.150).
3. If $g(k) > h$, issue an alarm and reinitialise the decision function to 0.

Result: A sequence of alarm time instants k_a and estimated change occurrence times $\hat{k}_0$, for increasing time horizon k.

When the dynamical profile of the change is accounted for (as in (6.155)), the study of the properties of the algorithm such as mean delay for detection and mean time between false alarms becomes much more difficult. The difficulty stems from the fact that the increments in the cumulative sum of log-likelihood ratios are not identically distributed. Therefore, an experimental approach for setting the design parameters is proposed in the algorithm below.

Algorithm 6.10 *CUSUM algorithm for Gaussian vector sequence, known dynamical profile of change*

Given:

1. A set of data $\{\boldsymbol{z}_0(1), \ldots, \boldsymbol{z}_0(N_0)\}$ under hypothesis $\mathcal{H}_0$ and a set $\{\boldsymbol{z}_1(1), \ldots, \boldsymbol{z}_1(N_1)\}$ under hypothesis $\mathcal{H}_1$.
2. A dynamical profile of change $\boldsymbol{\rho}(i) \neq 0$, $i \geq 0$.

Initialisation:

1. Determine $\hat{\boldsymbol{\mu}}_0$ and $\hat{\boldsymbol{Q}}$ from $\{\boldsymbol{z}_0(1), \ldots, \boldsymbol{z}_0(N_0)\}$.
2. Compute the decision function $g(i), i = 1, \ldots, N_0$ for the data set $\{\boldsymbol{z}_0(1), \ldots, \boldsymbol{z}_0(N_0)\}$ by (6.157) – (6.159) and choose the threshold h so that
   $g(i) < h$, $i = 1, \ldots, N_0$.
4. Compute the decision function $g(i)$, $i = 1, \ldots, N_1$ for the data set $\{\boldsymbol{z}_1(1), \ldots, \boldsymbol{z}_1(N_1)\}$ by (6.157) – (6.159); check that the fault is detected and that the delay for detection is acceptable.
6. Possibly iterate on the choice of h.

At each sampling time:

R1. Acquire the new data $\boldsymbol{z}(k)$.

R2. Compute $g(k)$ from (6.157) – (6.159).

R3. If $g(k) > h$, generate an alarm by setting $k_a = k$, and provide an estimate of the change occurrence time as $k_a - N(k_a)$ by (6.158).

Reinitialisation:

1. Reset $g(k_a)$, $N(k_a)$, and $S(k_a)$ to zero in (6.157) – (6.159).
2. Restart the recursive algorithm with (step R1).

Result: A sequence of alarm time instants k_a and estimated fault occurrence times $\hat{k}_0$, for increasing time horizon k.

Remark 6.12 *Reinitialisation procedure*
The reinitialisation may depend on what one wishes to detect. Here it is assumed that one wishes to check whether the observed change remains present. By reinitialising the algorithm

as proposed, repeated alarms will occur as long as the change is present in the signal. Another option for reinitialisation is to change the sign of the loglikelihood ratio which amounts to changing the sign of the last two terms in (6.159) and to reset all variables to zero as indicated is step 1 of the reinitialisation. In this way a return to normal will generate an alarm.

The proposed reinitilisation policies require that the dynamical profile of change does not asymptotically vanish. Should this not hold, one should resort to a GLR algorithm as illustrated in the ship example in Section 6.8.4 □

An example of application of this algorithm in the framework of a fault detection system is given in Section 6.8.2.

GLR algorithm Here also an experimental approach is used to set the design parameters. The algorithm is only presented for a change characterised by a dynamical profile with unknown magnitude.

Algorithm 6.11 *GLR algorithm, known dynamical profile but unknown change magnitude*

Given:

1. A set of data $\{\boldsymbol{z}_0(1), \ldots, \boldsymbol{z}_0(N_0)\}$ under hypothesis $\mathcal{H}_0$ and a set $\{\boldsymbol{z}_1(1), \ldots, \boldsymbol{z}_1(N_1)\}$ under hypothesis $\mathcal{H}_1$.
2. A dynamical profile of change $\tilde{\boldsymbol{\rho}}(i)$, $i \geq 0$.

Initialisation:

1. Choose M larger than or equal to largest admissible detection delay.
2. Determine $\hat{\boldsymbol{\mu}}_0$ and $\hat{\boldsymbol{Q}}$ from $\{\boldsymbol{z}_0(1), \ldots, \boldsymbol{z}_0(N_0)\}$.
3. Compute the decision function $g(i)$, $i = M+1, \ldots, N_0$ for the data set $\{\boldsymbol{z}_0(1), \ldots, \boldsymbol{z}_0(N_0)\}$ by (6.164), (6.165)), and choose the threshold h so that $g(i) < h$, $i = M+1, \ldots, N_0$.
4. Compute the decision function $g(i)$, $i = M+1, \ldots, N_1$ for the data set $\{\boldsymbol{z}_1(1), \ldots, \boldsymbol{z}_1(N_1)\}$ by (6.165) and the estimated change magnitude by (6.164); check that the fault is detected and that the delay for detection is acceptable given the estimated change magnitude.
5. Possibly iterate on the choice of M and h.
6. Acquire $M-1$ data samples.

At each sampling time:

R1. Acquire the new data $\boldsymbol{z}(k)$.

R2. Compute $g(k)$ from (6.164), (6.165)

R3. If $g(k) > h$, generate an alarm; provide the estimate of the change occurrence time, $\hat{k}_0$, by (6.166), and the estimated change magnitude $\hat{\nu}(k, \hat{k}_0)$ computed by (6.164).

Reinitialisation:

1. Collect a set of M data from time $\hat{k}_0$ to $\hat{k}_0 + M - 1$.
2. Compute the estimated change magnitude $\hat{\boldsymbol{\nu}}(\hat{k}_0 + M - 1,\, \hat{k}_0)$ for these data by (6.164).
3. Restart the recursive algorithm from $k = \hat{k}_0 + M$ onwards with step R1. Notice that the mean value to be subtracted from $\boldsymbol{z}(i)$ in (6.164), (6.165) should now be $\hat{\boldsymbol{\mu}}_0 + \hat{\nu}(\hat{k}_0 + M - 1,\, \hat{k}_0)\tilde{\boldsymbol{\rho}}(i - \hat{k}_0)$ (cf. Remark 6.13 below).

Result: A sequence of alarm time instants k_a, estimated change occurrence times $\hat{k}_0$ and change magnitudes $\hat{\nu}(k, \hat{k}_0)$.

Remark 6.13 *Reinitialisation for GLR algorithm*

- The reason for collecting a set of M data to estimate $\hat{\nu}(\hat{k}_0 + M - 1,\, \hat{k}_0)$ is to assure a sufficient precision of the change magnitude so that updating the mean of the signal is performed properly.
- After reinitialisation, (6.164) and (6.165) should be replaced in step R2 by the following expression which accounts for the estimated mean of the signal after change

$$\hat{\nu}(k,\, j) \;=\; \frac{\displaystyle\sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)'\,\boldsymbol{Q}^{-1}\left(\boldsymbol{z}(i) - \hat{\boldsymbol{\mu}}_0 - \hat{\nu}\tilde{\rho}(i - \hat{k}_0)\right)}{\displaystyle\sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)'\,\boldsymbol{Q}^{-1}\tilde{\boldsymbol{\rho}}\,(i-j)} \tag{6.169}$$

$$g(k) \;=\; \max_{k-M+1 \le j \le k} \left\{ \hat{\nu}(k,\, j) \sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)'\,\boldsymbol{Q}^{-1}\left(\boldsymbol{z}(i) - \hat{\boldsymbol{\mu}}_0 - \hat{\nu}\tilde{\rho}(i - \hat{k}_0)\right) - \right.$$
$$\left. - \frac{\hat{\nu}(k,\, j)^2}{2} \sum_{i=j}^{k} \tilde{\boldsymbol{\rho}}\,(i-j)'\,\boldsymbol{Q}^{-1}\tilde{\boldsymbol{\rho}}\,(i-j) \right\}, \tag{6.170}$$

where $\hat{\nu}$ stands for $\hat{\nu}(\hat{k}_0 + M - 1,\, \hat{k}_0)$. □

The method will be illustrated in Section 6.8.3, as a part of a fault detection and estimation system.

6.8 Stochastic model – Kalman filter approach

After a presentation of the model of the supervised process, the problems of detection, isolation and estimation of additive faults in a stochastic system will be successively considered in this section.

6.8.1 Model

Let us consider a system described by a linear discrete-time model of the form

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{F}_d\,\boldsymbol{d}(k) + \boldsymbol{F}_f\,\boldsymbol{f}(k) + \boldsymbol{w}(k) \\ \boldsymbol{x}(0) &= \boldsymbol{x}_0 \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) + \boldsymbol{E}_d\,\boldsymbol{d}(k) + \boldsymbol{E}_f\,\boldsymbol{f}(k) + \boldsymbol{v}(k), \end{aligned} \tag{6.171}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{u} \in \mathbb{R}^m$, $\boldsymbol{y} \in \mathbb{R}^p$ are respectively the state vector, the vector of known input signals and the vector of measured output signals, $\boldsymbol{w}$ is the vector of state noise, $\boldsymbol{v}$ denotes the measurement noise. $\boldsymbol{w}(k)$ and $\boldsymbol{v}(k)$ are samples of vector white noise sequences with zero mean and covariance matrix:

$$E\left[\begin{pmatrix} \boldsymbol{w}(k) \\ \boldsymbol{v}(k) \end{pmatrix} (\boldsymbol{w}(\ell)'\boldsymbol{v}(\ell)')\right] = \begin{pmatrix} \boldsymbol{Q}_w & \boldsymbol{Q}_{wv} \\ \boldsymbol{Q}'_{wv} & \boldsymbol{Q}_v \end{pmatrix} \delta_{k\ell}.$$

$\boldsymbol{x}_0$ is a stochastic vector with mean $\boldsymbol{m}_0$ and variance $\boldsymbol{\Pi}_0$ uncorrelated with the state and measurement noise sequences. Finally, $\boldsymbol{d} \in \mathbb{R}^{n_d}$ is a vector of unknown input signals or disturbances (deterministic or stochastic with non-zero mean), and $\boldsymbol{f} \in \mathbb{R}^{n_f}$ is a vector of unknown input signals representing the faults to be detected. The faults are said to be additive, since they enter linearly in the model as additional input.

Such a model can also be written in terms of a single vector white noise sequence, with variance equal to the identity matrix by considering the factorisation

$$\begin{pmatrix} \boldsymbol{Q}_w & \boldsymbol{Q}_{wv} \\ \boldsymbol{Q}'_{wv} & \boldsymbol{Q}_v \end{pmatrix} = \begin{pmatrix} \boldsymbol{B}_\epsilon \\ \boldsymbol{D}_\epsilon \end{pmatrix} \begin{pmatrix} \boldsymbol{B}'_\epsilon & \boldsymbol{D}'_\epsilon \end{pmatrix}.$$

A sample of this sequence will be denoted $\boldsymbol{\epsilon}(k)$, hence the index in $\boldsymbol{B}_\epsilon$ and $\boldsymbol{D}_\epsilon$. It is a n_ϵ-dimensional random vector, where n_ϵ is the rank of the variance of $(\boldsymbol{w}(k)' \quad \boldsymbol{v}(k)')'$, generally equal to $n+p$. The state-space model (6.171) can thus be rewritten as

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{F}_d\,\boldsymbol{d}(k) + \boldsymbol{F}_f\,\boldsymbol{f}(k) + \boldsymbol{B}_\epsilon\boldsymbol{\epsilon}(k) \\ \boldsymbol{x}(0) &= \boldsymbol{x}_0 \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) + \boldsymbol{E}_d\,\boldsymbol{d}(k) + \boldsymbol{E}_f\,\boldsymbol{f}(k) + \boldsymbol{D}_\epsilon\boldsymbol{\epsilon}(k). \end{aligned} \tag{6.172}$$

6.8.2 Fault detection

Problem statement. Fault detection amounts to determining whether the supervised process is working in a normal (or healthy) operating mode. The problem can be stated as follows.

Problem 6.12 (Fault detection)
Given

- *a model of the process of the form (6.171) or (6.172)*
- *a sequence of measured process input and output*

$$(\boldsymbol{y}(i), \boldsymbol{u}(i))_{1 \leq i \leq k},$$

where k denotes the current time instant.

Choose between the following two hypotheses:

$\mathcal{H}_0$*: healthy operating condition,*
$\mathcal{H}_1$*: faulty operating condition.*

The quality of a fault detection system is measured in terms of detection delay and time between false alarms. A typical objective is to minimise the mean delay for detection of a change subject to a fixed false alarm rate before the change time.

To achieve this goal, the task is usually divided into two parts: residual generation and residual evaluation. Each of them is addressed successively in the following subsections.

Residual generation. As in the deterministic case, the residual generators are filters with input signals $\boldsymbol{u}$ and $\boldsymbol{y}$, belonging to the following class of linear time invariant systems

$$\begin{aligned} \boldsymbol{z}(k+1) &= \boldsymbol{A}_z\, \boldsymbol{z}(k) + \boldsymbol{B}_{zu}\, \boldsymbol{u}(k) + \boldsymbol{B}_{zy}\, \boldsymbol{y}(k), \qquad \boldsymbol{z}(0) = \boldsymbol{z}_0 \\ \boldsymbol{r}(k) &= \boldsymbol{C}_{rz}\, \boldsymbol{z}(k) + \boldsymbol{D}_{ru}\, \boldsymbol{u}(k) + \boldsymbol{D}_{ry}\, \boldsymbol{y}(k) \end{aligned} \tag{6.173}$$

or, in transfer function form, after taking the z-transform of the above equations and assuming zero initial conditions:

$$\begin{aligned} \boldsymbol{r}(z) &= \boldsymbol{V}_{ru}(z)\boldsymbol{u}(z) + \boldsymbol{V}_{ry}(z)\boldsymbol{y}(z) \\ &= (\boldsymbol{V}_{ru}(z) \;\; \boldsymbol{V}_{ry}(z)) \begin{pmatrix} \boldsymbol{u}(z) \\ \boldsymbol{y}(z) \end{pmatrix}. \end{aligned} \tag{6.174}$$

The problem of designing a residual generator can be stated as follows.

Problem 6.13 (Residual generator design for fault detection)
Determine a stable linear time-invariant filter (6.173) or (6.174) such that:

1. *The sequence of output values $\boldsymbol{r}(k)$, $k = 1, 2, \ldots$ is a zero mean white noise vector sequence (which is not affected by $\boldsymbol{u}$ and $\boldsymbol{d}$), once the transient due to initial conditions has vanished.*

2. *In the presence of a fault ($\boldsymbol{f}(k) \neq 0$ for all $k \geq k_0$), the mean of $\boldsymbol{r}(k)$ is different from zero for at least some $k \geq k_0$.*

As $\boldsymbol{u}$ and $\boldsymbol{d}$ are arbitrary signals, they cannot affect $\boldsymbol{r}$ for the latter to be a white noise sequence. To define rigorously what is meant by this statement, notice that the global system made of the supervised process and the residual generator, obtained by combining Eqs. (6.172) and (6.173), is seen to have as input signals $\boldsymbol{u}$, $\boldsymbol{d}$, $\boldsymbol{f}$, $\boldsymbol{\epsilon}$, and state $(\boldsymbol{x} \;\; \boldsymbol{z})'$. Hence the residual at time k can be considered as a function of the above input and the initial state, namely:

$$\boldsymbol{r}(k) = \boldsymbol{r}(k;\, \boldsymbol{u},\, \boldsymbol{d},\, \boldsymbol{f},\, \boldsymbol{\epsilon};\, \boldsymbol{x}_0,\, \boldsymbol{z}_0).$$

Saying that the residual is not affected by $\boldsymbol{u}$ and $\boldsymbol{d}$ means that, for any two distinct input sequences $\boldsymbol{u}^1(k), \boldsymbol{u}^2(k)$ and $\boldsymbol{d}^1(k), \boldsymbol{d}^2(k), k = 1, 2, \ldots,$

$$\boldsymbol{r}\left(k;\, \boldsymbol{u}^1,\, \boldsymbol{d},\, \boldsymbol{f},\, \boldsymbol{\epsilon};\, \boldsymbol{x}_0,\, \boldsymbol{z}_0\right) = \boldsymbol{r}\left(k;\, \boldsymbol{u}^2,\, \boldsymbol{d},\, \boldsymbol{f},\, \boldsymbol{\epsilon};\, \boldsymbol{x}_0,\, \boldsymbol{z}_0\right)$$

and

$$\boldsymbol{r}\left(k;\, \boldsymbol{u},\, \boldsymbol{d}^1,\, \boldsymbol{f},\, \boldsymbol{\epsilon};\, \boldsymbol{x}_0,\, \boldsymbol{z}_0\right) = \boldsymbol{r}\left(k;\, \boldsymbol{u},\, \boldsymbol{d}^2,\, \boldsymbol{f},\, \boldsymbol{\epsilon};\, \boldsymbol{x}_0,\, \boldsymbol{z}_0\right),$$

whatever the initial state and the other input sequences.

One way to solve the problem is to design a filter which meets the first condition and then to check whether the second requirement is fulfilled. In order to maximise the chances for this second condition to hold, one should make sure that, when imposing condition 1, no useful information contained in $\boldsymbol{y}$ is lost. Only the information corrupted by an unknown input should be cancelled. A filter that meets the latter condition, together with the first condition of Problem 6.13 is called an innovation filter for reasons that will be clarified in the next subsection.

To construct a residual generator, one will first solve the innovation filter design problem below. Next fault sensitivity of the filter output will be checked to see whether condition 2 is met in Problem 6.13. In the affirmative, the innovation filter is a residual generator. The issue of fault sensitivity is the object of a specific subsection.

Problem 6.14 (Innovation filter design)
Determine a stable linear time-invariant filter (6.174) with the least number of output signals such that, in the absence of fault (i.e. $\boldsymbol{f}(k) = 0$ for all k):

1. *The sequence of output values $\boldsymbol{r}(k)$, $k = 1, 2, \cdots$ is a zero mean white noise vector sequence which is not affected by $\boldsymbol{u}$ and $\boldsymbol{d}$, once the transient due to initial conditions has vanished.*
2. *No information on the fault contained in $\boldsymbol{y}$ is lost, except if it is affected by the unknown input vector $\boldsymbol{d}$.*

An observer-based approach will be used to solve the Problem 6.14.

Two situations have to be distinguished, namely the absence of unknown input ($\boldsymbol{E}_d = \boldsymbol{O}$ and $\boldsymbol{F}_d = \boldsymbol{O}$) and the presence of unknown input.

No unknown input. In this situation the design of an innovation filter amounts to the design of a steady state Kalman filter. Such a filter provides a prediction of the output $\boldsymbol{y}(k)$, $\hat{\boldsymbol{y}}(k)$, given the data up to time $k-1$, namely, given $\boldsymbol{u}(i)$, $\boldsymbol{y}(i)$, $i = 1, 2, \ldots, k-1$. The output prediction error, $\boldsymbol{r}(k) = \boldsymbol{y}(k) - \hat{\boldsymbol{y}}(k)$ is called the innovation in standard Kalman filter literature, because it consists of the new information contained in $\boldsymbol{y}(k)$, which was not available in $\boldsymbol{y}(1), \ldots, \boldsymbol{y}(k-1)$. The innovation sequence is known to be a white noise sequence not affected by $\boldsymbol{u}$ (once the transient due to initial conditions has decayed to zero). Hence it fulfils condition 1 of Problem 6.14. Besides, the information about $\boldsymbol{f}$ contained in the sequence of data is also contained in the innovation sequence. For this reason, the innovation is said to be a sufficient statistics for the fault vector $\boldsymbol{f}$. Thus condition 2 of Problem 6.14 is also fulfilled by the innovation sequence, and hence the latter is a suitable candidate as a residual sequence. It is the fact that the innovation sequence meets conditions 1 and 2 of Problem 6.14 that justifies the name innovation filter.

To state the design procedure precisely, let us introduce the notion of a regular quadruple $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$.

Definition 6.6 (Regular quadruple $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$)
$(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ is regular if the matrix

$$\begin{pmatrix} -z\,\boldsymbol{I} + \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{C} & \boldsymbol{D} \end{pmatrix}$$

has full row rank for all z on the unit cycle ($z = \exp(j\omega)$, $\omega \in \mathbb{R}$).

It is assumed below that the pair $(\boldsymbol{C}, \boldsymbol{A})$ is detectable, and $(\boldsymbol{A}, \boldsymbol{B}_\epsilon, \boldsymbol{C}, \boldsymbol{D}_\epsilon)$ is regular.

Remark 6.14 *Uncorrelated state and measurement noise sequences*
In the particular case where the sequence $\boldsymbol{w}(k)$ and $\boldsymbol{v}(k)$ are uncorrelated (which amounts to $\boldsymbol{B}_\epsilon \boldsymbol{D}_\epsilon' = \boldsymbol{O}$), the above regularity assumption can be replaced by the classical requirement that the pair $(\boldsymbol{A}, \boldsymbol{B}_\epsilon)$ has no uncontrollable mode on the unit circle. □

Under such hypotheses, the equations for the steady state Kalman filter can be written as

$$\begin{aligned} \hat{\boldsymbol{x}}(k+1) &= \boldsymbol{A}\hat{\boldsymbol{x}}(k) + \boldsymbol{B}\boldsymbol{u}(k) - \boldsymbol{K}(\boldsymbol{y}(k) - \boldsymbol{C}\hat{\boldsymbol{x}}(k) - \boldsymbol{D}\boldsymbol{u}(k)), \\ \hat{\boldsymbol{x}}(0) &= \hat{\boldsymbol{x}}_0 \\ \boldsymbol{r}(k) &= \boldsymbol{y}(k) - \boldsymbol{C}\hat{\boldsymbol{x}}(k) - \boldsymbol{D}\boldsymbol{u}(k), \end{aligned} \tag{6.175}$$

where the filter gain $\boldsymbol{K}$ is given by

$$\boldsymbol{K} = -(\boldsymbol{A}\boldsymbol{P}\boldsymbol{C}' + \boldsymbol{Q}_{wv})(\boldsymbol{Q}_v + \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}')^{-1}, \tag{6.176}$$

$\boldsymbol{P}$ being the symmetric semi-positive definite solution of the following discrete algebraic Riccati equation

$$\begin{aligned} \boldsymbol{P} &= \boldsymbol{A}\boldsymbol{P}\boldsymbol{A}' - (\boldsymbol{A}\boldsymbol{P}\boldsymbol{C}' + \boldsymbol{Q}_{wv})(\boldsymbol{Q}_v + \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}')^{-1} \cdot \\ &\quad \cdot (\boldsymbol{C}\boldsymbol{P}\boldsymbol{A}' + \boldsymbol{Q}'_{wv}) + \boldsymbol{Q}_w. \end{aligned} \tag{6.177}$$

Fig. 6.16 illustrates the state-space implementation of the innovation filter.

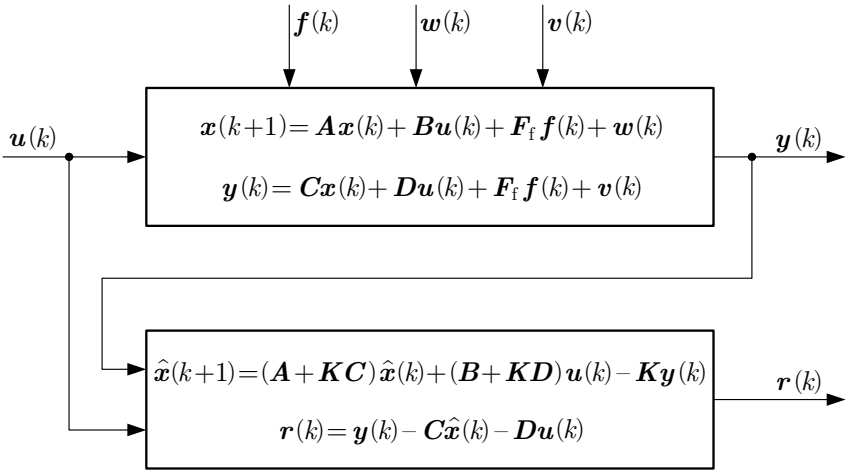


Fig. 6.16. Block diagram of the supervised system together with the innovation filter

In a transfer function form, this filter is described by

$$\begin{aligned} \boldsymbol{r}(z) &= \boldsymbol{V}_{ru}(z)\boldsymbol{u}(z) + \boldsymbol{V}_{ry}(z)\boldsymbol{y}(z) \\ &= (-\boldsymbol{C}(z\boldsymbol{I} - \boldsymbol{A} - \boldsymbol{K}\boldsymbol{C})^{-1}(\boldsymbol{B} + \boldsymbol{K}\boldsymbol{D}) - \boldsymbol{D})\boldsymbol{u}(z) + \\ &\quad + (\boldsymbol{C}(z\boldsymbol{I} - \boldsymbol{A} - \boldsymbol{K}\boldsymbol{C})^{-1}\boldsymbol{K} + \boldsymbol{I})\boldsymbol{y}(z)\,. \end{aligned} \tag{6.178}$$

If the pair $(\boldsymbol{C},\ \boldsymbol{A})$ is not detectable, it is still possible to design an innovation filter by extracting the observable part of system (6.172) and designing a Kalman filter for this observable subsystem. Notice that this approach can also be considered when (6.172) is detectable but not observable, if one wishes to obtain a residual generator with the lowest possible order.

Remark 6.15 *Time varying Kalman filter*
To assure coherency with Sections 6.3 and 6.4 and to ease the study of the sensitivity to the fault, a steady state Kalman filter is considered above. This implies that whiteness of the sequence $\boldsymbol{r}(k)$ is only reached after the transient has vanished. A white noise sequence can be generated from time $k = 0$, if a (time-varying) Kalman filter is used instead of a steady state Kalman filter. Equation (6.175), (6.176) and (6.177) are then replaced by

$$\begin{aligned} \hat{\boldsymbol{x}}(k+1) &= \boldsymbol{A}\hat{\boldsymbol{x}}(k) + \boldsymbol{B}\boldsymbol{u}(k) - \boldsymbol{K}(k)(\boldsymbol{y}(k) - \boldsymbol{C}\hat{\boldsymbol{x}}(k) - \boldsymbol{D}\boldsymbol{u}(k)) \\ \hat{\boldsymbol{x}}(0) &= \boldsymbol{m}_0 \\ \boldsymbol{r}(k) &= \boldsymbol{y}(k) - \boldsymbol{C}\hat{\boldsymbol{x}}(k) - \boldsymbol{D}\boldsymbol{u}(k), \end{aligned} \tag{6.179}$$

where the filter gain $\boldsymbol{K}(k)$ is given by

$$\boldsymbol{K}(k) = -(\boldsymbol{A}\boldsymbol{P}(k)\boldsymbol{C}' + \boldsymbol{Q}_{wv})(\boldsymbol{Q}_v + \boldsymbol{C}\boldsymbol{P}(k)\boldsymbol{C}')^{-1}, \tag{6.180}$$

$\boldsymbol{P}(k)$ being the solution of the following discrete Riccati equation

$$\begin{aligned} \boldsymbol{P}(k+1) \quad = \quad & \boldsymbol{A}\boldsymbol{P}(k)\boldsymbol{A}' - (\boldsymbol{A}\boldsymbol{P}(k)\boldsymbol{C}' + \boldsymbol{Q}_{wv})(\boldsymbol{Q}_v + \boldsymbol{C}\boldsymbol{P}(k)\boldsymbol{C}')^{-1} \cdot \\ & \cdot (\boldsymbol{C}\boldsymbol{P}(k)\boldsymbol{A}' + \boldsymbol{Q}'_{wv}) + \boldsymbol{Q}_w, \qquad \boldsymbol{P}(0) = \boldsymbol{\Pi}_0 \end{aligned} \tag{6.181}$$

For implementation purpose, it is interesting to separate the equations of the Kalman filter in a two-stage update procedure at each sampling time: a measurement update and a time update. The approach is the object of the following remark. This implementation of the Kalman filter allows one to handle missing measurements in a straightforward way. □

Remark 6.16 *Handling missing measurements*

The measurement update consists in taking into account the new information brought by an additional measurement, say $\boldsymbol{y}(k)$, in order to compute $\hat{\boldsymbol{x}}(k|k)$, the best estimate (in the least square sense) of $\boldsymbol{x}(k)$ given $\boldsymbol{u}(i), \boldsymbol{y}(i), i = 1, 2, \cdots, k$. The latter is deduced from $\boldsymbol{u}(k), \boldsymbol{y}(k)$ and from $\hat{\boldsymbol{x}}(k)$ the best prediction of $\boldsymbol{x}(k)$ given $\boldsymbol{u}(i), \boldsymbol{y}(i), i = 1, 2, \cdots, k-1$. The time update then uses the plant model in order to predict the state evolution one step ahead.

An additional hypothesis is needed to use the algorithm below: the variance of the measurement noise, $\boldsymbol{Q}_v$ should be positive definite and diagonal. Thus, $\boldsymbol{Q}_v = \text{diag}\ (q_{v,1} \cdots q_{v,p})$, where $q_{vi} > 0, i = 1, \cdots, p$. The "diagonality" condition can be enforced by a suitable change of output variable when $\boldsymbol{Q}_v$ is positive definite. It suffices to set $\boldsymbol{y}_d(k) = \boldsymbol{Q}_v^{-1/2}\boldsymbol{y}(k)$, so that the variance of $\boldsymbol{y}_d(k)$ is the $p \times p$ identity matrix.

The following notations are introduced in the algorithm below: $\boldsymbol{c}_i$ and $\boldsymbol{d}_i$ denote respectively the i^{th} row of $\boldsymbol{C}$ and $\boldsymbol{D}$.

Algorithm 6.12 *Measurement and time update for the innovation filter*

Initialization: Set $\boldsymbol{P}(0) = \boldsymbol{\Pi}_0, \hat{\boldsymbol{x}}(0) = \boldsymbol{m}_0$.

At each sampling time:

1. Measurement update.
   Set $\boldsymbol{P}_0(k) = \boldsymbol{P}(k), \hat{\boldsymbol{x}}_0(k|k) = \hat{\boldsymbol{x}}(k)$.
   For $i = 1$ up to p, compute $\boldsymbol{P}_i(k|k)^{-1} = \boldsymbol{P}_{i-1}(k)^{-1} + \boldsymbol{c}_i'\boldsymbol{c}_i/q_{v,i}$.
   Set $\boldsymbol{P}(k|k) = \boldsymbol{P}_p(k|k)$.
   For $i = 1$ up to p, compute

$$\begin{aligned} \boldsymbol{K}_{f,i}(k) &= \boldsymbol{P}(k|k)\boldsymbol{c}_i'/q_{v,i} \\ \hat{\boldsymbol{x}}_i(k|k) &= \hat{\boldsymbol{x}}_{i-1}(k|k) + \boldsymbol{K}_{f,i}(k)(y_i(k) - \boldsymbol{c}_i\hat{\boldsymbol{x}}(k) - \boldsymbol{d}_i\boldsymbol{u}(k)). \end{aligned}$$

   Set $\hat{\boldsymbol{x}}(k|k) = \hat{\boldsymbol{x}}_p(k|k)$.

2. Time update.
   Compute successively

$$\begin{aligned} \boldsymbol{K}_f(k) &= \boldsymbol{P}(k|k)\boldsymbol{C}'\boldsymbol{Q}_v^{-1} \\ \boldsymbol{P}(k+1) &= \boldsymbol{A}\boldsymbol{P}(k|k)\boldsymbol{A}' + \boldsymbol{Q}_w - \boldsymbol{Q}_{wv}(\boldsymbol{Q}_v + \boldsymbol{C}\boldsymbol{P}(k)\boldsymbol{C}')^{-1}\boldsymbol{Q}_{wv}' \\ &\quad -\boldsymbol{A}\boldsymbol{K}_f(k)\boldsymbol{Q}_{wv}' - \boldsymbol{Q}_{wv}\boldsymbol{K}_f(k)'\boldsymbol{A}' \\ \hat{\boldsymbol{x}}(k+1) &= \boldsymbol{A}\hat{\boldsymbol{x}}(k|k) + \boldsymbol{Q}_{wv}(\boldsymbol{Q}_v + \boldsymbol{C}\boldsymbol{P}(k)\boldsymbol{C}')^{-1} \cdot \\ &\quad \cdot(\boldsymbol{y}(k) - \boldsymbol{C}\hat{\boldsymbol{x}}(k) - \boldsymbol{D}\boldsymbol{u}(k)). \end{aligned}$$

3. Computation of the residual.
   For $i = 1$ up to p, compute the components of the residual vector

$$r_i(k) = y_i(k) - \boldsymbol{c}_i\hat{\boldsymbol{x}}(k) - \boldsymbol{d}_i\boldsymbol{u}(k)).$$

Result: Residual vector $\boldsymbol{r}(k)$ for increasing time horizon k.

When a measurement is missing, it suffices to skip the corresponding measurement update, namely to skip the corresponding value of index i in the "for" loops. □

With unknown input. In this case, the design of an innovation filter consists of a two step procedure. First a reduced system having no unknown input is extracted from the original system. Then a steady state Kalman filter is designed for this subsystem and the candidate residual signal is nothing but the innovation associated to this filter. As above, to check whether the innovation sequence is a residual, its sensitivity to the fault vector $\boldsymbol{f}$ has to be verified, which is the object of the next subsection.

The idea behind the extraction of the subsystem will first be sketched in the case, where $\boldsymbol{E}_d = \boldsymbol{O}$ (no unknown input affecting $\boldsymbol{y}$). Next a complete algorithm will be provided to solve Problem 6.14. The justification of this algorithm is relatively

involved, and the interested reader is invited to consult the bibliography for the proofs.

To extract a subsystem which has not $\boldsymbol{d}$ as input, let $\boldsymbol{x}_{sub}$ denote the state of this subsystem and set

$$\boldsymbol{x}_{sub}(k) = \boldsymbol{\Pi x}(k), \tag{6.182}$$

where $\boldsymbol{\Pi}$ is an $n_{sub} \times n$ matrix (with $n_{sub} \leq n$) to be determined. By multiplying the first Eq. (6.172) by $\boldsymbol{\Pi}$ on the left, and by taking (6.182) into account, one gets

$$\begin{aligned} \boldsymbol{x}_{sub}(k+1) &= \boldsymbol{\Pi A x}(k) + \boldsymbol{\Pi B u}(k) + \boldsymbol{\Pi F}_d \boldsymbol{d}(k) \\ &\quad + \boldsymbol{\Pi F}_f \boldsymbol{f}(k) + \boldsymbol{\Pi B}_\epsilon \boldsymbol{\epsilon}(k). \end{aligned} \tag{6.183}$$

If the following relations are imposed

$$\boldsymbol{\Pi A} = \bar{\boldsymbol{A}}\boldsymbol{\Pi} + \bar{\boldsymbol{B}}\boldsymbol{C} \tag{6.184}$$

$$\boldsymbol{\Pi F}_d = \boldsymbol{O}, \tag{6.185}$$

where $\bar{\boldsymbol{A}}$ and $\bar{\boldsymbol{B}}$ are unknown matrices to be determined, then (6.183) can be written as

$$\begin{aligned} \boldsymbol{x}_{sub}(k+1) &= \bar{\boldsymbol{A}}\boldsymbol{x}_{sub}(k) + \bar{\boldsymbol{B}}\left(\boldsymbol{y}(k) - \boldsymbol{D u}(k) - \boldsymbol{E}_f \boldsymbol{f}(k) - \boldsymbol{D}_\epsilon \boldsymbol{\epsilon}(k)\right) \\ &\quad + \boldsymbol{\Pi B u}(k) + \boldsymbol{\Pi F}_f \boldsymbol{f}(k) + \boldsymbol{\Pi B}_\epsilon \boldsymbol{\epsilon}(k) \end{aligned} \tag{6.186}$$

by using (6.182) and the output of Eq. (6.172) (in which $\boldsymbol{E}_d$ is assumed to be null). Introducing the abbreviations

$$\begin{aligned} \tilde{\boldsymbol{B}} &= \boldsymbol{\Pi B} - \bar{\boldsymbol{B}}\boldsymbol{D} \\ \tilde{\boldsymbol{F}}_f &= \boldsymbol{\Pi F}_f - \bar{\boldsymbol{B}}\boldsymbol{E}_f \\ \tilde{\boldsymbol{B}}_\epsilon &= \boldsymbol{\Pi B}_\epsilon - \bar{\boldsymbol{B}}\boldsymbol{D}_\epsilon \end{aligned}$$

into (6.186) yields

$$\boldsymbol{x}_{sub}(k+1) = \bar{\boldsymbol{A}}\boldsymbol{x}_{sub}(k) + \tilde{\boldsymbol{B}}\boldsymbol{u}(k) + \bar{\boldsymbol{B}}\boldsymbol{y}(k) + \tilde{\boldsymbol{F}}_f \boldsymbol{f}(k) + \tilde{\boldsymbol{B}}_\epsilon \boldsymbol{\epsilon}(k). \tag{6.187}$$

This system has no unknown input $\boldsymbol{d}$ as could be expected by imposing (6.185). To design a Kalman filter based on the state Eq. (6.187) when $\boldsymbol{f} = 0$, the part of the measurement $\boldsymbol{y}$ which depends on $\boldsymbol{x}_{sub}$, $\boldsymbol{u}$ and $\boldsymbol{\epsilon}$ only should be determined. This is achieved by defining the signal $\boldsymbol{y}_{sub}$ as

$$\boldsymbol{y}_{sub}(k) = \boldsymbol{M y}(k) = \boldsymbol{MCx}(k) + \boldsymbol{MDu}(k) + \boldsymbol{MD}_\epsilon \boldsymbol{\epsilon}(k), \tag{6.188}$$

where $\boldsymbol{M}$ is unknown. Imposing

$$\boldsymbol{MC} = \boldsymbol{L\Pi}, \tag{6.189}$$

(6.188) becomes

$$\boldsymbol{y}_{sub}(k) = \boldsymbol{L}\boldsymbol{x}_{sub}(k) + \boldsymbol{MDu}(k) + \boldsymbol{MD}_\epsilon \boldsymbol{\epsilon}(k), \tag{6.190}$$

which has the required form.

Now, provided $(\boldsymbol{L},\ \bar{\boldsymbol{A}})$ is detectable, and $(\bar{\boldsymbol{A}},\ \tilde{\boldsymbol{B}}_\epsilon,\ \boldsymbol{L},\ \boldsymbol{MD}_\epsilon)$ is regular, a Kalman filter can be designed for the subsystem (6.187), (6.190), when $\boldsymbol{f} = 0$

$$\begin{aligned}\hat{\boldsymbol{x}}_{sub}(k+1) &= \bar{\boldsymbol{A}}\hat{\boldsymbol{x}}_{sub}(k) + \tilde{\boldsymbol{B}}\boldsymbol{u}(k) + \bar{\boldsymbol{B}}\boldsymbol{y}(k) - \boldsymbol{K}_{sub}\,(\boldsymbol{y}_{sub} \\ &\quad -\boldsymbol{L}\hat{\boldsymbol{x}}_{sub}(k) - \boldsymbol{MD}\boldsymbol{u}(k))\,, \end{aligned} \tag{6.191}$$

where

$$\boldsymbol{K}_{sub} = -(\bar{\boldsymbol{A}}\boldsymbol{P}_{sub}\,\boldsymbol{L}' + \tilde{\boldsymbol{B}}_\epsilon\,\boldsymbol{D}'_\epsilon\,\boldsymbol{M}')(\boldsymbol{MD}_\epsilon\,\boldsymbol{D}'_\epsilon\,\boldsymbol{M}' + \boldsymbol{L}\boldsymbol{P}_{sub}\,\boldsymbol{L}')^{-1}$$

with $\boldsymbol{P}_{sub}$ the symmetric semi-positive definite solution of

$$\begin{aligned}\boldsymbol{P}_{sub} &= \bar{\boldsymbol{A}}\boldsymbol{P}_{sub}\,\bar{\boldsymbol{A}}' - \left(\bar{\boldsymbol{A}}\boldsymbol{P}_{sub}\,\boldsymbol{L}' + \tilde{\boldsymbol{B}}_\epsilon\,\boldsymbol{D}'_\epsilon\,\boldsymbol{M}'\right)\left(\boldsymbol{MD}_\epsilon\,\boldsymbol{D}'_\epsilon\boldsymbol{M}'\right. \\ &\quad \left.+\boldsymbol{L}\boldsymbol{P}_{sub}\,\boldsymbol{L}'\right)^{-1}\left(\boldsymbol{L}\boldsymbol{P}_{sub}\bar{\boldsymbol{A}}' + \boldsymbol{MD}_\epsilon\,\tilde{\boldsymbol{B}}'_\epsilon\right) + \tilde{\boldsymbol{B}}_\epsilon\,\tilde{\boldsymbol{B}}'_\epsilon. \end{aligned}$$

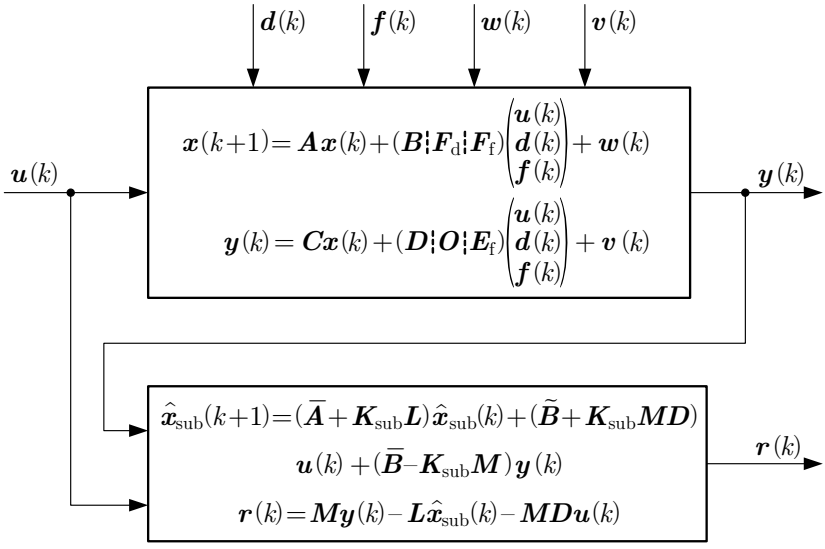


Fig. 6.17. Block diagram of the supervised system together with the innovation filter in the presence of unknown inputs

The associated output reconstruction error is given by

$$\boldsymbol{r}(k) = \boldsymbol{y}_{sub}\,(k) - \boldsymbol{L}\hat{\boldsymbol{x}}_{sub}\,(k) - \boldsymbol{MD}\boldsymbol{u}(k)\,. \tag{6.192}$$

which can be evaluated from the available data. It can be checked that it fulfils the conditions for an innovation sequence. Indeed, the state estimation error, $\tilde{\boldsymbol{x}}_{sub}(k) = \boldsymbol{x}_{sub}(k) - \hat{\boldsymbol{x}}_{sub}(k)$, verifies the following equation obtained by subtracting (6.191) from (6.187):

$$\begin{aligned}\tilde{\boldsymbol{x}}_{sub}(k+1) &= \left(\bar{\boldsymbol{A}} + \boldsymbol{K}_{sub}\,\boldsymbol{L}\right)\,\tilde{\boldsymbol{x}}_{sub}\,(k) + \tilde{\boldsymbol{F}}_f\,\boldsymbol{f}(k) \\ &\quad + \left(\tilde{\boldsymbol{B}}_\epsilon + \boldsymbol{K}_{sub}\,\boldsymbol{M}\boldsymbol{D}_\epsilon\right)\boldsymbol{\epsilon}(k)\,. \end{aligned} \tag{6.193}$$

This error is clearly not affected by $\boldsymbol{d}$ and $\boldsymbol{u}$, and so is the associated innovation $\boldsymbol{r}(k) = \boldsymbol{L}\tilde{\boldsymbol{x}}_{sub}\,(k) + \boldsymbol{M}\boldsymbol{D}_\epsilon\,\boldsymbol{\epsilon}(k)$. Condition 1 of Problem 6.14 is thus fulfilled. To assure that the maximum amount of information on the fault has been kept (condition 2 of Problem 6.14), $\boldsymbol{x}_{sub}$ should have the largest possible dimension ($\boldsymbol{\Pi}$ should have the largest possible number of rows). The implementation of the innovation filter is summarised in the block diagram of Fig. 6.17

The design of an innovation filter essentially amounts to solving the set of nonlinear algebraic Eqs. (6.184), (6.185), (6.189). Despite the nonlinearity, an algorithm based only on linear algebraic operations can be derived. It is presented below in the general situation, where $\boldsymbol{E}_d \neq \boldsymbol{O}$.

Algorithm 6.13 *Innovation filter design for a system with unknown input*

Given: A system of the form (6.172).

Compute:

1. Determine the integer n_{sub} together with full row rank and full column rank matrices $\boldsymbol{\Gamma}$ and $\boldsymbol{\Phi}$ respectively such that

$$\begin{pmatrix} -z\boldsymbol{I}_{n_{sub}} + \boldsymbol{A}_{sub} & \boldsymbol{B}_{sub} \\ \boldsymbol{C}_{sub} & \boldsymbol{D}_{sub} \end{pmatrix} = \boldsymbol{\Gamma} \begin{pmatrix} -z\boldsymbol{I}_n + \boldsymbol{A} & \boldsymbol{F}_d & \boldsymbol{B}_\epsilon \\ \boldsymbol{C} & \boldsymbol{E}_d & \boldsymbol{D}_\epsilon \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{\Phi} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I}_{n_\epsilon} \end{pmatrix}.$$

The Algorithm 6.14 presented below can be used to compute $n_{sub}, \boldsymbol{\Gamma}$ and $\boldsymbol{\Phi}$. The n_{sub}-dimensional subsystem $(\boldsymbol{A}_{sub}, \boldsymbol{B}_{sub}, \boldsymbol{C}_{sub}, \boldsymbol{D}_{sub})$ has no unknown input. Let p_{sub} denote the number of rows of $\boldsymbol{C}_{sub}$.

2. Design a Kalman filter for the following reduced system:

$$\begin{aligned} \boldsymbol{x}_{sub}(k+1) &= \boldsymbol{A}_{sub}\,\boldsymbol{x}_{sub}(k) + \tilde{\boldsymbol{B}}_{sub}\,\boldsymbol{u}(k) - \boldsymbol{\Gamma}_{12}\,\boldsymbol{y}(k) \\ &\quad + \boldsymbol{B}_{sub}\,\boldsymbol{\epsilon}_{sub}(k) \\ \boldsymbol{x}_{sub}(0) &= \boldsymbol{x}_{sub,0} \\ \boldsymbol{y}_{sub}(k) &= \boldsymbol{\Gamma}_{22}\,\boldsymbol{y}(k) = \boldsymbol{C}_{sub}\,\boldsymbol{x}_{sub}(k) + \tilde{\boldsymbol{D}}_{sub}\,\boldsymbol{u}(k) \\ &\quad + \boldsymbol{D}_{sub}\,\boldsymbol{\epsilon}_{sub}(k), \end{aligned}$$

where $\boldsymbol{\epsilon}_{sub}(k)$ is a sample of a white noise sequence with variance equal to the identity matrix,

$$\boldsymbol{\Gamma} = \begin{pmatrix} \boldsymbol{\Gamma}_{11} & \boldsymbol{\Gamma}_{12} \\ \boldsymbol{\Gamma}_{21} & \boldsymbol{\Gamma}_{22} \end{pmatrix}$$

with $\boldsymbol{\Gamma}_{11}$, $\boldsymbol{\Gamma}_{12}$, $\boldsymbol{\Gamma}_{21}$, $\boldsymbol{\Gamma}_{22}$ respectively $n_{sub} \times n$, $n_{sub} \times p$, $p_{sub} \times n$ and $p_{sub} \times p$ -dimensional matrices,

$$\begin{aligned} \tilde{\boldsymbol{B}}_{sub} &= \boldsymbol{\Gamma}_{11}\,\boldsymbol{B} + \boldsymbol{\Gamma}_{12}\,\boldsymbol{D}, \\ \tilde{\boldsymbol{D}}_{sub} &= \boldsymbol{\Gamma}_{21}\,\boldsymbol{B} + \boldsymbol{\Gamma}_{22}\,\boldsymbol{D}. \end{aligned}$$

The resulting innovation qualifies as a residual.

Result: An innovation filter for system (6.172).

Here is the algorithm to be used in step a).

Algorithm 6.14 *Computation of* n_{sub}, $\boldsymbol{\Gamma}$, $\boldsymbol{\Phi}$

Initialisation: Let

$$\boldsymbol{Z} = \begin{pmatrix} -\boldsymbol{I}_n & \boldsymbol{O}_{n\times n_d} \\ \boldsymbol{O}_{p\times n} & \boldsymbol{O}_{p\times n_d} \end{pmatrix}, \boldsymbol{W} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{F}_d \\ \boldsymbol{C} & \boldsymbol{E}_d \end{pmatrix}.$$

Set

$\boldsymbol{Z}^* = \boldsymbol{Z}$, $\boldsymbol{W}^* = \boldsymbol{W}$, $\boldsymbol{M} = \boldsymbol{I}_{n+p}$ and $\boldsymbol{N} = \boldsymbol{I}_{n+n_d}$.

Compute:

a. While $\boldsymbol{Z}^*$ is not full column rank, do

1. perform a singular value decomposition of $\boldsymbol{Z}^*$,

$$\boldsymbol{Z}^* = (\boldsymbol{U}^1_{Z^*}\ \boldsymbol{U}^2_{Z^*}) \begin{pmatrix} \boldsymbol{\Sigma}_{Z^*} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \boldsymbol{V}^1_{Z^*} \\ \boldsymbol{V}^2_{Z^*} \end{pmatrix},$$

and compress the columns of $\boldsymbol{Z}^*$ by computing the right hand side of the first equality below:

$$(\boldsymbol{Z}^*_1\ \boldsymbol{O}) = \boldsymbol{Z}^* \begin{pmatrix} \boldsymbol{V}^{1'}_{Z^*} & \boldsymbol{V}^{2'}_{Z^*} \end{pmatrix} = (\boldsymbol{U}^1_{Z^*}\boldsymbol{\Sigma}_{Z^*}\ \boldsymbol{O}).$$

2. Let $(\boldsymbol{W}^*_1\ \boldsymbol{W}^*_2) = \boldsymbol{W}^* \begin{pmatrix} \boldsymbol{V}^{1'}_{Z^*} & \boldsymbol{V}^{2'}_{Z^*} \end{pmatrix}$.

3. Find the highest rank full row rank matrix $\boldsymbol{L}$ satisfying $\boldsymbol{L}\boldsymbol{W}^*_2 = 0$ as follows. Perform a singular value decomposition of

$$\boldsymbol{W}^*_2\text{: } \boldsymbol{W}^*_2 = \begin{pmatrix} \boldsymbol{U}^1_{W^*_2} & \boldsymbol{U}^2_{W^*_2} \end{pmatrix} \begin{pmatrix} \boldsymbol{\Sigma}_{W^*_2} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \boldsymbol{V}^1_{W^*_2} \\ \boldsymbol{V}^2_{W^*_2} \end{pmatrix}.$$

Noticing that

$$\begin{pmatrix} \boldsymbol{U}^{1'}_{W^*_2} \\ \boldsymbol{U}^{2'}_{W^*_2} \end{pmatrix} \boldsymbol{W}^*_2 = \begin{pmatrix} \boldsymbol{\Sigma}_{W^*_2}\boldsymbol{V}^1_{W^*_2} \\ \boldsymbol{O} \end{pmatrix},$$

one gets $\boldsymbol{L} = \boldsymbol{U}^{2'}_{W^*_2}$.

4. Let $\boldsymbol{Z}^* = \boldsymbol{L}\boldsymbol{Z}^*_1$, $\boldsymbol{W}^* = \boldsymbol{L}\boldsymbol{W}^*_1$, $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{M}$, $\boldsymbol{N} = \boldsymbol{N}\boldsymbol{V}^{1'}_{Z^*}$, end do.

b. Determine an invertible matrix $\boldsymbol{T}$ such that

$$\boldsymbol{T}\boldsymbol{Z}^* = \begin{pmatrix} -\boldsymbol{I} \\ \boldsymbol{O} \end{pmatrix},$$

where the dimension of $\boldsymbol{I}$ is obviously equal to rank$\boldsymbol{Z}^*$. Such a matrix can be computed as follows:

$$\boldsymbol{T} = \begin{pmatrix} (\boldsymbol{\Sigma}_{Z^*}\boldsymbol{V}_{Z^*})^{-1} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{U}_{Z^*}^{1'} \\ \boldsymbol{U}_{Z^*}^{2'} \end{pmatrix},$$

where the notations are the same as for the singular value decomposition of $\boldsymbol{Z}^*$ above, except that $\boldsymbol{V}_{Z^*}^1 = \boldsymbol{V}_{Z^*}$ and $\boldsymbol{V}_{Z^*}^2$ does not exist since $\boldsymbol{Z}^*$ has full column rank now.

c. Set $\boldsymbol{\Gamma} = \boldsymbol{T} - \boldsymbol{M}, \quad \boldsymbol{\Phi} = \boldsymbol{N}$,

$$\begin{pmatrix} \boldsymbol{A}_{sub} \\ \boldsymbol{C}_{sub} \end{pmatrix} = -\boldsymbol{\Gamma}\boldsymbol{W}\boldsymbol{\Phi}, \quad \begin{pmatrix} \boldsymbol{B}_{sub} \\ \boldsymbol{D}_{sub} \end{pmatrix} = \boldsymbol{\Gamma} \begin{pmatrix} \boldsymbol{B}_{\epsilon} \\ \boldsymbol{D}_{\epsilon} \end{pmatrix}.$$

The above design procedure may fail in different ways:

- When the dimensions of $\boldsymbol{\Gamma}$ and $\boldsymbol{\Phi}$ are such that $\begin{pmatrix} \boldsymbol{A}_{sub} \\ \boldsymbol{C}_{sub} \end{pmatrix}$ is a square matrix, the obtained subsystem has no output, and hence no Kalman filter can be designed and no residual generator can be obtained. This typically occurs when $n_d \geq p$.
- When

$$\begin{pmatrix} z\boldsymbol{I} + \boldsymbol{A}_{sub} & \boldsymbol{B}_{sub} \\ \boldsymbol{C}_{sub} & \boldsymbol{D}_{sub} \end{pmatrix}$$

has full generic rank, but it looses rank for $z = \exp(-j\omega)$, $\omega \in \mathbb{R}$, then it is not possible to design a residual generator as the regularity assumption needed for the design of the Kalman filter is not fulfilled. [6]
- When the regularity assumption ceases to be met due to $\boldsymbol{B}_{sub} = \boldsymbol{O}, \boldsymbol{D}_{sub} = \boldsymbol{O}$ or due to

$$\begin{pmatrix} -z\boldsymbol{I} + \boldsymbol{A}_{sub} & \boldsymbol{B}_{sub} \\ \boldsymbol{C}_{sub} & \boldsymbol{D}_{sub} \end{pmatrix}$$

having not full generic rank, the design is more involved and the reader is referred to the bibliography for this case.

Example 6.15 *Innovation filter design for the ship example*

Let us consider the linearised augmented ship-steering model described by combining the wave model and the ship-steering system

[6] Fulfilment of this condition can be checked by computing the zeros of system $(\boldsymbol{A}_{sub}, \boldsymbol{B}_{sub}, \boldsymbol{C}_{sub}, \boldsymbol{D}_{sub})$ and by verifying that none of them lies on the unit circle.

$$\begin{pmatrix} \dot{x}_{w1} \\ \dot{x}_{w2} \\ \dot{\omega}_3 \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} -2\eta_\omega\,\sigma_0 & -\sigma_0^2 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & b\eta_1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_{w1} \\ x_{w2} \\ \omega_3 \\ \psi \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ b \\ 0 \end{pmatrix} \delta + \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} w_\omega$$

$$\begin{pmatrix} \omega_{3m} \\ \psi_m \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{w1} \\ x_{w2} \\ \omega_3 \\ \psi \end{pmatrix} + \begin{pmatrix} f_\omega \\ f_\psi \end{pmatrix} + \begin{pmatrix} \nu_\omega \\ \nu_\psi \end{pmatrix}.$$

A sampled-data model of this system has been obtained at a sampling rate of $0.5Hz$. The resulting equations are:

$$\begin{pmatrix} x_{w1}(k+1) \\ x_{w2}(k+1) \\ \omega_3(k+1) \\ \psi(k+1) \end{pmatrix} = \begin{pmatrix} -0.1281 & -0.6365 & 0 & 0 \\ 0.9945 & 0.1106 & 0 & 0 \\ 0 & 0 & 0.0000 & 0 \\ 0.9945 & -0.8894 & 0.0500 & 1.0000 \end{pmatrix} \begin{pmatrix} x_{w1}(k) \\ x_{w2}(k) \\ \omega_3(k) \\ \psi(k) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0.0500 \\ 0.0975 \end{pmatrix} \delta(k) + \boldsymbol{w}(k) \tag{6.194}$$

$$\begin{pmatrix} \omega_{3m}(k) \\ \psi_m(k) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{w1}(k) \\ x_{w2}(k) \\ \omega_3(k) \\ \psi(k) \end{pmatrix} + \begin{pmatrix} f_\omega(k) \\ f_\psi(k) \end{pmatrix} + \begin{pmatrix} v_\omega(k) \\ v_\psi(k) \end{pmatrix}. \tag{6.195}$$

The covariance matrix of the state noise $\boldsymbol{w}(k)$ can be evaluated by the sampling procedure described in Appendix 2. It yields

$$E(\boldsymbol{w}(k)\boldsymbol{w}(k)') = \boldsymbol{Q}_w = \begin{pmatrix} 0.0015 & 0.0056 & 0.0019 & 0.0056 \\ 0.0056 & 0.0322 & 0.0077 & 0.0322 \\ 0.0019 & 0.0077 & 0.0024 & 0.0077 \\ 0.0056 & 0.0322 & 0.0077 & 0.0322 \end{pmatrix}.$$

The measurement noise sequence is characterised by a covariance matrix given as

$$\boldsymbol{Q}_v = \begin{pmatrix} 0.0001 & 0 \\ 0 & 0.005 \end{pmatrix}.$$

State and measurement noise are supposed to be uncorrelated, hence $\boldsymbol{Q}_{wv} = \boldsymbol{O}$.

The considered input signal $\delta(t)$ is a sine wave with period 20π seconds.

Figure 6.18 gives the evolution of the sampled output signals in healthy working mode (first 300 samples), when a 0.1 deg/s bias on the turn rate $\omega_3(k)$ is added (from sample 301 to sample 600), and when this bias disappears bringing the system back to healthy working mode (sample 601 to 900) In other words, a step-like fault f_ω occurs between sample 301 and 600.

From the above model, the following Kalman filter is deduced [7].

[7] The gain of this filter can be computed by MATLAB function *dlqe* for instance.

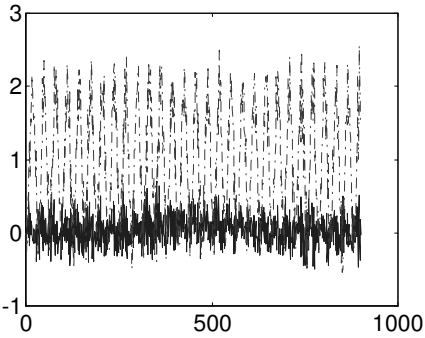


Fig. 6.18. Sampled output sequence of ship model in healthy and faulty working modes; ω_{3m} as a function of sample number (continuous line), ψ_m as a function of sample number (dash-dotted line)

$$\begin{pmatrix} \hat{x}_{w1}(k+1) \\ \hat{x}_{w2}(k+1) \\ \hat{\omega}_3(k+1) \\ \hat{\psi}(k+1) \end{pmatrix} = \begin{pmatrix} -0.1281 & -0.6365 & 0 & 0 \\ 0.9945 & 0.1106 & 0 & 0 \\ 0 & 0 & 0.0000 & 0 \\ 0.9945 & -0.8894 & 0.0500 & 1.0000 \end{pmatrix} \begin{pmatrix} \hat{x}_{w1}(k) \\ \hat{x}_{w2}(k) \\ \hat{\omega}_3(k) \\ \hat{\psi}(k) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0.0500 \\ 0.0975 \end{pmatrix} \delta(k) + \begin{pmatrix} -0.3265 & -0.4544 \\ 0.6710 & 0.0067 \\ 0.0000 & 0.0000 \\ 0.6880 & 0.0119 \end{pmatrix} \cdot \left(\begin{pmatrix} \omega_{3m}(k) \\ \psi_m(k) \end{pmatrix} - \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{x}_{w1}(k) \\ \hat{x}_{w2}(k) \\ \hat{\omega}_3(k) \\ \hat{\psi}(k) \end{pmatrix} \right). \tag{6.196}$$

The innovation is computed from

$$r(k) = \begin{pmatrix} \omega_{3m}(k) \\ \psi_m(k) \end{pmatrix} - \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{x}_{w1}(k) \\ \hat{x}_{w2}(k) \\ \hat{\omega}_3(k) \\ \hat{\psi}(k) \end{pmatrix}. \tag{6.197}$$

The innovation sequences for the data of Fig. 6.18 is plotted in Fig. 6.19. The change in the mean of the innovation sequence due to the fault is visible. However, such a change cannot be detected by comparing the signals to a simple threshold. □

The existence of a filter that meets the conditions in Problem 6.14 does not guarantee that the filter output (namely the innovation) is useful for fault detection. It should be affected by $\boldsymbol{f}$ in order to meet the second condition of Problem 6.13, and thus to be suitable as a residual. This issue is addressed in the next subsection.

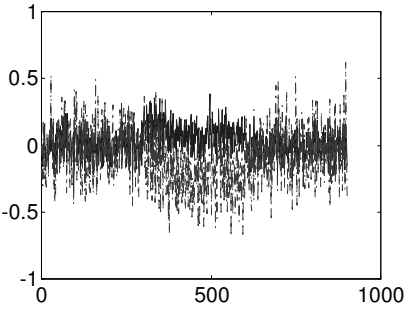


Fig. 6.19. Innovation sequences computed by (6.196), (6.197) from the data of Fig. 6.18; first component (continuous line); second component (dash-dotted line)

Sensitivity to faults or fault detectability. There are several ways to define the sensitivity of an innovation signal to a fault $\boldsymbol{f}$ or, equivalently, the detectability of a fault by a given innovation signal.

In a similar way as for the deterministic case, an innovation filter for system (6.172) is said to be *fault sensitive* if its output is affected by $\boldsymbol{f}$. Equivalently, the fault is said to be detectable in this case.

If the transfer function from $\boldsymbol{f}$ to $\boldsymbol{r}$ is left invertible, then the innovation filter is *strictly fault sensitive*.

It can be shown that system (6.172) has a (strict) fault sensitive innovation filter which solves Problem 6.14 if and only if every innovation filter solution of Problem 6.14 is (strictly) fault sensitive. Thus (strict) fault sensitivity is a property of the supervised system (6.172), and it does not depend upon the choice of innovation filter. Therefore, (strict) fault sensitivity of (6.172) will be referred to in the sequel.

Assuming the pair $(\boldsymbol{C}_{sub}, \boldsymbol{A}_{sub})$ resulting from the design procedure is observable, the following necessary and sufficient conditions for sensitivity can be exploited.

System (6.172) is fault sensitive if and only if [8]:

$$\Im \begin{pmatrix} \boldsymbol{F}_f \\ \boldsymbol{E}_f \end{pmatrix} \not\subset Ker(\boldsymbol{\Gamma}) \,. \tag{6.198}$$

System (6.172) is strictly fault sensitive if and only if system $(\boldsymbol{A}_{sub}, \boldsymbol{F}_{f,sub}, \boldsymbol{C}_{sub}, \boldsymbol{E}_{f,sub})$, where

$$\begin{pmatrix} \boldsymbol{F}_{f,sub} \\ \boldsymbol{E}_{f,sub} \end{pmatrix} = \boldsymbol{\Gamma} \begin{pmatrix} \boldsymbol{F}_f \\ \boldsymbol{E}_f \end{pmatrix} \tag{6.199}$$

is left invertible.

[8] The image (space) $\Im(\boldsymbol{X})$ of a linear transformation associated to the $n \times m$ matrix $\boldsymbol{X}$ is the set of all vectors $\boldsymbol{y}$ in $\mathbb{R}^n$ that equal $\boldsymbol{X}\boldsymbol{u}$ for some $\boldsymbol{u}$ in $\mathbb{R}^m$. The kernel (or null space) $Ker(\boldsymbol{X})$ of a linear transformation associated to the $n \times m$ matrix $\boldsymbol{X}$ is the set of all vectors $\boldsymbol{u}$ in $\mathbb{R}^m$ that fulfil $\boldsymbol{X}\boldsymbol{u} = 0$

Yet another notion is strong fault sensitivity, which is typically considered for scalar faults. As for a deterministic residual, the innovation signal is strongly fault sensitive when it reaches a non-zero steady state value for a step-like fault, $f(k) = \bar{f}1_{\{k>k_0\}}$, for any constant non-zero $\bar{f}$. This property can be checked a posteriori by computing the steady state gain of the transfer function between fault f and innovation $\boldsymbol{r}$ and verifying that it has at least one non-zero entry.

Remark 6.17 *Comments on strong fault detectabilty*
In a deterministic framework, necessary and sufficient conditions for the existence of a residual generator which is strongly fault sensitive for a given system have been developed [187]. The corresponding fault is said to be strongly detectable. It is unclear whether the innovation signal computed as the output of the filter (6.175) (or as the innovation of a Kalman filter for the subsystem in step 2 of the algorithm 6.13) is necessarily strongly fault sensitive, when a strongly detectable fault is considered. □

Distribution of the residual vector and residual evaluation. For proper choice of the residual evaluation method, it is necessary to analyse the statistical distribution of $\boldsymbol{r}(k)$. For the sake of simplicity, the situation, where $\boldsymbol{x}_0$, $\boldsymbol{v}(k)$, $\boldsymbol{w}(k)$, $k = 0, 1, \ldots$, are normally distributed is considered. Then, the residual has asymptotically (as k tends to infinity) a Gaussian distribution with known variance and with zero mean or non-zero mean, depending on whether $\boldsymbol{f}(k)$ asymptotically vanishes or not assuming the fault is strongly detectable. The normal distribution results from the linearity of the filter and the supervised process.

In order to characterise this distribution, let us consider the situation, where there is no unknown input, and hence the residual generator is given by (6.175). The reasoning below also applies to the Kalman filter designed for the system given in step 2 of the algorithm 6.13, but the notations are more cumbersome. The first two moments of the distribution of $\boldsymbol{r}(k)$ can be computed as follows. Let $\tilde{\boldsymbol{x}}(k) = \boldsymbol{x}(k) - \hat{\boldsymbol{x}}(k)$. Then classical results on steady state Kalman filters provide the following expression for the mean and the variance of $\tilde{\boldsymbol{x}}(k)$ in the absence of fault:

$$\begin{aligned} \lim_{k\to\infty} E(\tilde{\boldsymbol{x}}(k)) &= 0 \\ \lim_{k\to\infty} E\left(\tilde{\boldsymbol{x}}(k)\,\tilde{\boldsymbol{x}}(k)'\right) &= \boldsymbol{P}, \end{aligned}$$

with $\boldsymbol{P}$ given as the semi-positive definite solution of (6.177). By substituting the second equation of (6.172) (with $\boldsymbol{E}_d = \boldsymbol{O}$) for $\boldsymbol{y}(k)$ in the expression of $\boldsymbol{r}(k)$ (6.175), the residual can be written as

$$\boldsymbol{r}(k) = \boldsymbol{C}\tilde{\boldsymbol{x}}(k) + \boldsymbol{E}_f\,\boldsymbol{f}(k) + \boldsymbol{D}_\epsilon\,\boldsymbol{\epsilon}(k). \tag{6.200}$$

When $\boldsymbol{f}(k)$ vanishes as k tends to infinity, one deduces from (6.200) with $\boldsymbol{f}(k) = 0$:

$$\begin{aligned} \boldsymbol{r}_m &= \lim_{k\to\infty} E\left(\boldsymbol{r}(k)\right) = 0 \\ \boldsymbol{Q}_r &= \lim_{k\to\infty} E\left(\left(\boldsymbol{r}(k) - \boldsymbol{r}_m\right)\left(\boldsymbol{r}(k) - \boldsymbol{r}_m\right)'\right) = \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}' + \boldsymbol{D}_\epsilon\,\boldsymbol{D}_\epsilon'. \end{aligned}$$

If, on the contrary $\lim_{k\to\infty} \boldsymbol{f}(k) = \bar{\boldsymbol{f}} \neq 0$, the residual mean is non-zero. It can be obtained from the transfer function between $\boldsymbol{f}(z)$ and $\boldsymbol{r}(z)$ deduced from (6.172) and (6.178), namely $\boldsymbol{V}_{ry}(z)\left(\boldsymbol{C}(z\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{F}_f + \boldsymbol{E}_f\right)$. Indeed, the mean of the residual is nothing but the steady state value of the residual for $\boldsymbol{f}(k) = \bar{\boldsymbol{f}}$. Thus,

$$\boldsymbol{r}_m = \lim_{k\to\infty} E(\boldsymbol{r}(k)) = \boldsymbol{V}_{ry}(1)\left(\boldsymbol{C}(\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{F}_f + \boldsymbol{E}_f\right)\bar{\boldsymbol{f}}.$$

Stability of the supervised system is implicitly assumed when writing this expression. The variance of the residual is unchanged, since the fault signal is considered as deterministic.

The problem of fault detection thus amounts to deciding between the following two hypotheses

$$\mathcal{H}_0: \quad \mathcal{L}(\boldsymbol{r}(k)) = As\mathcal{N}(0,\, \boldsymbol{Q}_r) \tag{6.201}$$

$$\mathcal{H}_1: \quad \mathcal{L}(\boldsymbol{r}(k)) = As\mathcal{N}\left(\boldsymbol{V}_{ry}(1)\left(\boldsymbol{C}(\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{F}_f + \boldsymbol{E}_f\right)\bar{\boldsymbol{f}},\, \boldsymbol{Q}_r\right), \tag{6.202}$$

where the notation $\mathcal{L}(\boldsymbol{r}(k))$ denotes the distribution of $\boldsymbol{r}(k)$, and

$$\mathcal{L}(\boldsymbol{r}(k)) = As\mathcal{N}(\boldsymbol{a},\, \boldsymbol{X})$$

indicates that this distribution is normal with mean $\boldsymbol{a}$ and variance $\boldsymbol{X}$ as k tends to infinity. Notice that the residual must be strongly sensitive to fault $\boldsymbol{f}$ for the distributions under $\mathcal{H}_0$ and $\mathcal{H}_1$ to be different.

The asymptotic character of (6.201), (6.202) is due to the effects of initial conditions and filter transients upon occurrence of a fault. Neglecting such transients, and assuming that $\bar{\boldsymbol{f}}$ is known, one can recast the above problem as the following test of hypotheses.

Problem 6.15 (Test of hypotheses: transient not accounted for)
Given a sequence of residual vectors $\boldsymbol{r}(1), \ldots, \boldsymbol{r}(k)$, obtained as the output of filter (6.175), choose between the following two hypotheses at the current time instant k:

$\mathcal{H}_0$*:* $\mathcal{L}(\boldsymbol{r}(i)) = \mathcal{N}(0,\, \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}' + \boldsymbol{D}_\epsilon\boldsymbol{D}_\epsilon')$ *for* $1 \leq i \leq k$,

$\mathcal{H}_1$*: From time instant 1 up to an unknown time instant* k_0,$\boldsymbol{r}(i)$, $i = 1, \ldots, k_0-1$ *is distributed as*

$$\mathcal{L}(\boldsymbol{r}(i)) = \mathcal{N}(0,\, \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}' + \boldsymbol{D}_\epsilon\boldsymbol{D}_\epsilon')$$

while for time instant $i \geq k_0$

$$\mathcal{L}(\boldsymbol{r}(i)) = \mathcal{N}\left((\boldsymbol{V}_{ry}(1)\,\boldsymbol{C}(\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{F}_f + \boldsymbol{E}_f)\bar{\boldsymbol{f}},\, \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}' + \boldsymbol{D}_\epsilon\boldsymbol{D}_\epsilon'\right).$$

This problem is of the form of a change detection in the mean of a Gaussian vector sequence (Problem 6.9) with $\boldsymbol{z}(i)$ replaced by $\boldsymbol{r}(i)$, $\boldsymbol{\mu}_0 = 0$, $\boldsymbol{Q} = \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}' + \boldsymbol{D}_\epsilon\boldsymbol{D}_\epsilon'$ and $\boldsymbol{\mu}_1 = \boldsymbol{\beta} = \left(\boldsymbol{V}_{ry}(1)\,\boldsymbol{C}(\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{F}_f + \boldsymbol{E}_f\right)\bar{\boldsymbol{f}}$. Hence, the CUSUM algorithm based on a step-like change can be used for residual evaluation, with $\bar{f}$ taken as twice the minimum magnitude of the fault to be detected or as the most likely magnitude of this fault. The complete fault detection system is depicted in Fig. 6.20.

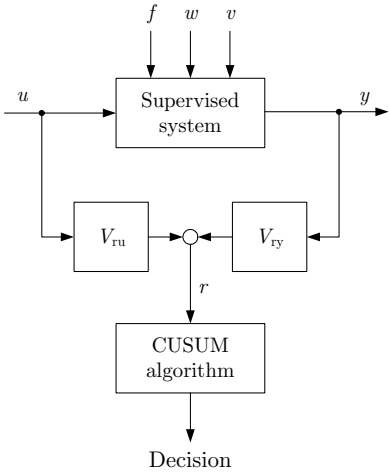


Fig. 6.20. Fault detection system

Remark 6.18 *χ^2-test*
In some applications, particularly in the area of predictive maintenance, the delay for detection may not be a crucial factor, and one may resort to an off-line approach to solve a simplified version of the above hypotheses testing problem. The most recent data over a sliding window $[k-M+1,\ k]$ are considered, and the time instant k_0 is set to 1, which amounts to considering that the change has affected all elements of the batch of data. The method to solve this hypotheses testing problem relies on the χ^2-test which is presented in the appendix for a scalar data set. □

When stating the above hypotheses testing problem, the transient of the system and the residual generator upon occurrence of a fault are not taken into account. This may significantly affect the detection delay. If a priori knowledge on the fault sequence $\boldsymbol{f}(i),\ i = k_0,\ k_0+1,\ \ldots$ is available, the performance of the detection system can be improved by introducing a suitable dynamical profile of change in the CUSUM algorithm.

Most commonly, step-like changes in the fault sequence are considered, namely

$$\begin{array}{lllll} \boldsymbol{f}(i) & = & 0 & i & = & 1,\ 2,\ \ldots,\ k_0-1 \\ \boldsymbol{f}(i) & = & \bar{\boldsymbol{f}} & i & \geq & k_0 \end{array} \tag{6.203}$$

or, in a compact way, $\boldsymbol{f}(i) = \bar{\boldsymbol{f}}_{\{i \geq k_0\}}$, where $\bar{\boldsymbol{f}}$ is a constant vector.

Due to the linearity of the system (6.172) and the filter (6.175), the residual sequence can be written as

$$\boldsymbol{r}(k) = \boldsymbol{r}_0(k) + \boldsymbol{\rho}\,(k,\ k_0), \tag{6.204}$$

where $\boldsymbol{r}_0(k)$ is the value of the residual in the absence of fault, and $\boldsymbol{\rho}\,(k,\ k_0)$ is the contribution to $\boldsymbol{r}(k)$ of a fault occurring at time $k_0 \leq k$. In the case of a step-like fault considered above, $\boldsymbol{\rho}(k,\ k_0)$ can be computed easily; it only depends on the difference $k - k_0$, and hence, is written with an abuse of notation $\boldsymbol{\rho}(k,\ k_0) =$

$\boldsymbol{\rho}(k-k_0)$. For the sake of simplicity, only a scalar fault sequence is considered. Then, $\boldsymbol{\rho}\,(k-k_0) = \tilde{\boldsymbol{\rho}}\,(k-k_0)\bar{f}$, where $\tilde{\boldsymbol{\rho}}\,(k)$ is the response of system (6.172), (6.175) to a fault signal of the form (6.203) with $\bar{f}=1$, for $\boldsymbol{u}(k)=0,\ \boldsymbol{d}(k)=0$ and $\boldsymbol{\epsilon}(k)=0$ for all $k>0$, and for zero initial conditions. It coincides with the step response of the system with transfer function $\boldsymbol{V}_{ry}(z)\ \left(\boldsymbol{C}(z\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{E}_f+\boldsymbol{F}_f\right)$. The hypotheses testing problem when taking into account the dynamical profile of the change can be written as

Problem 6.16 (Test of hypotheses: transient accounted for)
Given a sequence of residual vectors $\boldsymbol{r}(1),\ \ldots,\ \boldsymbol{r}(k)$, obtained as the output of filter (6.175), choose between the following two hypotheses at the current time instant k

$\mathcal{H}_0$*:* $\mathcal{L}(\boldsymbol{r}(i)) = \mathcal{N}(0,\, \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}' + \boldsymbol{D}_\epsilon\,\boldsymbol{D}_\epsilon')$ *for* $1 \le i \le k$,
$\mathcal{H}_1$*: From time instant 1 up to an unknown time instant* k_0, $\boldsymbol{r}(i)$, $i=1,\ \ldots,\ k_0-1$ *is distributed as*

$$\mathcal{L}\,(\boldsymbol{r}(i)) = \mathcal{N}(0,\, \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}' + \boldsymbol{D}_\epsilon\,\boldsymbol{D}_\epsilon') \tag{6.205}$$

while for time instant $i \ge k_0$,

$$\mathcal{L}\,(\boldsymbol{r}(i)) = \mathcal{N}(\tilde{\boldsymbol{\rho}}\,(i-k_0)\bar{f},\, \boldsymbol{C}\boldsymbol{P}\boldsymbol{C}' + \boldsymbol{D}_\epsilon\,\boldsymbol{D}_\epsilon'). \tag{6.206}$$

This problem is in the form of Problem 6.10. (6.205), (6.206) precisely have the form (6.154), (6.155) with $\boldsymbol{r}(i)$ replacing $\boldsymbol{z}(i)$, $\boldsymbol{C}\boldsymbol{P}\boldsymbol{C}'+\boldsymbol{D}_\epsilon\,\boldsymbol{D}_\epsilon'$ replacing $\boldsymbol{Q}$, $\tilde{\boldsymbol{\rho}}\,(i-k_0)\bar{f}$ replacing $\boldsymbol{\rho}(i-k_0)$ and $\boldsymbol{\mu}_0=0$. The CUSUM algorithm based on a known dynamical profile of change can thus be applied with $\boldsymbol{\rho}(k) = \tilde{\boldsymbol{\rho}}\,(k)\bar{f}$, where $\bar{f}$ is taken as twice the minimum magnitude of the change to be detected or as the most likely magnitude of this change.

Remark 6.19 *Delay in dynamic profile*
In the statement of problem 6.10, $\rho(j)$ is supposed to be different from zero for $j>0$. This hypothesis is not verified when the transfer function $\boldsymbol{V}_{ry}(z)\left(\boldsymbol{C}(z\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{E}_f+\boldsymbol{F}_f\right)$ has no direct feedthrough term. In this case, one should use $\rho(k)=\tilde{\rho}(k-\tau)\bar{f}$, where τ denotes the minimum delay in the n_r elements of the mentioned transfer function. □

Notice that strong fault sensitivity is no more a required property of the residual in order to achieve fault detection, when the dynamical profile of the change is accounted for. Indeed, it suffices that the distributions (6.205), (6.206) be different for some time interval. Checking that the fault subsists by reinitialisation of the CUSUM algorithm is however impossible when the residual is not strongly fault sensitive.

Remark 6.20 *Fault sequence*
The choice of a step-like fault sequence can be made without loss of generality. Indeed, other signal forms could possibly be represented as the step response of a linear system, and this linear model could be included in the state-space Eqs. (6.171). □

Example 6.15 (cont.) *Ship example*

The CUSUM algorithm based on the knowledge of the dynamical profile of change will be used to detect the occurrence of fault f_ω. In order to determine the dynamical profile of the change to be used in the algorithm, it suffices to consider the response of the system made of Eqs. (6.194), (6.195), (6.196), (6.197) to a step-like fault f_ω, keeping all other input signals equal to zero and starting with zero initial conditions. This corresponds to the step response of the transfer function $\boldsymbol{V}_{ry}(z) = \left(\boldsymbol{C}(z\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{E}_f + \boldsymbol{F}_f\right)$ with respect to the first input.

Given the specifications, one decides that the smallest bias on ω_3 to be detected is 0.025 deg/s. $\bar{f}_{\omega_3}$ is set to twice this value, which yields 0.05 deg/s. A step of magnitude 0.05 deg/s is thus applied as signal for f_ω. The vector dynamical profile of change with respect to fault f_ω , $0.05\,\tilde{\boldsymbol{\rho}}_{\omega_3}$ is plotted in Fig. 6.21.

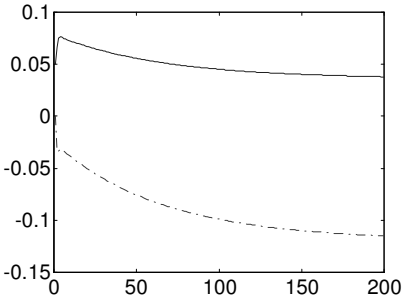


Fig. 6.21. Dynamic profile of change for fault f_ω; first component of $0.05\,\tilde{\rho}_{\omega_3}$ (continuous line); second component (dash-dotted line)

The evolution of the CUSUM decision functions for detection of f_ω, g_{ω_3} is plotted in Fig. 6.22. The indicated threshold (dashed line) has been set on the basis of the value of the decision function for the first 300 samples (healthy working mode). The reinitialisation policy is the reset procedure indicated in the description of the algorithm. One notices the repeated threshold crossing of the decision function g_{ω_3} while the fault is present (from sample 300 to 600). □

6.8.3 Fault estimation

In this section, a model of the form (6.171) in which $n_f = 1$ is considered. Besides, it is assumed that step-like faults of unknown magnitude occur. Thus a scalar sequence $f(i)$, $i = 1, 2, \ldots$ of the form (6.203) with an unknown constant $\bar{f}$ is assumed.

The problem can be stated as:

Problem 6.17 (Fault estimation)
Given

1. *a model of the process of the form (6.171) subject to a scalar step-like fault sequence* $f(i) = \bar{f}1_{\{i \geq k_0\}}$ *of unknown magnitude* $\bar{f}$.

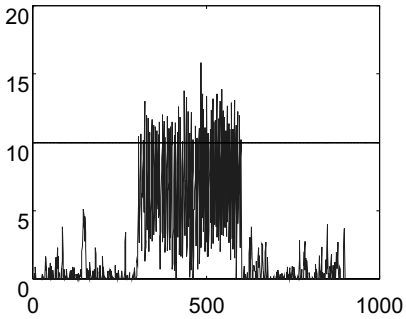


Fig. 6.22. CUSUM decision function resulting from application to the innovation sequence of Fig. 6.19 of the CUSUM algorithm based on the known dynamical profile of change (Fig. 6.21)

2. *a sequence of measured process input and output values:*

$$(\boldsymbol{y}(i),\, \boldsymbol{u}(i))_{1\leq i\leq k}\,,$$

where k denotes the current time instant.

Choose between the following two hypotheses

$\mathcal{H}_0$*: healthy operating condition,*
$\mathcal{H}_1$*: faulty operating condition.*

When $\mathcal{H}_1$ is selected, an estimate of the fault occurrence time, $\hat{k}_0$, and of the fault magnitude, $\hat{\bar{f}}$, should be provided.

As for the fault detection problem, a two step procedure is used to solve this problem. The first step, namely the residual generation, is the same for both problems. For residual evaluation, a generalised likelihood ratio algorithm is used to obtain an estimate of the fault magnitude. Indeed, given the specific fault model, the residual evaluation reduces to Problem 6.16 in which $\bar{f}$ is unknown. Hence, it is of the form of Problem 6.11. (6.205), (6.206) precisely have the form (6.161), (6.162) with $\boldsymbol{r}(i)$ replacing $\boldsymbol{z}(i)$, $\boldsymbol{C}\boldsymbol{P}\boldsymbol{C}' + \boldsymbol{D}_\epsilon\,\boldsymbol{D}_\epsilon'$ replacing $\boldsymbol{Q}$, $\bar{f}$ replacing ν and $\boldsymbol{\mu}_0 = 0$. The GLR algorithm based on a known dynamical profile of change but an unknown fault magnitude can thus directly be used to process the residual vector in order to obtain an on-line solution to Problem 6.17.

Example 6.15 (cont.) *Ship example*

Let us again consider the innovation sequence depicted in Fig. 6.19. Instead of using a CUSUM algorithm, we now perform a GLR algorithm on this sequence. A dynamical profile of change has to be provided. It can be computed as for the CUSUM algorithm and one gets a profile similar to Fig. 6.21 except that the minimum fault magnitude is not accounted for. Thus to obtain the dynamical profile $\tilde{\rho}_{\omega_3}$, the signal f_ω which is used is a step function with unit magnitude instead of the magnitude of 0.05 deg/s used previously.

M is chosen as 50. This allows one to determine a quite precise estimate of the fault magnitude in the reinitialisation procedure. Given the values of the decision function obtained

for the first 300 data, which correspond to the set $\{z_0(1), \cdots, z_0(N_0)\}$, and given its values upon occurrence of the fault, the threshold h is set to 30. The evolution of the GLR decision function is plotted in Fig. 6.23. Each time the threshold is crossed, an alarm is generated, and the decision function remains equal to zero until enough data are available for estimating the fault magnitude in a reliable way. The recursive algorithm restarts at $\hat{k}_0 + M$.

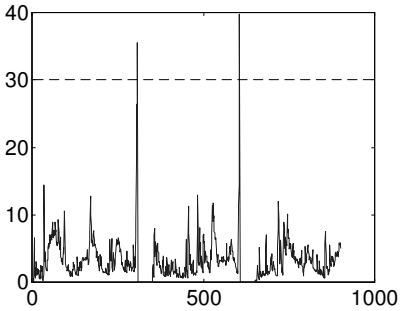


Fig. 6.23. GLR decision function resulting from application to the innovation sequence of Fig. 6.19 of the algorithm with known dynamical profile of change

Note that successive changes separated by less than M samples cannot be handled properly. For the considered data, an alarm is generated at time instants 308 and 606. The estimated change times are 300 and 601 while the actual changes occur at 301 and 601. All numbers should be multiplied by the sampling period to obtain time in seconds. The estimates of the change magnitude used for reinitialisation are respectively 0.1020 for the positive change and -0.1073 for the negative change (disappearance of the fault). Remember that the actual change magnitude is 0.1 in both cases. Notice that the estimate of the change magnitude plotted in Fig. 6.24 converges relatively fast after occurrence of a fault. Hence the horizon M could possibly be chosen smaller for this situation, yet this value is used to make the convergence of the estimate visible in the plot.

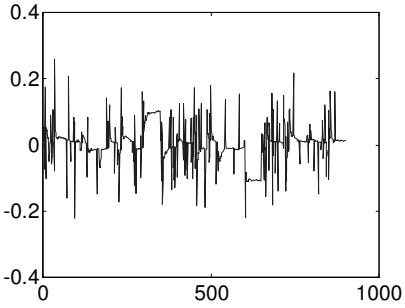


Fig. 6.24. Estimate of the change magnitude resulting from application to the innovation sequence of Fig. 6.19 of the GLR algorithm with known dynamical profile of change

6.8.4 Fault isolation

Up to now the plant model used in the stochastic framework only contained one single (possibly vector) fault to be detected. However, most often several faults may affect the behaviour of the supervised process, and one should not only detect them, but also isolate the faulty components. An appropriate model to describe the process then takes the form

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) + \sum_{i=1}^{n_f} \boldsymbol{F}_i f_i(k) + \boldsymbol{B}_\epsilon\, \boldsymbol{\epsilon}(k) \\ \boldsymbol{x}(0) &= \boldsymbol{x}_0 \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) + \sum_{i=1}^{n_f} \boldsymbol{E}_i f_i(k) + \boldsymbol{D}_\epsilon\, \boldsymbol{\epsilon}(k), \end{aligned} \tag{6.207}$$

where, for the sake of simplicity, scalar faults f_i, $i = 1, \ldots, n_f$ are considered. One way to detect and isolate a single fault, say f_α, is to design a residual which is only sensitive to that fault and to evaluate it in an appropriate way. This can be achieved by recasting the problem as a fault detection problem for a specific system with unknown input. Obviously, the unknown input vector must be made of the faults not to be detected, and thus the model takes the form (6.171) with $\boldsymbol{d} = \left[f_1, \ldots, f_{\alpha-1}, f_{\alpha+1}, \ldots, f_{n_f}\right]'$ and $f = f_\alpha$. One can now proceed as in the section on fault detection (and estimation) to build a system that detects, isolates and possibly estimates fault f_α.

If each fault must be detected and isolated, one solution is to solve n_f fault detection (and estimation) problems of the form just mentioned. This yields a bank of residual generators, each one being affected by a single fault. The table below represents the situation when $n_f = 3$.

Table 6.4 Effects of the faults on the residuals

	$\boldsymbol{f}_1$	$\boldsymbol{f}_2$	$\boldsymbol{f}_3$
$\boldsymbol{r}_1$	$\times$	0	0
$\boldsymbol{r}_2$	0	$\times$	0
$\boldsymbol{r}_3$	0	0	$\times$

where a $\times$ indicates that the fault in the corresponding column affects the residual of the corresponding row. Each residual can be processed individually by a GLR algorithm or a CUSUM algorithm, according as an estimate of the fault magnitude is needed or not.

From the conditions for fault detectability, the following necessary conditions can be deduced for the above scheme to work

$$\begin{aligned} \text{rank}\,\left(\left(\boldsymbol{H}_{yf_\ell}(s)\ \ \boldsymbol{H}_{yf_j}(s)\right)\right) > \ \text{rank}\ \boldsymbol{H}_{yf_j}(s) \\ \text{for all}\ \ \ell, j = 1, \ldots, n_f,\ \ell \neq j, \end{aligned} \tag{6.208}$$

where $\boldsymbol{H}_{yf_\ell}(z)$ is the transfer matrix between f_ℓ and y[9]. A necessary condition for (6.208) to hold is $n_f \leq p$, where p is the number of measured output signals (dimension of $\boldsymbol{y}$).

When it is not possible to design residual generators in such a way that each residual is only sensitive to a single fault, it is still possible to achieve fault isolation provided the zero entries in the table characterising the effect of the faults on the residual have a different pattern in each column. However, only a diagonal structure such as in Table 6.4 allows isolation of multiple simultaneous faults.

Remark 6.21 *Accounting for correlation between residual vectors*
Since the different residual vectors are built on the basis of the same stochastic model, they are generally correlated. Hence the residual evaluation should ideally be carried out on the stacked residual vector

$$\boldsymbol{r}(k) = \begin{pmatrix} \boldsymbol{r}_1(k)' & \boldsymbol{r}_2(k)' & \dots & \boldsymbol{r}_{n_f}(k)' \end{pmatrix}'.$$

The problem can then be written in the form of a multiple hypotheses testing. The interested reader is referred to [179], [114] for the algorithms to be used. □

Example 6.15 (cont.) *Ship example*

Faults on both the rate sensor and the angular position sensor are now considered. Figure 6.25 depicts the measurement signals obtained when step-like faults with magnitude 0.1 deg/s and 0.5 deg are respectively introduced on ω_{3m} between time instant 301 and 600 and on ψ_m between time instant 900 and 1200. All time data are expressed in number of sampling periods.

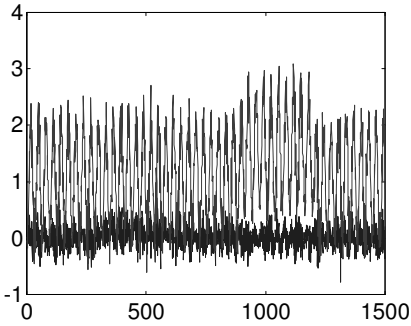


Fig. 6.25. Angular rate and heading measurements

In order to achieve fault isolation, two residual signals are generated, one being sensitive to f_ω, the other to f_ψ. To this end, consider the sampled-data ship model (6.194), (6.195). If a Kalman filter is designed for this system using only the first measurement equation in (6.195), the resulting residual will only be affected by f_ω. Such a filter cannot be designed because the resulting system is not detectable. However, there is no need to estimate the whole state

[9] rank $\boldsymbol{H}(z)$ stands for the normal rank of matrix $\boldsymbol{H}(z)$; it can be computed as $\max_z$ rank $\boldsymbol{H}(z)$

to generate a residual; it suffices to design a Kalman filter for the first 3 state equations in (6.194) and the first measurement equation. This filter takes the form:

$$\begin{pmatrix} \hat{x}_{w1}(k+1) \\ \hat{x}_{w2}(k+1) \\ \hat{\omega}_3(k+1) \end{pmatrix} = \begin{pmatrix} -0.1281 & -0.6365 & 0 \\ 0.9945 & 0.1106 & 0 \\ 0 & 0 & 0.0000 \end{pmatrix} \begin{pmatrix} \hat{x}_{w1}(k) \\ \hat{x}_{w2}(k) \\ \hat{\omega}_3(k) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0.0500 \end{pmatrix} \delta(k) + \begin{pmatrix} -0.6760 \\ 0.8104 \\ 0.0000 \end{pmatrix} \cdot \left(\omega_{3m}(k) - (1 \;\; 0 \;\; 1) \begin{pmatrix} \hat{x}_{w1}(k) \\ \hat{x}_{w2}(k) \\ \hat{\omega}_3(k) \end{pmatrix} \right) \tag{6.209}$$

The innovation is computed from

$$r_{\omega_3}(k) = \omega_{3m}(k) - (1 \;\; 0 \;\; 1) \begin{pmatrix} \hat{x}_{w1}(k) \\ \hat{x}_{w2}(k) \\ \hat{\omega}_3(k) \end{pmatrix} \tag{6.210}$$

It is plotted in Fig. 6.26 (a). A significant change in the mean of this signal is visible when the fault on ω_3 is present.

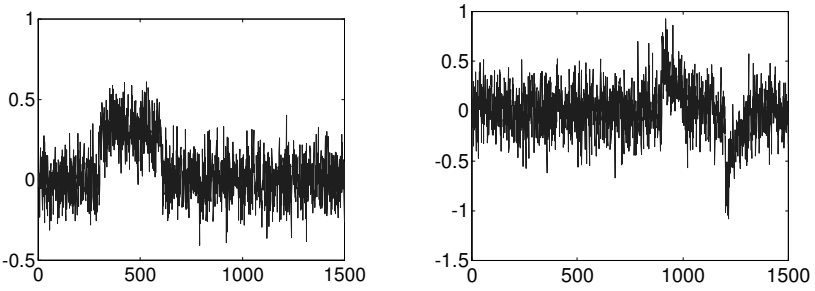
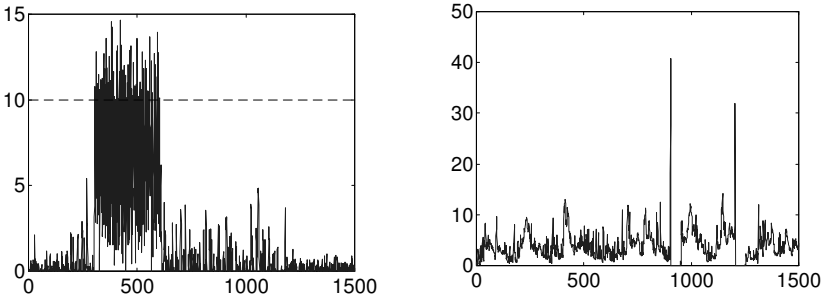


Fig. 6.26. Residual affected by f_ω (a) or f_ψ (b) only

The design of a residual generator for detection and isolation of f_ψ is based on the model made of Eqs. (6.194) and (6.195). This system is detectable and the innovation, r_ψ, of the Kalman filter based on the above model is affected by fault f_ψ as can be seen in Fig. 6.26 (b). However, the latter fault is not strongly detectable.

Hence for evaluation of residual r_ψ, one has to resort to the GLR algorithm, since it is not possible to detect fault disappearance by successive reinitialisation of a CUSUM algorithm. The latter option is possible for evaluation of r_{ω_3} however. Figure 6.27 represent the GLR decision function obtained by processing the residual sequence of Fig. 6.26 (a) and the CUSUM decision function obtained by processing the residual of Fig. 6.26 (b). Repeated alarms are issued by the CUSUM algorithm, the first occurring at time 300, the last one at time 597. In this time interval, the CUSUM decision function crosses its threshold every 5 samples on the average. Appearance and disappearance of the fault on the angular rate measurement can thus be detected and isolated. As far as the GLR decision function of Fig. 6.27 (a) is concerned, it reaches its threshold at time 904, and the estimated fault occurrence time of f_ψ is

900 (actual value 901). The estimated fault magnitude based on the residual in the time window $[900\ \ 949]$ is 0.449, which is in error by $10\ \%$. After reinitialisation, the GLR algorithm detects fault disappearance at time 1203 and it provides instant 1201 as the estimate of the change occurrence time, namely the correct time instant. The estimated change magnitude is -0.618 which is in error by $23\ \%$.



(a) (b)

Fig. 6.27. CUSUM decision function and GLR decision function resulting from evaluation of r_ψ (a) and r_{ω_3} (b)

6.9 Exercises

Exercise 6.1 *Residual generator for position actuator*
Consider the system in Fig. 3.8 and parameters given in Exercise 3.3. There is no measurement noise in the exercise.

1. Implement a candidate residual generator. Use the parity equations
$$e(s) = y_m(s) - \hat{y}(s),$$
where
$$\hat{y}_1(s) = \frac{1}{sI_{tot} + \alpha} \left(k_q \eta \, i_m(s) \right),$$
and
$$\hat{y}_2(s) = \frac{1}{Ns} \left(n_m(s) \right).$$
Investigate the properties of these potential residual generators by applying step changes on either of the faults.
2. Consider further the possible fault in the shaft speed sensor. Investigate experimentally whether all three faults can be detected and isolated.
3. Derive the transfer function matrix $\boldsymbol{H}_{yf}(s)$ and use this to explain the observations. □

Exercise 6.2 *Residual generation using the parity space approach*
This exercise deals with residual generator for the industrial actuator. Refer to Fig. 3.7. The disturbance is Q_l. The input is $i_{\rm com}$. The measurements are n_m and θ_m.

1. Determine the transfer function matrices $\boldsymbol{H}_{yu}(s)$ and $\boldsymbol{H}_{yd}(s)$.
2. Write the transfer function matrix
$$\boldsymbol{H}(s) = \begin{pmatrix} \boldsymbol{H}_{yu}(s) & \boldsymbol{H}_{yd}(s) \\ \boldsymbol{I} & \boldsymbol{O} \end{pmatrix}.$$
3. Write $\boldsymbol{H}(s)$ in the form
$$\boldsymbol{H}(s) = \frac{1}{h(s)}\tilde{\boldsymbol{H}}(s)$$
where $\tilde{H}$ is a polynomial matrix.
4. Determine the rank of $\tilde{\boldsymbol{H}}(s)$.
5. Determine the nullspace of $\tilde{\boldsymbol{H}}'(s)$.
6. From the nullspace of $\tilde{\boldsymbol{H}}'(s)$, determine residual generator(s)
$$\boldsymbol{r}(s) = \boldsymbol{V}_{ru}(s)\boldsymbol{u}(s) + \boldsymbol{V}_{ry}(s)\boldsymbol{y}(s)$$
that make the residual independent of unknown input. Verify this property by showing that
$$\boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yd} = 0.$$
7. Determine the relation
$$\boldsymbol{r}(s) = \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yf}(s) = \frac{1}{h(s)}\tilde{\boldsymbol{H}}(s)$$
and determine which of the three faults f_i, f_n and f_θ are detectable. □

Exercise 6.3 *Residual generation for single-axis satellite*
In continuation of Exercise 5.3 this exercise deals with residual generation for the single axis satellite.

A state-space model for the single axis is given by

$$\begin{aligned}
\dot{x}_1 &= \tfrac{1}{I}(ku_1 + ku_2 + w_0) \\
\dot{x}_2 &= x_1 \\
y_1 &= x_1 + f_1 \\
y_2 &= x_2 + f_2 \\
y_3 &= x_2 + f_3 \\
y_4 &= k_1 u_1 + f_4 \\
y_5 &= k_2 u_2 + f_5,
\end{aligned}$$

where x_1 is the angular velocity, x_2 the angle of the satellite and nominal parameters are

$$\begin{aligned}
I &= 14.33\ \text{kgm}^2 \\
k_1 &= k_2 = 0.5\,.
\end{aligned}$$

There are two input signals, u_1 and u_2 to actuators 1 and 2, respectively. There is one unknown input d. The magnitude of d is not known prior to the launch of the satellite, but it is known that d is constant over time.

There are five measurements: y_1 measures the state x_1, y_2 and y_3 measure the state x_2. y_4 measures the actual torque from actuator 1, y_5 measures the actual torque from actuator 2.

1. Determine the transfer function matrices $\boldsymbol{H}_{yu}(s)$ and $\boldsymbol{H}_{yd}(s)$.
2. Determine the transfer function matrix
$$\boldsymbol{H}(s) = \begin{pmatrix} \boldsymbol{H}_{yu}(s) & \boldsymbol{H}_{yd}(s) \\ \boldsymbol{I} & \boldsymbol{O} \end{pmatrix}.$$
3. Write $\boldsymbol{H}(s)$ in the form
$$\boldsymbol{H}(s) = \frac{1}{h(s)}\tilde{\boldsymbol{H}}(s)$$
where $\tilde{\boldsymbol{H}}$ is a polynomial matrix.
4. Determine the rank of $\tilde{\boldsymbol{H}}(s)$.
5. How many independent residual generators can be expected that are independent of input $\boldsymbol{u}(s)$ and of disturbances $d(s)$.
6. Find the left nullspace of $\tilde{\boldsymbol{H}}(s)$.
7. Determine a residual generator based on the nullspace. □

Exercise 6.4 *Properties of residual generators for single-axis satellite*
This exercise is a continuation of 6.3.

1. Determine the response of the residual vector to the additive faults on y_1 to y_5 by calculating
$$\boldsymbol{r}(s) = \boldsymbol{V}_{ry}(s)\boldsymbol{H}_{yf}(s)\boldsymbol{f}(s).$$
2. Determine which of the above faults are detectable and which are strongly detectable.
3. Determine which of above faults can be isolated.
As pure differentiation or integration are not feasible in the presence of measurement noise, a filter is applied on one of the residuals. Investigate the features of two proposed residual generators. Both have the form
$$\begin{aligned} r_{12}(s) &= \frac{1}{s+\alpha}y_1(s) - \frac{s}{s+\alpha}y_2(s) \\ r_{23}(s) &= y_2(s) - y_3(s). \end{aligned}$$
Version a has $\alpha = 0.01$, version b has $\alpha = 10$.
4. Discuss the properties of the two residual generators (detectability, strong detectability, isolability). Apply a fixed threshold on either set of generators to detect if a fault is present and verify your results by simulation. □

Exercise 6.5 *Residual generator design - optimisation method*
This exercise addresses the position servo from Exercise 3.2, Fig. 5.35. The exercise is to design residual generators based on the standard setup used in robust control. It is assumed that only a single fault can appear at a time.

1. Formulate the FDI problem for the system as a standard problem. Identify the matrices that need to be selected in connection with the design.
2. Design residual generators for fault detection using the standard setup and standard design methods.

3. Design a residual generator for fault isolation and fault estimation using the standard setup and standard design methods. □

Exercise 6.6 *Residual generator with an explicit specification*
This exercise addresses the position servo from Exercise 3.2, Fig. 5.36.
Assume the load possess a dominant disturbance above 0.5 rad/s.

1. Formulate a specification $\boldsymbol{H}_{zd}(s)$ and $\boldsymbol{H}_{zf}(s)$ for the design.
2. Formulate the fault detection and isolation problem for the system as a standard problem. Identify the matrices that need to be selected in connection with the design.
3. Design residual generators for fault detection sing the standard setup and standard design methods.
4. Design a residual generator for fault isolation and fault estimation using the standard setup and standard design methods. □

Exercise 6.7 *Covariance of LP filter output - input is band limited noise*
Given a low-pass filter with the state-space representation

$$\begin{aligned} \dot{x}(t) &= -\alpha x(t) + \alpha w(t) \\ y(t) &= x(t) \end{aligned}$$

with input $w(t)$, a band-limited random signal generated by

$$dw(t) = -\beta w(t)dt + \sigma_w^2 \sqrt{2\beta}\, dv(t).$$

1. Represent the filter in the form
$$d\boldsymbol{x}(t) = \boldsymbol{A}x(t) + \boldsymbol{B}d\boldsymbol{v}(t).$$
2. Let the covariance matrix be
$$\boldsymbol{Q} = E\left\{ \begin{pmatrix} w(t) \\ x(t) \end{pmatrix} \begin{pmatrix} w(t) & x(t) \end{pmatrix} \right\} = \begin{pmatrix} a & c \\ c & b \end{pmatrix}.$$
Calculate the covariance $\boldsymbol{Q}$ as the solution to the Lyapunov equation
$$\boldsymbol{A}\boldsymbol{Q} + \boldsymbol{Q}\boldsymbol{A}' + \boldsymbol{B}\boldsymbol{S}_v\boldsymbol{B}' = 0.$$
3. Show that the variance on y , σ_y^2, is
$$\sigma_y^2 = \frac{\alpha}{\alpha+\beta}\sigma_w^2 \tag{6.211}$$
and determine the value of the pole α required to obtain a desired value of σ_y^2 given σ_w^2.
□

Exercise 6.8 *Change detection for industrial actuator*
Given a simulation model and residual generators based on parity equations, this exercise deals with detection of faults in the presence of measurement noise and random disturbances.

Let a noise specification for n_m, θ_m and i_m be given by the autocorrelation function

$$R_{ii}(\tau) = \sigma_i^2 e^{-\beta_i|\tau|}$$

where

$$\begin{array}{lll} n_m: & \sigma_n = 3\,\text{rad/s} & \beta = 10 \\ \theta_m: & \sigma_\theta = 0.01\,\text{rad} & \beta = 2 \\ i_m: & \sigma_i = 0.2\,\text{A} & \beta = 10 \end{array} .$$

The noise sources are not correlated.

1. Implement a simple threshold (level) detector on the two residuals from the parity equations. Investigate whether you can detect
   a) a step change of 0.015 rad in the position sensor
   b) a step change of 0.15 A in the power drive current.
2. Design a scalar CUSUM detector of change in mean value. Test for the hypothesis that a fault is present an reflected in the a mean value change of the values given above.
3. Design a detector for the position sensor fault that has a time to detect of 2.5 s. Determine the average time between false alarms. Increase the specified time to detect to 10 s and determine the new average time between false alarms.
4. Investigate experimentally (by simulation) whether the two faults can be detected.
5. Verify the time to detect and the false alarm rates using different seeds of your measurement noise generators.
   Note 1: The results on time to detect and mean time between false alarms assume white noise statistics of the log-likelihood test quantity $s(k)$. When $s(k)$ in not white, the statistical results are only approximate figures that can only be used as guidelines for design. □

Exercise 6.9 *GLR change detection design*
As a continuation of Exercise 6.7, design a GLR estimator.

Design a scalar GLR based detector. Investigate the limits of faults that can be detected and find the threshold on the decision function that gives a time to detect of 2.5 s for a fault of similar magnitude as in Exercise 6.7. □

Exercise 6.10 *Change detection for single-axis satellite*
Referring to 6.3, measurements are subject to measurement noise. A state-space model for the single axis is given by:

$$\begin{aligned} \dot{x}_1 &= \frac{1}{I}(ku_1 + ku_2 + d) \\ \dot{x}_2 &= x_1 \\ y_1 &= x_1 + f_1 + w_1 \\ y_2 &= x_2 + f_2 + w_2 \\ y_3 &= x_2 + f_3 + w_3 \\ y_4 &= k_1 u_1 + f_4 \\ y_5 &= k_2 u_2 + f_5, \end{aligned}$$

where d is an unknown disturbance. The noise specification for w_1, w_2 and w_3 are given by the autocorrelation function

$$R_{ii}(\tau) = \sigma_i^2 e^{-\beta_i|\tau|}$$

where

$$\begin{array}{lll} w_1: & \sigma_1 = 2 \cdot 10^{-4} \text{ rad/s} & \beta = 10 \\ w_2: & \sigma_2 = 1 \cdot 10^{-5} \text{ rad} & \beta = 10 \\ w_3: & \sigma_3 = 2 \cdot 10^{-3} \text{ rad} & \beta = 10. \end{array}$$

The three noise sources are uncorrelated.

Consider two residuals that are supposed used to detect a fault f_2 in the measurement y_2.

$$\begin{aligned} r_{12}(k) &= T\sum_{i=1}^{k} y_1(i) + y_2(0) - y_2(k) \\ r_{23}(k) &= y_2(k) - y_3(k) \end{aligned} \tag{6.212}$$

where T is the sampling time of the measurements. Assume the sampling time is $T = 1$ s.

1. Calculate the variance of r_{12} and r_{23} above.
   It is desired to design a change detector such that faults larger than $2 \cdot 10^{-3}$ rad on y_2 are detected after max. 10 s (10 samples). This is not possible due to the large variance of the noise w_3 on y_3.
2. Determine which variance y_3 should have in order to meet the average time to detect as specified. Design a low-pass filter on r_{23} that will reduce the variance as required. Note that the ARL function is derived on the assumption of a white residual. As you are violating this assumption, validation by simulation is required at a later stage.
3. Design a set of scalar-based change detection algorithms for the case the fault on y_2 has a magnitude of $2 \cdot 10^{-3}$ rad and the change is a step. Verify that the desired time to detect can be obtained. □

Exercise 6.11 *Vector-based change detection for single-axis satellite*

1. Determine the fault signature in the residual vector assuming a fault on y_2 appears as a step.
2. Design a vector-based change detection algorithm for the case the fault on y_2 has a magnitude of $2 \cdot 10^{-3}$ rad and appears as a step. Discuss the properties of the vector-based change detection compared with the set of scalar algorithms. □

Exercise 6.12 *Residual generation in a Luenberger observer*

Consider the following linear time invariant system

$$\begin{aligned} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} &= \begin{pmatrix} 0 & 1 \\ -2 & -3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} \\ y &= \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + f_3 \end{aligned}$$

where $\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ is the state, $\boldsymbol{f} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}$ is the fault vector ($f = 0 \iff$ normal operation) and y is the measured output.

1. Define the parameters k_1 and k_2 of a Luenberger observer

$$\begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -2 & -3 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} + \begin{pmatrix} k_1 \\ k_2 \end{pmatrix} (y - \hat{y})$$

$$\hat{y} = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$$

   which has the following property: in the absence of faults, the estimation error

$$\begin{pmatrix} z_1 - x_1 \\ z_2 - x_2 \end{pmatrix}$$

   converges to zero with a dynamics associated with the two eigenvalues $\lambda_1 = \lambda_2 = -5$.

2. Determine the transfer function between the residual $r = y - \hat{y}$ and the fault vector $\boldsymbol{f}$ under the form

$$r = G_1(s) f_1 + G_2(s) f_2 + G_3(s) f_3. \; \square \tag{6.213}$$

Exercise 6.13 *Static and dynamic redundancy*

Consider the following system composed of 4 components: process, sensor 1, sensor 2, sensor 3.

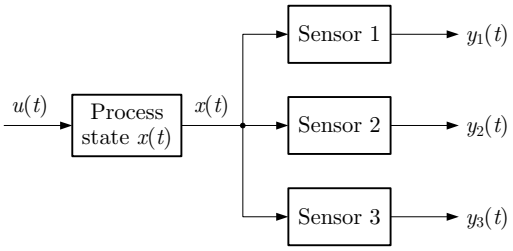


Fig. 6.28. System with three sensors

It is assumed that it can be described by the following linear time invariant model

$$\begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -2 & -3 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u(t) + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f_1(t) \\ f_2(t) \\ f_3(t) \\ f_4(t) \end{pmatrix}$$

$$\begin{pmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} f_1(t) \\ f_2(t) \\ f_3(t) \\ f_4(t) \end{pmatrix}$$

where

$$\boldsymbol{x}(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}$$

characterises the state of the process component, $u(t)$ is the control input,

$$\boldsymbol{y}(t) = \begin{pmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{pmatrix}$$

is the vector of all measurements and

$$\boldsymbol{f}(t) = \begin{pmatrix} f_1(t) \\ f_2(t) \\ f_3(t) \\ f_4(t) \end{pmatrix}$$

is the fault vector.

1. What is the association between the faults f_i, $i = 1, 2, 3, 4$ and the system components.
2. Is the state $\boldsymbol{x}(t)$ observable?
3. Is there any static redundancy in this system? What are the detectable / isolable faults?
4. Assume that during a given period of time, only sensor y_1 is operational (for example, y_2 and y_3 are disconnected for maintenance). Is it still possible to estimate the state ? to detect and isolate the faults? □

6.10 Bibliographical notes

The parity relations that were initially studied in [44],[79] and [135] are functions of a sliding window of the most recent sensor output and actuator input values. The idea used to develop parity relations in the time domain was extended to the frequency domain. This has lead to the so-called generalised parity relations [260] which do not necessarily involve only the data of a sliding window. Later this distinction between parity relations and generalised parity relations tended to disappear. The presentation given here is in the line of [187], [188]. A way to assure causality and stability of a filter involving the inverse of a transfer matrix can be found in [113]. A thorough study of the parity space approach to residual generation can also be found in [78]. The equivalence between observer-based and parity space approaches is developed in [166] for instance. Further results on the design of residual generators in the frequency domain can be found in [113].

The systematic computation of analytical redundancy relations for polynomial nonlinear models is developed in [238] and [278]. Details on elimination theory may be found in [49] and [218]. Gröbner bases used in Buchberger's algorithm [27], details and definitions can be found in [46]. The reader is referred to [81] for details as the use of characteristic sets.

A comprehensive reference to fault diagnosis treated as an optimisation problem is [168]. Earlier research results, that relate to the presentation in this book, were published in [56] and [57]. The book [43] has a chapter devoted to this subject. The design of fault diagnosis filters using the standard setup presented in this book originates in [184] and [242].

More information on threshold detection can be found in the classical presentation of this subject of [54] and, for later results, in [48] and [104].

The observer-based approach for residual generation has been the object of numerous studies, cf. to the book [194] for an introductory treatment and references on early works in this area. [43] provides more recent developments on the topic as well as a very complete list of references.

The non-sequential and sequential algorithms for change detection in signals are described in details in the book [5]. The result of Nieman and Pearson, on the optimality of tests based on the likelihood ratio between two hypotheses can be found in Section 4.2.2 of [5]. The properties of sequential algorithms deduced from the ARL function are investigated in Chapter 4 of [5]. A heuristic approach for choosing the design parameters of the GLR algorithm for detection of changes in the mean is presented in [198]. An alternative to the GLR algorithm, which is less time-consuming, is presented in [180].

A numerically stable algorithm to extract the observable part of a given system can be found on page 220 of [42]; it can be used as a first step to design a residual generator based on a Kalman filter for an unobservable system. The design of a residual generator based on a Kalman filter for a system subject to unknown input and additive faults is adapted from [181]. An alternative approach to compute an innovation sequence is to use parity relations and to filter the obtained residual by an appropriate whitening filter. This method has been considered in [198] for instance. It was not possible to examine here the question of robustness with respect to modelling uncertainties of the statistical approach to fault detection. A valuable reference to study this question is [168].

System identification based methods for fault detection, estimation and isolation have not been considered in this chapter but they have also proved useful in many applications. They can be separated in two classes: methods based on explicit parameter estimation and methods based on statistics (such as the statistical local approach). An introduction to the first class of methods can be found in [97] and [99]. For the second class, the reader is referred to [277] and [5].

Active fault detection and isolation has been briefly mentioned in this chapter. The problem of determining an optimal input signal to distinguish between different models (representing healthy and faulty modes) for a given process has been the object of a thorough study in [34] and [175]. [177] suggested novel ways to achieve active fault isolation while a plant is running.

For more information on the material in the appendix on random variables and stochastic processes, the reader can consult [4], [109] or [192] for instance. In particular the approach for sampling a linear stochastic differential equation is borrowed from pages 147-151 of [4].

Chapter 7

Fault-tolerant control of continuous-variable systems

Fault-tolerant control does not yet comprise a unique theoretic framework but employs specific ideas to treat the different problems. This chapter gives an overview of the available methods for dealing with faults in sensors, actuators and within the controlled system. The earlier results on diagnosis are employed to automatically re-design the control law. Small faults can be tackled by fault accommodation, where the controller parameters are adapted to the parameters of the faulty plant. When accommodation cannot be used like in the case of an actuator or sensor break-down, the control loop has to be reconfigured and new controller parameters determined.

7.1 The fault-tolerant control problem

7.1.1 Standard control problem

In order to explain the fault-tolerant control problem in more detail, the standard control problem is first stated as a problem that is defined by a given objective, a set of constraints and a set of admissible control laws. Standard control aims at finding a control law in a given set of control laws U, such that

the controlled system achieves the control objectives O,
while its behaviour satisfies a set of constraints C.

Thus, the solution of the problem is completely defined by the triple

$$< O, C, U > .$$

Definition 7.1 (The control problem)
Solve the problem $< O, C, U >$.

The following remarks should explain this problem in more detail:

- The set U of admissible control laws defines the algorithms that can be implemented, e.g. open-loop control (a mapping from the time domain to the control space), closed-loop control (a mapping from the output $\times$ reference spaces to the control space), using continuous or discrete-valued arguments for the variables, allowing for continuous or discontinuous, differentiable or non-differentiable mappings, etc.
- The objective O defines what the system is expected to achieve, when controlled by the above mentioned control law. It may range from very general statements (e.g. achieve closed-loop stability) to much more specific ones (e.g. reach a given point, on a given circular orbit around the earth, at a given time, for a space *rendez-vous*).
- Constraints C are functional relations that the behaviour of the controlled system must satisfy over time. They are expressed by algebraic and differential or difference equations, when continuous variables are considered, and by other models when discrete values are of interest (see Chapter 2). Inequality constraints express that some saturations act on the system admissible solutions (e.g. any trajectory which results from an admissible control law must end on a given point of a given circular orbit, at a given time, but the energy consumed all along the trajectory is limited by the capacity of the rocket's reservoirs).

Example 7.1 *Control of the single-tank system*

Consider the problem of filling an initially empty tank up to a certain mass m of some liquid, as fast as possible, so as to start a batch operation process in the food industry. Let $m(t)$ be the mass present in the tank at time t, and let $u(t)$ be the controlled input flow. The control problem is a very classical minimum time problem defined by the triple:

O: The objective is to change the mass $m(t)$ from its initial value $m(t_0) = 0$ to the given final value $m(t_f) = M$, in minimum time, i.e. minimizing the functional

$$\int_{t_0}^{t_f} dt.$$

C: The behaviour of the system is constrained by the state equation

$$\dot{m}(t) = u(t).$$

U: The control law belongs to the class of piecewise continuous open-loop controls with saturation

$$\begin{aligned} u: \quad & \mathbb{R}^+ \rightarrow \mathbb{R} \\ & t \longmapsto u(t) \\ & u(t) \in [0,\, u_{\max}] \\ & u \in \mathcal{C}^0 \end{aligned}$$

Once the tank has been filled with the mass m of liquid, the production process of the batch will start. Assume that for the proper biological reaction to take place, the temperature $\eta(t)$ must be regulated around some given reference η_{ref}. The associated control problem is again well known. A PI-regulation example is given below:

O: The objective is to regulate the temperature $\eta(t)$ of the batch around the reference value $\eta_{\rm ref}$, during the processing period $(0, \quad T)$. It is expressed by

$$|\eta_{\rm ref} - \eta(t)| \leq \lambda, \ \forall t \in [T_0, \, T] \, ,$$

where λ is some constant which defines the admissible temperature excursion around the reference on the time interval $[T_0, \, T]$, and $T_0 > 0$ is the time value after which the neighborhood of the reference temperature must have been reached.

C: The behaviour of the system is described by the state and measurement equations

$$\begin{aligned} \dot{\eta}(t) &= f(\eta(t)) + \zeta(t) + v(t) \\ \dot{\zeta}(t) &= g(\eta(t), \, t) \\ y(t) &= \eta(t) + \varepsilon(t), \end{aligned}$$

which expresses that the temperature variation is the result of thermal losses $f(\eta(t))$, of the exothermic character of the biological reaction (the reaction energy is $\zeta(t)$, whose evolution is given by $g(\eta(t), t)$ in the second state equation), and of the control $v(t)$, and that the available measurement signal is the temperature, which is corrupted by some measurement error $\varepsilon(t)$.

U: The control law belongs to the class of closed-loop PI controls

$$\begin{aligned} v : \qquad & \Psi_{\rm ref} \times Y \rightarrow \mathsf{IR} \\ & (\eta_{\rm ref}, \ y(t)) \longmapsto v(t) = k_p(\eta_{\rm ref} - y(t)) + k_i \int_0^t (\eta_{\rm ref} - y(\tau)) \, d\tau, \end{aligned}$$

where k_p and k_i are coefficients to be found as the solution of the control problem, $\Psi_{\rm ref}$ is the set of possible references and Y is the set of the sensor output values. □

7.1.2 Impacts of faults on the control problem

Fault-tolerant control is concerned with the control of the faulty system. This can be done by changing the control law without changing the plant which is operated (adaptation, accommodation, are terms often encountered in the literature), or by changing both the control and the system (in this case reconfiguration is used). Since the control algorithm just implements the solution of a control problem for a given system, changing the control or the system means that the control problem has been changed as the result of faults. In order to understand the different strategies which can be applied to the design of fault-tolerant control, let us first consider the impact of faults on the control problem $< O, \, C(\theta), \, U >$, where $C(\theta)$ denotes the dependency of the constraint C upont the parameter θ, which in turn depends upon the fault. The different fault-tolerant control strategies will then be introduced, as a consequence of the available knowledge.

System objectives. The occurence of faults should not change the system objectives. The objectives are associated with the users (they define what the users expect the system to achieve), and the very nature of fault-tolerant control is to still try to achieve these objectives, *in spite* of the faults. However, this will be possible or not. Therefore, two cases have to be distinguished:

1. There is a means of still achieving the system objectives in the presence of certain faults. The system is said to be fault tolerant, with respect to that objectives and to these faults. The control engineer's task is to design some control law which is able to do that.

2. The objectives cannot be achieved in the presence of the considered faults. The system is not fault tolerant with respect to that objectives and these faults. However, it is not enough to stand with this conclusion. The control engineer should provide, in this case, indications about what to do with the system. Since the current objectives cannot be achieved, the problem is transformed into finding new objectives that are of interest in the current situation, and to design the control law which is able to achieve these new objectives.

System constraints. The occurence of faults may obviously change the constraints $C(\theta)$ of the control problem.

- First, the constraints may remain the same but the parameters may change, thus transforming the control problem $< O, C(\theta_n), U >$ into the problem $< O, C(\theta_f), U >$, where θ_n (respectively θ_f) denotes the nominal (respectively the faulty) system parameters.
- Second, the constraints themselves might change, transforming the control problem $< O, C_n(\theta_n), U >$ into the problem $< O, C_f(\theta_f), U >$, where C_n is the set of nominal constraints, and $C_f(\theta_f)$ is a set of new constraints with new associated parameters.

Both cases may be summarised by the change of $C_n(\theta_n)$ into $C_f(\theta_f)$ since the change of parameters only is a particular case, described by $C_f = C_n$.

Example 7.2 *A tank with two exit pipes*

Consider for example a tank with two exit pipes, respectively situated at levels l_1 and l_2 meters. The system nominal constraints are the following

$$\begin{array}{rl} x(t) \in [0,\, l_1[& \dot{x}(t) = q_i(t) \\ x(t) \in [l_1,\, l_2[& \dot{x}(t) = q_i(t) - q_1(v_1, t) \\ x(t) \geq l_2 & \dot{x}(t) = q_i(t) - q_1(v_1, t) - q_2(v_2,\, t), \end{array}$$

where $x(t)$ is the level in the tank, $q_i(t)$ is the input flow, $q_1(v_1,\, t)$ (respectively $q_2(v_2, t)$) is the output flow through pipe 1 (respectively through pipe 2) which depends on some external variable or control signal v_1 (respectively v_2). This might be, for example, the level in another tank connected to the pipe, or the control signal of an output valve on the pipe. Suppose now that pipe 1 is clogged, then as the result of the fault, the system constraints become

$$\begin{array}{rl} x(t) \in [0,\, l_1[& \dot{x}(t) = q_i(t) \\ x(t) \in [l_1,\, l_2[& \dot{x}(t) = q_i(t) \quad \forall v_1(t) \\ x(t) \geq l_2 & \dot{x}(t) = q_i(t) - q_2(v_2,\, t), \end{array}$$

which can also be represented by adding a fourth constraint to the three nominal ones

$$\begin{array}{rl} x(t) \in [0,\, l_1[& \dot{x}(t) = q_i(t) \\ x(t) \in [l_1,\, l_2[& \dot{x}(t) = q_i(t) - q_1(v_1,\, t) \\ x(t) \geq l_2 & \dot{x}(t) = q_i(t) - q_1(v_1,\, t) - q_2(v_2,\, t) \\ & q_1(v_1,\, t) = 0, \forall v_1,\, t. \quad \square \end{array}$$

Admissible control laws. The occurence of faults may also change the set of admissible control laws since faults may occur in the computing and communication devices in which they are implemented. As in the previous subsection, the new set of admissible control laws is noted U_f while the nominal one is U_n.

Example 7.1 (cont.) *Control of a single-tank system*

Consider the standard control problem of filling a tank in minimum time for processing a batch in food industry: find the control law in the set U_n of piecewise continuous functions satisfying

$$\begin{array}{ll} u: & \mathbb{R}^+ \rightarrow \mathbb{R} \\ & t \longmapsto u(t) \\ & u(t) \in [0,\, u_{\max}] \\ & u \in \mathcal{C}^0 \end{array}$$

such that the initial mass $m(t_0) = 0$ is changed into the final mass $m(t_f) = M$, in minimum time, while satisfying the state equation $\dot{m}(t) = u(t)$. Suppose now that the pump is faulty and can only deliver a fraction of its nominal maximum output flow, namely $u_{\max}$ is changed into $u'_{\max} < u_{\max}$. The set U_n is obviously changed into the set U_f, where the saturation level is lower (thus obviously leading to a larger filling time for the optimal solution). $\square$

7.1.3 Passive versus active fault-tolerant control

In the passive approach, the control algorithm is designed so that the system is able to achieve its given objectives, in healthy as well as in faulty situations, without any change in the control law. Therefore, passive fault-tolerant control sets the control problem in a context, where the ability of the system to achieve its given objective is preserved, using the same control law, whatever the system situation (healthy or faulty).

In active approaches, the control law is changed when faults occur, so that the ability of the system to achieve its given objective is preserved, using a control law adapted to each fault situation. Therefore, active fault-tolerant control algorithms implement the solution of problems which are specifically set for each of the possible (healthy and faulty) situations.

As the result of faults, the control problem is transformed

$$\text{from } < O,\, C_n(\theta_n),\, U_n > \text{ into } < O,\, C_f(\theta_f),\, U_f > .$$

Suppose that both $C_f(\theta_f)$ and U_f are perfectly known, then the fault-tolerant control law has to solve $< O, C_f(\theta_f), U_f >$. If such a solution exists, the system is fault tolerant with respect to the objective O and the fault situation $C_f(\theta_f)$, U_f. If the problem $< O, C_f(\theta_f), U_f >$ has no solution, then the system is not fault tolerant and objective reconfiguration has to be explored, as previously explained.

The difference between passive and active fault-tolerant control can now be very simply explained.

Passive fault tolerance. In *passive fault tolerance*, the control law is not changed when faults occur. This means that the system objectives can be obtained when the system is healthy (thus it solves $< O, C_n(\theta_n), U_n >$), as well as when the system is faulty (thus it also solves $< O, C_f(\theta_f), U_f >$). Implementing passive fault tolerance for a given set of faults means that there is a common solution to problem $< O, C_n(\theta_n), U_n >$ and to all problems $< O, C_f(\theta_f), U_f >$, $f \in \mathcal{F}$, where $\mathcal{F}$ indexes the set of all the considered faults.

This is a very particular situation, which is fulfilled, in general, only for objectives associated with very low levels of performances (this is a so-called *conservative* approach). Note that since the control law is not changed, the passive fault tolerance approach is similar to the robust approach when uncertain systems are considered (cf. Chapter 1). Indeed, faults can be considered as uncertainties which affect the system parameters. The difference lies not only in the size and interpretation of these changes, but also in the fact that the structure of the constraints may change as the result of faults.

Active fault tolerance. In active fault tolerance, each of the problems

$$< O, C_n(\theta_n), U_n > \text{ and } < O, C_f(\theta_f), U_f >,$$

$f \in \mathcal{F}$, has its own specific solution, thus allowing for much more demanding objectives. However, for each of these problems to be solved the knowledge about $C_f(\theta_f)$ and U_f must be available. This is the role of fault detection and isolation algorithms. This chapter deals with active fault-tolerant control.

7.1.4 Available knowledge

Providing information about the fault impact is the aim of the fault diagnosis algorithms. However, the power and efficiency of these algorithms is limited. Fault detection informs that the problem to solve is no longer $< O, C_n(\theta_n), U_n >$. Fault isolation informs about the subset of the constraints $C_n(\theta_n)$ which are unchanged (those associated with the still healthy components), and the subset $U_f \subseteq U_n$ of control laws which can still be used. The knowledge about the changed constraints calls for fault estimation, which is a new function to be considered for the design of fault-tolerant control. According to its performances, three cases must be considered:

1. The fault diagnosis algorithm is able to provide an estimate $\hat{C}_f(\hat{\theta}_f)$, $\hat{U}_f$ of the fault impact. Then, the problem to be solved is the standard control problem $< O, \hat{C}_f(\hat{\theta}_f), \hat{U}_f >$. Note that, when a solution exists, there is still a risk that the actual faulty system (described by $C_f(\theta_f)$ and U_f) fails to satisfy the objectives O, although the available model of the faulty system does satisfy them.
2. The fault diagnosis algorithm is able to provide an estimate $\hat{\Gamma}_f(\hat{\Theta}_f)$, U_f of the fault impact, where $\hat{\Gamma}_f$ is a set of possible constraints and $\hat{\Theta}_f$ is a set of associated parameters. Then the problem to be solved is the robust control problem $< O, \hat{\Gamma}_f(\hat{\Theta}_f), \hat{U}_f >$. When a solution exists, the actual faulty system will satisfy the objectives O provided the actual constraints $C_f(\theta_f) \in \hat{\Gamma}_f(\hat{\Theta}_f)$, otherwise, the same risk as above exists.
3. The fault diagnosis algorithm detects and isolates the faults, but it cannot provide any estimate of the fault impact. The control engineer is faced with the problem of designing the control of a completely unknown system, which is not possible. Obtaining knowledge about that system could be thought of, using e.g. learning approaches, but then an estimation of the fault impact could indeed be obtained, which would bring the problem back to case 2.

Other possible cases are those, where the fault diagnosis system detects the fault, but it cannot isolate it, nor is it able to provide any estimate, and the case, where the fault diagnosis system does not even detect the fault. In the first case, the only possibility to keep mastering the system is to move to a fall back mode, while in the second case, any catastrophic behaviour is possible. Active fault tolerance is only concerned with cases 1, 2 and 3.

7.1.5 Active fault-tolerant control strategies

Fault accommodation. Fault accommodation is the fault-tolerant control strategy which is associated with cases 1 and 2. It solves the problem $< O, \hat{C}_f(\hat{\theta}_f), \hat{U}_f >$ or $< O, \hat{\Gamma}_f(\hat{\Theta}_f), \hat{U}_f >$, which are associated with the control of the faulty system. The fault situation can be accommodated with respect to the objectives O when the problem has a solution.

Definition 7.2 (Fault accommodation)
Solve the control problem $< O, \hat{C}_f(\hat{\theta}_f), \hat{U}_f >$, where $\hat{C}_f(\hat{\theta}_f)$ is the estimate of the actual constraints provided by fault diagnosis algorithms.

Note that the interpretation of fault accommodation is that it is a strategy by which the faulty system is controlled in a specific way, so as to still achieve the objectives which were (before the fault) achieved by the healthy system.

System reconfiguration. System reconfiguration is the fault-tolerant control strategy which is associated with case 3. Remind that in this case the faulty system

is absolutely unknown, but the control engineer wishes to design a control that achieves the system objectives. The only means to set any control problem is to switch-off the faulty components (which are known from the isolation function), and to try to achieve the objectives using only the remaining (healthy) ones. Let $C_f(\theta_f) = C'_n(\theta_n) \cup C''_f(\theta_f)$, where $C'_n(\theta_n)$ is the subset of the constraints which are associated with the healthy part of the system, and $C''_f(\theta_f)$ is the subset of the constraints which are associated with the faulty part.

Definition 7.3 (Reconfiguration)
Find a new set of system constraints $C_f(\theta_f)$ such that the control problem $< O, C_f(\theta_f), U >$ has a solution, find and activate this solution.

The choice of a new set of constraints will imply that input-output relations between controller and plant are changed.

Note that the constraints $C'_n(\theta_n)$ are known while $C''_f(\theta_f)$ are unknown. Using similar notations, let $U_f = U'_n \cup U''_f$. Then, the reconfiguration strategy solves the problem $< O, C'_n(\theta_n), U_n >$, i.e. it tries to achieve the system objectives by controlling only the healthy part of the system.

Fault accommodation and reconfiguration are distinguished according to whether the I/O signals between controller and plant are changed. Reconfiguration implies use of different I/O relations between the controller and the system. Switch of the system to a different internal structure, to change its mode of operation, is an example of such I/O switching. Accommodation does not use such means.

Both fault accommodation and system reconfiguration strategies may need new control laws in response to faults. They also have to manage transient behaviour, which result from the change of control law or change of the constraints' structure.

7.1.6 Supervision

Suppose that the accommodation and the reconfiguration strategies fail to provide a solution. This means that there is no possibility, using the faulty system (accommodation) or a subsystem of it (reconfiguration) to achieve the objective. In this case, another objective has to be provided to the system. This introduces the most general problem, defined by the 4-tuple $< \mathcal{O}, \mathcal{S}, \Theta, \mathcal{U} >$, where $\mathcal{O}$ is a set of possible control objectives. This problem is called the supervision problem. It is a decision problem in which the system objective is not pre-defined, but has to be determined, according to the system possibilities at each time, taking into account the actual system possibilities.

A supervision problem is thus a fault-tolerant control problem associated with a decision problem: if fault are such that fault tolerance cannot be achieved, the system goal itself has to be changed. When far reaching decisions with respect to the system goal have to be taken, human operators are generally involved.

It may happen that no achievable objective exists under the actual system possibilities. This can be a design error, or a deliberate choice to accept certain failure scenarios, e.g. for reasons of benefit or small likelihood of certain events. Note that

fail-to-safe conditions are intended to avoid this case in some situations, since they express that for certain classes of faults, the objective of stopping the system must always be achievable.

7.2 Fault-tolerant control architecture

The method to achieve fault tolerance, which is considered in this chapter, relies on employing fault diagnosis schemes on-line and on reacting to the results of the diagnosis. A discrete-event signal to a supervisor agent is generated by the diagnostic algorithm when a fault is detected, another when it is isolated. This activates an alternative control that is supposed to handle the fault. The control for the particular case could be pre-determined for each type of critical fault or obtained from real-time analysis and on-line redesign. In any event, the design process must run through a number of cases equal to the number of faults to be handled and the control system need to be re-designed for each such case. Some types of faults in sensors and actuators are simple to handle, others require a detailed re-design. It is therefore worthwhile to first consider the simplest possibilities, thereafter the more general and complicated case of re-design.

The architecture of a fault-tolerant control system is illustrated in Fig. 7.1. A fault is a discrete event that acts on a system and by that changes some of its properties. Having diagnosed a fault, a decision needs to be taken about a remedial action.

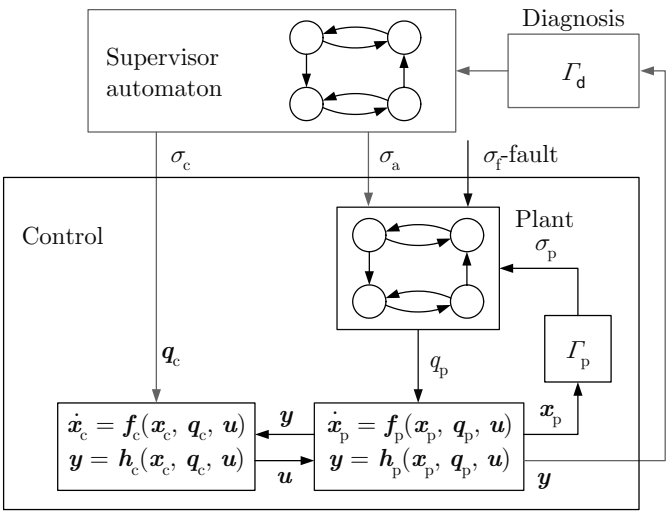


Fig. 7.1. The plant can change in a discrete way through change in states, a plant fault can cause a discrete event. Plant architecture can be changed by switch-over functions. Parameters or structure of the controller can be changed by logic in a supervisor automaton. The automaton gets its input from fault diagnosis.

The goal of fault-tolerant control is to respond to the occurrence of a fault such that the faulty system still satisfies the given specifications. Due to the discrete nature of the fault occurrence and reconfiguration, fault-tolerant control systems are hybrid in nature (cf. Section 3.7). In the figure, σ_f denotes fault events, σ_a are control events reconfiguring the system and q_c the control mode, which selects a control law. The actual physical mode q_p of the plant may be viewed as the discrete state of an automaton which is driven by plant internal events σ_p, the fault events σ_f and the control events σ_a. While many different approaches can be used to solve the fault-tolerant control problem, it may not be possible to control the system to a desired performance for an arbitrary change of parameter or structure. A final remedial action is then to close down to a safe state should proper control not be possible. The key issue is to be able to obtain certain specified properties of the control of the faulty system, and this chapter therefore focus on methods for re-design based on specifications.

A fault in the plant can affect the structure and the parameters of a plant. The complexity of designing a controller for the faulty system is therefore immense, and there is no single, systematic way to design a control system with reconfiguration as depicted. Most research work deal either with diagnosis or controller reconfiguration, but not both. One approach is based on a bank of controllers, each one being associated to a healthy or a faulty plant working mode. The selection of the controller to be used for the present working mode must be assumed to be achieved with some delay and possibly false alarms.

The theory of logic-based switching control also relies on a bank of controllers (Fig. 7.2). It has recently been used for fault-tolerant control. The supervisor is made of a set of estimators, followed by performance evaluation, and a switching logic scheme (Fig. 7.3).

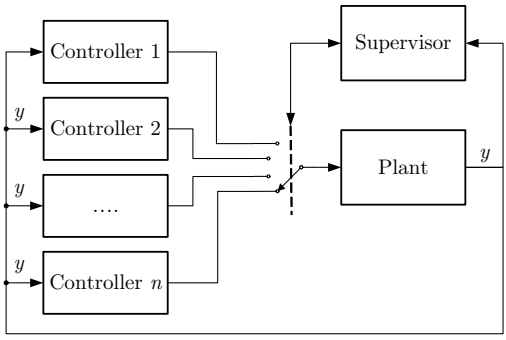


Fig. 7.2. Structure of logic-based switching controller

Each estimator reconstructs the plant output in either one of the healthy or faulty working modes. Its performance is evaluated by computing a norm of the output estimation error, and the estimator that yields the smallest performance index is

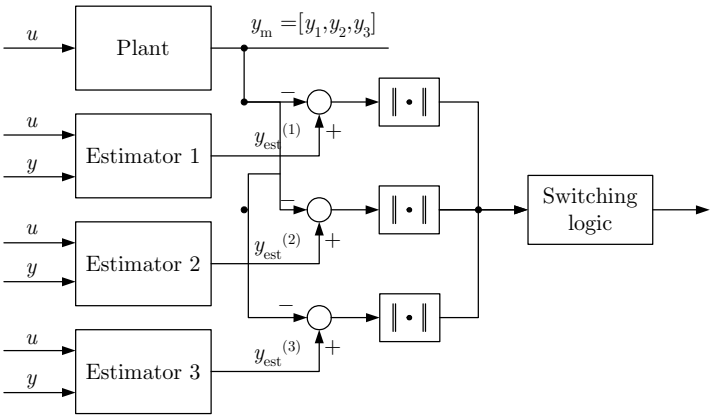


Fig. 7.3. Logic within a supervisor selects an output estimate from a bank of estimators

assumed to correspond to the present working mode. The output of the switching logic, η is the integer associated with that estimator, i.e. the estimator number. The corresponding controller is applied to the process using the switching logic.

This approach, however, presupposes that for each fault a reasonable controller has been designed before the plant is put into operation. From a practical point of view, this is not reasonable if a considerable number of faults has to be taken into account. Therefore, this chapter focuses on methods that can be applied on-line. That is, the fault-tolerant control algorithm includes a method for re-designing the controller after a fault has been identified.

In the sequel of this chapter, different approaches to the fault-tolerant control problem are presented, which refer to different objectives and faults. The linear quadratic problem under actuator faults is considered in Section 7.3, the model-matching problem under sensor and actuator faults is described in Section 7.4, the control configuration for reference-tracking and disturbance rejection in Section 7.5, where the concept of virtual sensors and actuators are introduced, and finally Section 7.6 presents some results on the general stabilisation and disturbance rejection problem, under sensor and actuator faults, using the Youla-Kucera parameterisation.

7.3 Fault-tolerant linear quadratic design

7.3.1 Control problem

Linear quadratic (LQ) problems constitute a very popular frame for control design. In this section, the LQ problem is analysed with respect to the possible occurrence of actuator faults. It is shown that fault tolerance can only be achieved if admissible (rather than optimal) solutions exist. Conditions on an actuator fault to be possibly

tolerated are given both for the fault accommodation and for the system reconfiguration strategies.

Consider the system whose nominal operation is modelled by

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \\ &= \boldsymbol{A}\boldsymbol{x}(t) + \sum_{i \in I} \boldsymbol{B}_i \, \boldsymbol{u}_i(t). \end{aligned} \tag{7.1}$$

$\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^n$ is the state vector and $\boldsymbol{u} \in \mathcal{U} \subset \mathbb{R}^m$ is the control vector. I is the set of the actuators, $\boldsymbol{u}_i(t) \in \mathbb{R}^{m_i}$ is the input of actuator $i \in I$, and $m = \sum_{i \in I} m_i$. $\boldsymbol{A}$ and $\boldsymbol{B}$ are constant matrices of suitable dimensions, and it is assumed that the pair $(\boldsymbol{A}, \boldsymbol{B})$ is controllable. The following optimal control problem is considered:

Problem 7.1 (Optimal control problem)

1. **Objective** O**:** *Transfer the system state from* $\boldsymbol{x}(0) = \boldsymbol{\gamma}$ *towards* $\boldsymbol{x}(\infty) = \boldsymbol{0}$*, where* $\boldsymbol{\gamma} \in \mathbb{R}^n$*, and* $\boldsymbol{x}(\infty)$ *stands for* $\lim_{t \longrightarrow \infty} \boldsymbol{x}(t)$ *while minimising the functional*

$$J(\boldsymbol{u}, \boldsymbol{\gamma}) = \frac{1}{2} \int_0^\infty \left(\boldsymbol{u}'(t) \boldsymbol{R} \boldsymbol{u}(t) + \boldsymbol{x}'(t) \boldsymbol{Q} \boldsymbol{x}(t) \right) dt, \tag{7.2}$$

   where $\boldsymbol{Q}$ *and* $\boldsymbol{R}$ *are symmetric matrices, and* $\boldsymbol{Q} \geq 0, \boldsymbol{R} > 0$*.*

2. **Constraints** C*: Equation (7.1) is satisfied* $\forall t \in [0, \infty)$*,* $\boldsymbol{x}(t)$ *and* $\boldsymbol{u}(t)$ *are continuous functions of time, and* $\mathcal{X} = \mathbb{R}^n$*,* $\mathcal{U} = \mathbb{R}^m$ *hold.*

7.3.2 Control of the nominal plant

The solution of Problem (7.2) is well known from the classical theory of optimal control. Let $H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t)$ be the system Hamiltonian

$$H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t) = -\frac{1}{2} \left(\boldsymbol{u}'(t) \boldsymbol{R} \boldsymbol{u}(t) + \boldsymbol{x}'(t) \boldsymbol{Q} \boldsymbol{x}(t) \right) + \boldsymbol{p}'(t) \left(\boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \right),$$

where $\boldsymbol{p}(t)$ is the adjoint state vector, then the necessary optimality condition is

$$\dot{\boldsymbol{x}} = \frac{\partial H}{\partial \boldsymbol{p}} (\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}(t) \tag{7.3}$$

$$\dot{\boldsymbol{p}} = -\frac{\partial H}{\partial \boldsymbol{x}} (\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t) = \boldsymbol{Q}\boldsymbol{x} - \boldsymbol{A}'\boldsymbol{p} \tag{7.4}$$

$$\boldsymbol{0} = \frac{\partial H}{\partial \boldsymbol{u}} (\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t) = \boldsymbol{u} - \boldsymbol{R}^{-1} \boldsymbol{B}' \boldsymbol{p}.$$

It is easy to show that the optimal solution is given by

$$\begin{aligned} \boldsymbol{p}(t) &= -\boldsymbol{P}\boldsymbol{x}(t) \\ \boldsymbol{u}(t) &= -\boldsymbol{R}^{-1} \boldsymbol{B}' \boldsymbol{P}\boldsymbol{x}(t), \end{aligned}$$

where $\boldsymbol{K}$ is the (symmetric) solution of the algebraic Riccatti equation

$$\boldsymbol{Q} + \boldsymbol{A}'\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\,\boldsymbol{B}'\boldsymbol{P} = \boldsymbol{0}$$

such that the closed-loop system

$$\dot{\boldsymbol{x}}(t) = \left(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{R}^{-1}\,\boldsymbol{B}'\boldsymbol{P}\right)\boldsymbol{x}(t)$$

is stable. The solution exists since the pair $(\boldsymbol{A}, \boldsymbol{B})$ is controllable, and the optimal value of the criterion is given by

$$J\,(0,\,\emptyset,\,\boldsymbol{\gamma}) = \boldsymbol{\gamma}'\boldsymbol{P}\boldsymbol{\gamma}, \tag{7.5}$$

where the argument $0,\,\emptyset$ recalls that there is no faulty actuator on the time window $[0,\,\infty)$.

Nominal performances. Equation (7.5) shows that the nominal performance of the actuator set I depends on the value of $\boldsymbol{\gamma}$.

$$\mathit{\Gamma} = \{\boldsymbol{\gamma} \in \mathbb{R}^n,\ \text{s.t. } \boldsymbol{\gamma}'\boldsymbol{P}\boldsymbol{\gamma} \leq 1\}$$

represents the set of points in the state space from which the origin can be reached with a cost less than 1. The characterisation of the actuation scheme I independently of the control objective $\boldsymbol{\gamma}$ leads to consider the worst control problem from the quadratic criterion point of view: transfer the system state from $\boldsymbol{x}(0) = \boldsymbol{\gamma}^*$ to $\boldsymbol{x}(\infty) = \boldsymbol{0}$, where

$$\boldsymbol{\gamma}^* = \arg \max_{|\boldsymbol{\gamma}|=1} J\,(0, \emptyset, \boldsymbol{\gamma})\,.$$

The set of actuators I is thus characterised by the maximum eigenvalue of $\boldsymbol{P}$ which is interpreted as the maximum cost which might be spent in transferring the system state from $\boldsymbol{x}(0) = \boldsymbol{\gamma}$ to $\boldsymbol{x}(\infty) = \boldsymbol{0}$ for some $\boldsymbol{\gamma} \in \mathbb{R}^n$ such that $|\boldsymbol{\gamma}| = 1$

$$J\,(0, \emptyset, \boldsymbol{\gamma}^*) = \lambda_{\max}(\boldsymbol{P}). \tag{7.6}$$

7.3.3 Fault tolerance with respect to actuator faults

This section considers the situation in which the system is faultless until the time instant t_f and has afterwards a fault in one or several actuators. Hence, the whole set of actuators I is healthy in the time interval $(0,\, t_f[$ while there is a subset I_F of faulty actuators in the interval $[t_f,\, \infty)$. Let $I = I_N \cup I_F$, where I_N is the subset of the still normal actuators. After t_f the faulty system behaviour is described by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \sum_{i \in I_N} \boldsymbol{B}_i\,\boldsymbol{u}_i\,(t) + \sum_{i \in I_F} \boldsymbol{\beta}_i\,\left(\boldsymbol{u}_i(t),\,\theta_i\right), \tag{7.7}$$

where $\boldsymbol{\beta}_i\,(\boldsymbol{u}_i(t),\,\theta_i)$ describes the contribution of the faulty actuator i. This vector may be known, known with unknown parameters θ_i or completely unknown,

depending on the faults which are considered, and of the capability of the fault diagnostic algorithm to estimate them. The objective, constraints and criterion of the fault-tolerant control problem are identical to those of the control problem, with the exception of constraint (7.1) being valid on $(0,\ t_f[$ and being replaced by constraint (7.7) on $[t_f,\ \infty)$.

Problem constraints. Two cases can be considered as far as the status of constraint (7.7) is concerned.

1. In the first case, the fault tolerance analysis is done (off-line) for given faults, which are known to possibly occur in the considered system (from the failure-modes and effect analysis, for example). Therefore, constraint (7.7) *is known* and the fault-tolerant control can be designed beforehand (but it can be applied on-line only when the actual fault matches the fault for which it has been designed, which needs the actual fault to be identified).

2. In the second case, the analysis is done for any kind of fault which might occur during the system operation and, therefore, constraint (7.7) *being not known*, has again to be identified (or replaced by another constraint if identification is impossible or not available).The identification of the subset I_F of faulty actuators is normally done by the fault diagnostic algorithm, which detects and isolates the faults. Defining the constraints resumes to identifying the functions $\boldsymbol{\beta}_i(\boldsymbol{u}_i(t), \theta_i), i \in I_F$. This is not usually done by fault diagnostic algorithms, and could be referred to as a *diagnostic* (or fault estimation) possibility, which rests on fault modelling and on fault parameter identification, and it could be - or not - provided by the fault diagnostic system.

Therefore, the two approaches to fault-tolerant control can be applied in dependence upon the situation. Fault accommodation consists of controlling the faulty system after replacing Eq. (7.7) by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \sum_{i \in I_N} \boldsymbol{B}_i\, \boldsymbol{u}_i(t) + \sum_{i \in I_F} \hat{\boldsymbol{\beta}}_i \left(\boldsymbol{u}_i(t),\, \hat{\boldsymbol{\theta}}_i\right), \tag{7.8}$$

where the functions $\hat{\boldsymbol{\beta}}_i\,(\boldsymbol{u}_i(t),\, \hat{\boldsymbol{\theta}}_i)$ and parameters $\hat{\boldsymbol{\theta}}_i,\ i \in I_F$ are estimated. System reconfiguration consists of controlling only the healthy part of the system (thus switching off the faulty actuators), which means replacing Eq. (7.7) by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \sum_{i \in I_N} \boldsymbol{B}_i\, \boldsymbol{u}_i(t). \tag{7.9}$$

Admissible solutions. Whatever the selected strategy, a solution to the fault-tolerant control problem exists provided the objective $\boldsymbol{x}(\infty) = \boldsymbol{0}$ can still be reached from the initial state $\boldsymbol{x}(0) = \boldsymbol{\gamma}$. When solutions are needed to exist for any objective $\boldsymbol{\gamma}$, it is obviously necessary that the system (7.7) or (7.9) is still controllable.

Suppose that the fault-tolerant control problem has a solution, i.e. the system state can be transferred from $\boldsymbol{x}(0) = \boldsymbol{\gamma}$ to $\boldsymbol{x}(\infty) = \mathbf{0}$, and introduce the notation

$$J\left((0,\, t_f)\,,\ (\emptyset,\, I_F)\,,\, \boldsymbol{\gamma}\right) \tag{7.10}$$

for the minimal cost associated with the two time instants $(0,\, t_f)$ at which the failed actuators are respectively $(\emptyset,\, I_F)$. Obviously, the fact that a solution exists does not mean that it is satisfactory. Two cases can be distinguished.

- The cost is of no importance provided the system objective is achieved in spite of the fault. In this case, the actuation scheme I is fault tolerant with respect to the situation I_F occurring at time t_f if and only if system (7.8) - when accommodation is used - or (7.9) - when reconfiguration is concerned - is controllable.
- Some cost limitation is considered. Although optimal, the cost might be too high, thus denying the actuation scheme I to deserve the "fault-tolerant" label with respect to the situation I_F.

Definition 7.4 (Admissibility)
Let I_F be a fault situation occurring at time t_f. The solution of the fault-tolerant control problem is admissible with respect to the control objective $\boldsymbol{\gamma}$ if and only if

$$J\left((0,\, t_f)\,,\ (\emptyset,\, I_F)\,,\, \boldsymbol{\gamma}\right) \leq \rho(\boldsymbol{\gamma}) J\left(0,\, \emptyset,\, \boldsymbol{\gamma}\right), \tag{7.11}$$

where $\rho(\boldsymbol{\gamma}) \geq 1$ is some predefined function.

Obviously, $\rho(\boldsymbol{\gamma})$ is the maximal loss of efficiency which can be admitted when a control solution, which still achieves the objective $\boldsymbol{\gamma}$, but under the situation, where the fault I_F occurs at time t_f is used. Three special choices of $\rho(\boldsymbol{\gamma})$ may be of interest.

- $\rho(\boldsymbol{\gamma}) = \infty,\ \forall \boldsymbol{\gamma} \in \mathbb{R}^n$.
  In this case, fault tolerance is only concerned with the existence of an optimal solution, whatever its cost, thus resuming the fault tolerance property to the permanence of the controllability property,

- $\rho(\boldsymbol{\gamma}) = \frac{\sigma}{\boldsymbol{\gamma}' P \boldsymbol{\gamma}},\ \forall \boldsymbol{\gamma} \in \mathbb{R}^n,\ |\boldsymbol{\gamma}| \leq 1$
  defines a uniform bound σ for the cost of controlling the faulty system, whatever the initial state in the unit sphere,

- $\rho(\boldsymbol{\gamma}) = \rho^*,\ \forall \boldsymbol{\gamma} \in \mathbb{R}^n$
  defines a uniform bound for the loss of efficiency in the control of the faulty system, whatever the control objective.

Based on the definition of admissibility, fault tolerance can be defined as follows.

Definition 7.5 (Fault tolerance of a system subject to actuator faults)
The actuation scheme I is fault tolerant with respect to the fault I_F occurring at time t_f for the control objective $\boldsymbol{\gamma}$ if and only if the accommodation or the reconfiguration problem has an admissible solution.

7.3.4 Fault accommodation

The accommodation strategy is now analysed for the system described by

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \sum_{i\in I} \boldsymbol{B}_i\, \boldsymbol{u}_i(t) \quad \text{for } t \in [0,\, t_f[\\ \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \sum_{i\in I_N} \boldsymbol{B}_i\, \boldsymbol{u}_i(t) + \sum_{i\in I_F} \boldsymbol{\beta}_i\, (\boldsymbol{u}_i(t),\, \theta_i)\,, \boldsymbol{x}(t_f) = \boldsymbol{x}_f, \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{for } t \in [t_f,\, \infty)\,.\end{aligned}$$

Identifying the faulty system. Since the functions $\boldsymbol{\beta}_i\,(\boldsymbol{u}_i(t),\, \theta_i)$ and parameters $\theta_i,\, i \in I_F$ are not known, they must be estimated, and therefore the LQ control problem is set for the model

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \sum_{i\in I} \boldsymbol{B}_i\, \boldsymbol{u}_i(t) \quad \text{for } t \in [0,\, t_f[\\ \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \sum_{i\in I_N} \boldsymbol{B}_i\, \boldsymbol{u}_i(t) + \sum_{i\in I_F} \hat{\boldsymbol{\beta}}_i\, \left(\boldsymbol{u}_i(t),\, \hat{\theta}_i\right) \quad \text{for } t \in [t_f,\, \infty)\,,\end{aligned}$$

where the functions $\hat{\boldsymbol{\beta}}_i\, \left(\boldsymbol{u}_i(t),\, \hat{\boldsymbol{\theta}}_i\right)$ and parameters $\hat{\boldsymbol{\theta}}_i,\, i \in I_F$ are known. This approach obviously needs some fault model to be defined, and its parameters to be identified.

Assume it is known that the faulty actuators can still be described by a linear model

$$\hat{\boldsymbol{\beta}}_i\, \left(\boldsymbol{u}_i(t),\, \hat{\boldsymbol{\theta}}_i\right) = \hat{\boldsymbol{B}}_i\, \boldsymbol{u}_i(t),\, i \in I_F$$

and, therefore, the model of the faulty system is

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= \boldsymbol{A}x(t) + \sum_{i\in I_N} \boldsymbol{B}_i u_i(t) + \sum_{j\in I_F} \hat{\boldsymbol{B}}_j u_j(t) \\ &= \boldsymbol{A}x(t) + \boldsymbol{B}_f(t),\end{aligned} \tag{7.12}$$

where $\boldsymbol{B}_f = (\boldsymbol{B}_N, \hat{\boldsymbol{B}}_F)$ is the new actuator matrix, formed by the concatenation of the $\boldsymbol{B}_i$ matrices associated with the healthy actuators $\boldsymbol{B}_N = (\boldsymbol{B}_i, i \in I_N)$ and the $\hat{\boldsymbol{B}}_j$ matrices associated with the faulty actuators $\hat{\boldsymbol{B}}_F = (\hat{\boldsymbol{B}}_j, j \in I_F)$

Accomodating the control to the faulty system. From Bellman's optimality principle, the accommodation strategy consists of applying the optimal control solution to system (7.12), with initial condition $\boldsymbol{x}_f = \boldsymbol{x}(t_f)$, on the time interval $[t_f,\, \infty)$, thus leading to compute the accommodated control and trajectories as the solution of

$$\begin{aligned}\dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}_f\boldsymbol{u}(t) \\ \dot{\boldsymbol{p}} &= \boldsymbol{Q}\boldsymbol{x} - \boldsymbol{A}'\boldsymbol{p} \\ \boldsymbol{u} &= \boldsymbol{R}^{-1}\, \boldsymbol{B}_f'\boldsymbol{p}\end{aligned} \tag{7.13}$$

with the result that the value of the criterion is now

$$J\left((0,\, t_f),\, (\emptyset,\, I_F),\, \boldsymbol{\gamma})\right) = J_{0f} + \boldsymbol{x}_f' \boldsymbol{P}_f \boldsymbol{x}_f \tag{7.14}$$

instead of

$$J(0,\, \emptyset,\, \boldsymbol{\gamma}) = \boldsymbol{\gamma}' \boldsymbol{P} \boldsymbol{\gamma},$$

where J_{0f} is the cost already spent between $t = 0$ and $t = t_f$ and $\boldsymbol{P}_f$ is the solution of the algebraic Riccatti equation in which $\boldsymbol{B}$ has been replaced by $\boldsymbol{B}_f$, namely

$$\boldsymbol{Q} + \boldsymbol{A}' \boldsymbol{P}_f + \boldsymbol{P}_f\, \boldsymbol{A} - \boldsymbol{P}_f\, \boldsymbol{B}_f\, \boldsymbol{R}^{-1}\, \boldsymbol{B}_f'\, \boldsymbol{P}_f = \boldsymbol{0}. \tag{7.15}$$

Testing the admissibility of the accommodated control. From simple calculations, and taking into account that

$$J(0,\, \emptyset,\, \boldsymbol{\gamma}) = J_{0f} + \boldsymbol{x}_f'\, \boldsymbol{P} \boldsymbol{x}_f$$

one has

$$J_{0f} = \boldsymbol{\gamma}' \boldsymbol{P} \boldsymbol{\gamma} - \boldsymbol{x}_f'\, \boldsymbol{P} \boldsymbol{x}_f$$

and therefore

$$J\left((0,\, t_f),\, (\emptyset,\, I_F),\, \boldsymbol{\gamma})\right) = \boldsymbol{\gamma}' \boldsymbol{P} \boldsymbol{\gamma} + \boldsymbol{x}_f'\, \left(\boldsymbol{P}_f - \boldsymbol{P}\right) \boldsymbol{x}_f. \tag{7.16}$$

From (7.14) and the different definitions of admissibility, the set of triples

$$(\boldsymbol{B}_f, t_f, \boldsymbol{\gamma})$$

which can be tolerated by an accommodation strategy are characterised as follows:

- $\rho(\boldsymbol{\gamma}) = \infty,\ \forall \boldsymbol{\gamma} \in \mathbb{R}^n$

$$(\boldsymbol{A},\, \boldsymbol{B}_f) \text{ controllable} \tag{7.17}$$

- $\rho(\boldsymbol{\gamma}) = \frac{\sigma}{\boldsymbol{\gamma}' \boldsymbol{P} \boldsymbol{\gamma}},\ \forall \boldsymbol{\gamma} \in \mathbb{R}^n,\ |\boldsymbol{\gamma}| \leq 1$

$$\begin{aligned} &(\boldsymbol{A},\, \boldsymbol{B}_f) \text{ controllable} \\ &\boldsymbol{x}_f'\, \left(\boldsymbol{P}_f - \boldsymbol{P}\right) \boldsymbol{x}_f \leq \sigma - \boldsymbol{\gamma}' \boldsymbol{P} \boldsymbol{\gamma} \end{aligned} \tag{7.18}$$

- $\rho(\boldsymbol{\gamma}) = \rho^*,\ \forall \boldsymbol{\gamma} \in \mathbb{R}^n$

$$\begin{aligned} &(\boldsymbol{A},\, \boldsymbol{B}_f) \text{ controllable} \\ &\boldsymbol{x}_f'\, \left(\boldsymbol{P}_f - \boldsymbol{P}\right)\, \boldsymbol{x}_f \leq (\rho^* - 1)\, \boldsymbol{\gamma}' K \boldsymbol{\gamma} \end{aligned} \tag{7.19}$$

Note that these conditions depend on the value of the state $\boldsymbol{x}_f$ at the time of the fault occurrence, which is computed by

$$\boldsymbol{x}_f = e^{\boldsymbol{A} t_f}\, \boldsymbol{\gamma} + \int_0^{t_f} e^{\boldsymbol{A}(t_f - t)}\, \boldsymbol{B} \boldsymbol{u}(t)\, dt,$$

where $\boldsymbol{u}(t)$ is the optimal control computed from (7.4), and can also be expressed as

$$\boldsymbol{x}_f = e^{(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{P})t_f}\boldsymbol{\gamma}.$$

Since t_f is obviously unknown beforehand, these conditions can only be checked on-line, at time t_f when the fault is detected, isolated and diagnosed. Of course, it might be unpleasant to discover on-line that the fault that just occurred cannot be accommodated. Therefore, it is interesting to look for sufficient conditions, which could be checked off-line. Such conditions can be found under the reasonable assumption that if the objective can be reached by an admissible control using the faulty system from the beginning, then it can also be reached by an admissible control when the nominal system is first used and replaced (at an unknown time) by the faulty one.

This assumption is obviously satisfied as it can be seen by considering the worst case value of $\boldsymbol{x}_f$ in the previous conditions. Under the assumption that $(\boldsymbol{P}_f - \boldsymbol{P}) \geq 0$ (which is reasonable since it states that the faulty actuators are less efficient than the healthy ones), the worst case situation is that in which the fault occurs right at time $t_f = 0$, and therefore one has $\boldsymbol{x}_f = \boldsymbol{\gamma}$, which leads to the sufficient conditions (7.20) – (7.22) for the fault I_F to be tolerated using an accommodation strategy. Note that these conditions characterise all the pairs $(\boldsymbol{B}_f, \boldsymbol{\gamma})$ for which the system is fault tolerant, whatever the time at which the fault $\boldsymbol{B}_f$ occurs.

- $\rho(\boldsymbol{\gamma}) = \infty, \forall \boldsymbol{\gamma} \in \mathbb{R}^n$

$$(\boldsymbol{A},\, \boldsymbol{B}_f) \text{ controllable} \tag{7.20}$$

- $\rho(\boldsymbol{\gamma}) = \frac{\sigma}{\boldsymbol{\gamma}'\boldsymbol{P}\boldsymbol{\gamma}},\ \forall \boldsymbol{\gamma} \in \mathbb{R}^n,\ |\boldsymbol{\gamma}| \leq 1$

$$\begin{array}{c}(\boldsymbol{A},\, \boldsymbol{B}_f) \text{ controllable} \\ \boldsymbol{\gamma}'\boldsymbol{P}_f\boldsymbol{\gamma} \leq \sigma\end{array} \tag{7.21}$$

- $\rho(\boldsymbol{\gamma}) = \rho^*,\ \forall \boldsymbol{\gamma} \in \mathbb{R}^n$

$$\begin{array}{c}(\boldsymbol{A},\, \boldsymbol{B}_f) \text{ controllable} \\ \boldsymbol{\gamma}'\left(\boldsymbol{P}_f - \rho^* \cdot \boldsymbol{P}\right)\boldsymbol{\gamma} \leq 0\end{array} \tag{7.22}$$

The conditions under which the fault $\boldsymbol{B}_f$ can be tolerated for any objective $\boldsymbol{\gamma}$, whatever the time at which it occurs, are obviously given by

- $\rho(\boldsymbol{\gamma}) = \infty,\ \forall \boldsymbol{\gamma} \in \mathbb{R}^n$

$$(\boldsymbol{A},\, \boldsymbol{B}_f) \text{ controllable,} \tag{7.23}$$

- $\rho(\boldsymbol{\gamma}) = \frac{\sigma}{\boldsymbol{\gamma}'\boldsymbol{P}\boldsymbol{\gamma}},\ \forall \boldsymbol{\gamma} \in \mathbb{R}^n,\ |\boldsymbol{\gamma}| \leq 1$

$$\begin{array}{c}(\boldsymbol{A},\, \boldsymbol{B}_f) \text{ controllable} \\ \lambda_{\max}\left(\boldsymbol{P}_f\right) \leq \sigma,\end{array} \tag{7.24}$$

- $\rho(\boldsymbol{\gamma}) = \rho^*,\ \forall \boldsymbol{\gamma} \in \mathbb{R}^n$

$$\begin{aligned} &(\boldsymbol{A},\ \boldsymbol{B}_f) \text{ controllable} \\ &\lambda_{\max}\left(\boldsymbol{P}_f - \rho^* \cdot \boldsymbol{P}\right) \leq 0. \end{aligned} \tag{7.25}$$

7.3.5 Control reconfiguration

Reconfiguration strategies set the control problem of a system in which the faulty part has been switched off. The choice of a reconfiguration strategy might follow from the impossibility of estimating the fault, or it can be deliberate, so as to implement fault-tolerant strategies which provide guaranteed results, and are as simple and as understandable as possible by operators. In many cases, reconfiguration is understood as the replacement of the faulty part by some non-faulty one. Considering the problem under investigation, this means that some actuators were not in service before the fault occurrence and that they can be switched on after the fault.

Let I_{off} be the set of those actuators, which are assumed without loss of generality to be non-faulty. It obviously follows that considering from the beginning the whole set of actuators $I \cup I_{off}$ reduces the problem to that of reconfiguring the system $I_{off} \cup I_N \cup I_F$ by simply removing the faulty part. Thus, including I_{off} within I (namely, into I_N), one can go on with unchanged notations. In this situation, the fault-tolerant control problem has to be analysed replacing Eq. (7.7) by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \sum_{i \in I_N} \boldsymbol{B}_i\, \boldsymbol{u}_i(t).$$

Therefore, all the preceding results and statements also apply to the reconfiguration strategy, provided $\boldsymbol{B}_f = \left(\boldsymbol{B}_N,\ \hat{\boldsymbol{B}}_F\right)$ is replaced by $\boldsymbol{B}_f = (\boldsymbol{B}_N,\ \boldsymbol{O})$.

It is easily seen that $\boldsymbol{B}_f$ only depends on the subset of actuators I_N whatever the faults that act on the subset of actuators I_F. Therefore, neither is it needed to assume that the faulty actuators be described by a linear model $\hat{\boldsymbol{B}}_f$, nor is it needed to identify this model. Moreover, since there is only a finite number of actuator subsets, the reconfigured controls can be computed off-line for each possible subset I_F, and the solution can be switched on-line as soon as the FDI has provided the current subset of faulty actuators (this needs only fault detection and isolation). Note that for some subsets I_F an admissible solution will not exist, therefore it is of interest to analyse off-line all the possible subsets of faulty actuators.

The lattice of actuators subsets. Since I is the set of all actuators in the system, the power set 2^I is the set of all possible actuator subsets. According to the fact that the pair $(\boldsymbol{A}, \boldsymbol{B}_f)$ associated with a given subset I_N satisfies or not conditions (7.23)-(7.25), 2^I can be partitioned into

$$2^I = 2^{I+} \cup 2^{I-},$$

where

$$\begin{aligned} 2^{I+} &= \{I_N \subseteq I : \text{the fault } I_F = I \setminus I_N \text{ can be tolerated}\} \\ 2^{I-} &= \{J \subset I : \text{the fault } I_F = I \setminus I_N \text{ cannot be tolerated}\} \end{aligned}$$

It is well known that power sets have a lattice structure. That means that 2^I can be represented by a hierarchical graph, where nodes are actuator subsets organised into levels as follows:

- level 0 contains only I,
- level 1 contains all subsets I_N such that I_F has only one element,
- level 2 contains all subsets I_N such that I_F has two elements,
- etc ...
- the last level is the empty set (I_F contains all actuators I).

Each level is partitioned into nodes which belong to 2^{I+} to 2^{I-}. Edges connect nodes which belong to adjacent levels and differ by only one actuator. Let I_N be a node, $F(I_N)$ the set of its followers and $P(I_N)$ the set of its predecessors. One has

$$\begin{aligned} F(I_N) &= \{I' = I_N \setminus \{\sigma\}, \sigma \in I_N\} \\ P(I_N) &= \{I'' = I_N \cup \{\sigma\}, \sigma \in I \setminus I_N\} \end{aligned}$$

From this definition it is easy to define $F^k(I_N)$ - resp. $P^k(I_N)$ - the set of all followers of I_N - resp. of all predecessors - that differ from I_N by exactly k actuators, for any $k = 1, 2, .. |I_N|$.

Example 7.3 *Reconfiguration after actuator faults*

Consider a system with 7 states, and 4 actuators: $I = \{a, b, c, d\}$. The matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are as follows:

$$\begin{aligned} \boldsymbol{A} &= \text{diag}\ \{-1, -0.5, -3, -4, -2, -1.5, -2.5\}, \\ \tilde{\boldsymbol{B}} &= \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}. \end{aligned}$$

The considered criterion is

$$J(\boldsymbol{u}, \boldsymbol{\gamma}) = \frac{1}{2} \int_0^\infty \boldsymbol{u}'(t)\boldsymbol{u}(t)\, dt,$$

which means that only the control energy is of interest and $\boldsymbol{R}$ is the identity matrix. In that case, it is known that

$$J(I, 0, \boldsymbol{\gamma}) = \boldsymbol{\gamma}' \boldsymbol{W}_c^{-1} \boldsymbol{\gamma},$$

where $\boldsymbol{W}_c$ is the Gramian associated with the pair $(\boldsymbol{A}, \boldsymbol{B})$, i.e.

$$\boldsymbol{W}_c = \int_0^\infty e^{\boldsymbol{A}t} \boldsymbol{B}\boldsymbol{B}' \left(e^{\boldsymbol{A}t}\right)' dt.$$

The maximal eigenvalue is $\lambda_{\max}\left(\boldsymbol{W}_c^{-1}(I)\right) = 0.4357$ energy units.

Assume that admissible solutions are defined such that the worst situation control cost should not exceed 1.125 energy units. Then, there are 10 fault situations in which the system is controllable by reconfiguration, namely when only actuators $\{a, b, c, d\}$, $\{b, c, d\}$,

$\{a, c, d\}$, $\{a, b, d\}$, $\{a, b, c\}$, $\{c, d\}$, $\{b, c\}$, $\{a, d\}$, $\{a, b\}$, $\{a, c\}$ remain available, but only 6 of them are admissible when energy limitation is considered, as shown by Table 7.1.

Table 7.1 Admissible actuators subsets and associated characteristics

Actuator subsets	$\lambda_{\max}$ (energy units)
$\{a, b, c, d\}$	0.4357
$\{b, c, d\}$	1.1197
$\{a, c, d\}$	0.4676
$\{a, b, d\}$	0.8274
$\{a, b, c\}$	0.4778
$\{\boldsymbol{c}, \boldsymbol{d}\}$	**3.0201**
$\{\boldsymbol{b}, \boldsymbol{c}\}$	**1.3948**
$\{\boldsymbol{a}, \boldsymbol{d}\}$	**2.2576**
$\{a, b\}$	1.0612
$\{\boldsymbol{a}, \boldsymbol{c}\}$	**1.1452**

Figure 7.4 shows the actuators lattice and its five levels. Dark grey nodes are subsets of actuators that cannot control the system (the corresponding pair $(\boldsymbol{A}, \boldsymbol{B}_f)$ is not controllable), light grey nodes are subsets of actuators by which the system is controllable, but energy limitations are not met, white nodes are subsets of actuators which allow to control the system in an admissible way. Grey nodes correspond to faults that cannot be tolerated. □

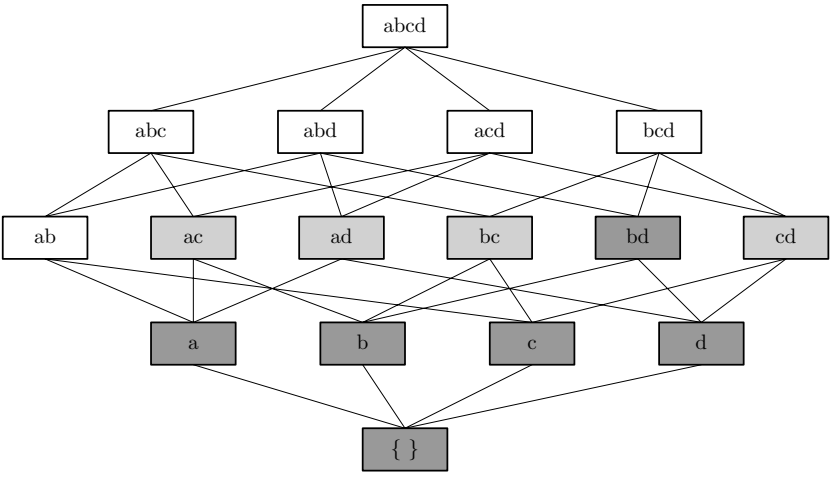


Fig. 7.4. The lattice of the actuators subsets in the example

Discrete state behaviour of the actuation system. Define the discrete state of the actuation system as the subset of actuators $I_N(t)$, that are healthy at time t and assume, without loss of generality, that $I_N(0) = I$. Assume that at time t_1 actuator

σ_1 becomes faulty, then the SR mechanism, by discarding actuator σ_1, results in the discrete state $I_N(t_1) = I \setminus \{\sigma_1\}$, which belongs to $F(I)$. Further faults will result in discrete states $I_N(t)$ moving to lower levels in the lattice, according to the dynamic dicrete state equation

$$I_N(t^+) = I_N(t^-) \setminus \{\Sigma_f(t)\},$$

where $I_N(t^-)$ is the discrete state before the fault, $\Sigma_f(t) \subseteq I_N(t^-)$ is the subset of actuators that become faulty at time t, and $I_N(t^+)$ is the discrete state after the system reconfiguration. Symmetrically, repair operations move the discrete state to higher levels, according to

$$I_N(t^+) = I_N(t^-) \cup \{\Sigma_r(t)\},$$

where $\Sigma_r(t) \subseteq I \setminus I_N(t^-)$ is the subset of actuators that have been repaired at time t (it can be checked that subsets of faulty or repaired actuators can be treated one by one, in an arbitrary order, resulting in the same post-fault or post-maintenance node in the lattice). Repair operations have an important impact on system reliability (by means of fault avoidance), but due to space limitation, they will not be considered any further here.

Critical actuator subsets. Assume that $I_N(t) \in 2^{I+}$. A subset of actuators $\Sigma \subset I_N(t)$ such that

$$I_N(t) \setminus \Sigma \in 2^{I+}$$

is not critical, in the sense that its loss can be tolerated. The **critical actuator subsets** associated with a discrete state $I_N(t)$ are all the minimal subsets that satisfy

$$C(I_N(t)) = \left\{\Sigma \subset I_N(t) : I_N(t) \setminus \Sigma \in 2^{I-}\right\}$$

Minimality is required in the definition because the loss of any superset of a critical actuator subset could obviously not be tolerated.

Evaluation of the fault tolerance level. The system is tolerant to actuator faults, when the SR strategy is used, as long as

$$I_N(t) \in 2^{I+}$$

Since the set 2^{I+} can be determined off-line, it is easy to check on-line, when actuators fail and are switched-off, if the state $I_N(t)$ satisfies this condition. Better, at any time t, it is possible to evaluate the "remaining fault tolerance" by means of the *redundancy degrees*:

$$\begin{aligned} k_{\min}[I_N(t)] &= \min\left\{k : \mathcal{F}^k(I_N(t)) \not\supseteq 2^{I+}\right\} \\ k_{\max}[I_N(t)] &= \min\left\{k : \mathcal{F}^k(I_N(t)) \cap 2^{I+} = \emptyset\right\} \end{aligned}$$

$k_{\min}[I_N(t)]$ is called the strong redundancy degree of $I_N(t)$. It follows from its definition that no matter which actuators are lost, as long as their number does not exceed $k_{\min}[I_N(t)] - 1$, the fault can be tolerated.

$k_{\max}\left[I_N(t)\right]$ is the weak redundancy degree of $[I_N(t)]$. It follows from its definition that the biggest set of actuators whose loss can be tolerated is of size $k_{\max}\left[I_N(t)\right]-1$. It obviously follows from their definition that for any $I_N(t)$

$$k_{\min}\left[I_N(t)\right] \leq k_{\max}\left[I_N(t)\right].$$

Example 7.4 *Redundancy degrees*

Figure 7.4 shows that $k_{\min}\left[a,b,c,d\right]=1$: the loss of one actuator brings the system to a Level 1 node, which is white for sure. The loss of two actuators does not guarantee this property. It is also seen that $k_{\max}\left[a,b,c,d\right]=2$: there exists a situation, where the loss of two actuators can be tolerated (there is one white node at level 2). □

7.4 Fault-tolerant model-matching design

7.4.1 Reconfiguration problem

The basic scheme of fault-tolerant control is depicted in Fig. 1.1 on p. 2. At the execution level, a feedback controller

$$\boldsymbol{u}(t)=\boldsymbol{k}(\boldsymbol{y}(t),\,\boldsymbol{y}_{\mathrm{ref}}(t))$$

is used to attenuate the disturbance $\boldsymbol{d}$ and to ensure command tracking for the command input $\boldsymbol{y}_{\mathrm{ref}}$. The control law $\boldsymbol{k}$ is designed so that the closed-loop system satisfies the given requirements for the faultless plant. Before a fault f occurs the supervision level shown in the figure is not active.

The adaptation of the controller to the faulty system is accomplished at the supervision level. This process results in new controller parameters and possibly in a new control configuration. If the sensors and actuators work differently as before but the faulty plant is still observable and controllable, the control configuration can remain as before but the controller parameters have to be adapted to the faulty system. This process is called fault accommodation.

However, if the sensor or actuator faults break the control loop, new sensors or actuators, respectively, have to be used. Then the controller has to be "reconfigured" in the sense that the whole process of selecting a suitable control configuration and of choosing appropriate controller parameters has to be repeated after the fault is present. It is not sufficient to change some controller parameters. Instead, the control problem has to be considered "from the scratch" by appropriately choosing

- the signal vector $\boldsymbol{y}$ to be controlled and the input vector $\boldsymbol{u}$ to be used,
- the control law $\boldsymbol{k}$ including the controller parameters,
- the set-point $\boldsymbol{y}_{\mathrm{ref}}$ of underlying control loops.

Control reconfiguration can be thought of as an "analytical repair" of the closed-loop system, where instead of repairing the plant the controller software is changed while exploiting the redundant measurement or control signals for satisfying the control specifications despite of the fault (Fig. 7.5).

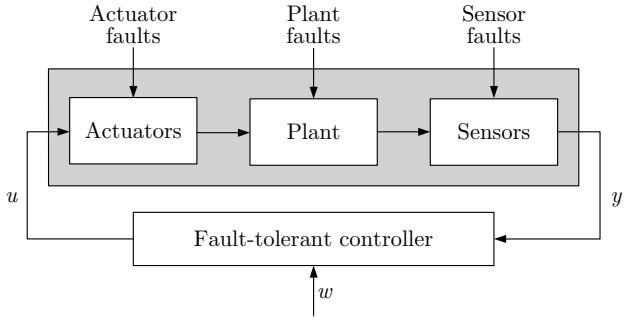


Fig. 7.5. Fault-tolerant controller

To solve the fault-tolerant control problem, it is assumed that a state-space model

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), f), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{7.26}$$

$$\boldsymbol{y}(t) = \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), f) \tag{7.27}$$

with state $\boldsymbol{x} \in \mathbb{R}^n$, input $\boldsymbol{u} \in \mathbb{R}^m$ and output $\boldsymbol{y} \in \mathbb{R}^r$ is available which also describes the dependence of the plant dynamics upon the faults $f \in \mathcal{F}$, where the set $\mathcal{F}$ includes all faults under consideration. Furthermore, it is assumed that a diagnostic algorithm has identified the current fault f.

According to these assumptions, the fault-tolerant control problem can be summarised as follows:

Problem 7.2 (Fault-tolerant control problem)

Given: *Model (7.26), (7.27) of the plant.*
Nominal controller $\boldsymbol{k}$.
Control specifications.
Fault f.

Find: *Control configuration and new control law* $\boldsymbol{k}_f$.

Note that in contrast to the usual controller design problem, also a nominal controller $\boldsymbol{k}$ is given. One of the important aspects of fault-tolerant control is to take advantage of the knowledge of the nominal controller $\boldsymbol{k}$.

7.4.2 Pseudo-inverse method

One of the earliest methods for the controller re-design is based on model-matching. As the nominal closed-loop system is known, the model of this system can be used as a description of the dynamical properties that the new controller should produce in connection with the faulty plant. That is, the closed-loop system should match the model of the nominal loop.

The idea of model-matching is depicted in Fig. 7.6. The nominal closed-loop system is composed of the linear nominal plant

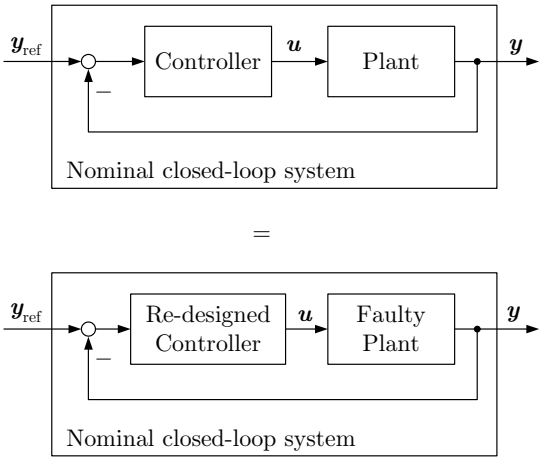


Fig. 7.6. Idea of the model-matching approach to control reconfiguration

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \tag{7.28}$$
$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) \tag{7.29}$$

and a state feedback controller $\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{x}(t)$ both of which yield the model of the closed-loop system

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K})\,\boldsymbol{x}(t)\\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t).\end{aligned}$$

If the controller does not use all the inputs u_i of the input vector $\boldsymbol{u}$, the matrix $\boldsymbol{K}$ has zero rows. When the fault f occurs, the faulty plant is given by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_f\boldsymbol{x}(t) + \boldsymbol{B}_f\boldsymbol{u}(t) \tag{7.30}$$
$$\boldsymbol{y}(t) = \boldsymbol{C}_f\boldsymbol{x}(t), \tag{7.31}$$

where the fault f has changed the system properties, which are now described by the matrices $\boldsymbol{A}_f$, $\boldsymbol{B}_f$ and $\boldsymbol{C}_f$. If the sets of available input or output signals have changed, the matrices $\boldsymbol{B}_f$ and $\boldsymbol{C}_f$ have vanishing columns or rows, respectively. A new state feedback controller

$$\boldsymbol{u}(t) = -\boldsymbol{K}_f\boldsymbol{x}(t)$$

should be found such that the closed-loop system

$$\begin{aligned}\dot{\boldsymbol{x}}(t) &= (\boldsymbol{A}_f - \boldsymbol{B}_f\boldsymbol{K}_f)\,\boldsymbol{x}(t)\\ \boldsymbol{y}(t) &= \boldsymbol{C}_f\boldsymbol{x}(t)\end{aligned}$$

behaves like the nominal loop. That is, the relation

$$\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K} = \boldsymbol{A}_f - \boldsymbol{B}_f\boldsymbol{K}_f \tag{7.32}$$

has to hold, which means that both closed-loop systems have similar dynamics.

Equation (7.32) cannot be satisfied unless $\boldsymbol{B}$ and $\boldsymbol{B}_f$ have the same image, (like in the case of a redundant actuator). Therefore, the new controller $\boldsymbol{K}_f$ is chosen so as to minimise the difference

$$\|(\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}) - (\boldsymbol{A}_f - \boldsymbol{B}_f\boldsymbol{K}_f)\|. \tag{7.33}$$

The solution to this problem is given by

$$\boldsymbol{K}_f = \boldsymbol{B}_f^+ \left(\boldsymbol{A}_f - \boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}\right) = \left(\boldsymbol{B}_f' \boldsymbol{B}_f\right)^{-1} \boldsymbol{B}_f' \left(\boldsymbol{A}_f - \boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}\right), \tag{7.34}$$

where $\boldsymbol{B}_f^+$ denotes the pseudoinverse of $\boldsymbol{B}_f$ given on the right-hand side of (7.34). Its use provides the reason for the name *pseudo-inverse method* of this approach.

The new controller (7.34) is adapted to the faulty system and minimises the difference (7.33) between the dynamical properties of the nominal loop and the closed-loop system with the faulty plant. However, it does not ensure the stability of the closed-loop system. Therefore, the stability of $\boldsymbol{A}_f - \boldsymbol{B}_f\boldsymbol{K}_f$ has to be tested separately. Extensions of this method ensure the stability without a separate test.

Fault accommodation and control reconfiguration. The method described so far is rather general. It includes both fault accommodation and control reconfiguration. Depending on the sensors and the actuators used, the controller is simply adapted to the new plant dynamics or it uses new sensors or actuators. In the latter case vanishing rows in the nominal controller $\boldsymbol{K}$ are replaced by nonzero elements, which means that new actuators are used and, hence, a new control configuration results.

7.4.3 Model-matching control for sensor failures

This section considers the case of complete sensor failures. If the i-th sensor fails, the output y_i is identical to zero. In the plant model the matrix $\boldsymbol{C}$ changes to $\boldsymbol{C}_f$, whose i-th row is zero, but the other matrices remain the same as in the nominal case. The corresponding reconfiguration problem will be investigated here for output feedback

$$\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{y}(t)\,,$$

for which the closed-loop system is described by

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}\boldsymbol{C})\,\boldsymbol{x}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t). \end{aligned}$$

For the faulty plant, the new controller

$$\boldsymbol{u}(t) = -\boldsymbol{K}_f\,\boldsymbol{y}_f(t)$$

should be found such that the closed loop

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}_f\,\boldsymbol{C}_f)\,\boldsymbol{x}(t) \\ \boldsymbol{y}_f(t) &= \boldsymbol{C}_f\,\boldsymbol{x}(t) \end{aligned}$$

has the same dynamics as the nominal loop.

The controller has to satisfy the simplified version of Eq. (7.32)

$$\boldsymbol{K}_f \boldsymbol{C}_f = \boldsymbol{K}\boldsymbol{C}. \tag{7.35}$$

To find an appropriate matrix $\boldsymbol{K}_f$ is possible only if the condition

$$\text{Kern}(\boldsymbol{C}_f) \subseteq \text{ Kern}(\boldsymbol{C}) \tag{7.36}$$

is satisfied, where Kern denotes the kernel[1] of a matrix. The condition means that the measurement information obtained by the full output vector $\boldsymbol{y}$ is the same as the information obtained by the remaining sensors through $\boldsymbol{y}_f$. The condition (7.36) can be written in an equivalent form as

$$\text{rank } \boldsymbol{C}_f = \text{rank} \begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{C}_f \end{pmatrix}.$$

Lemma 7.1 *In case of sensor failures, exact model-matching can be reached if the relation (7.36) holds. Then, the controller*

$$\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{P}\boldsymbol{y}(t) \tag{7.37}$$

solves the reconfiguration problem where

$$\boldsymbol{P} = \boldsymbol{C}\boldsymbol{C}_f^{+} = \boldsymbol{C}\boldsymbol{C}_f^{'}\left(\boldsymbol{C}_f\boldsymbol{C}_f^{\prime}\right)^{-1} \tag{7.38}$$

satisfies the relation

$$\boldsymbol{C} = \boldsymbol{P}\boldsymbol{C}_f. \tag{7.39}$$

The reconfigured controller $\boldsymbol{K}_f = \boldsymbol{K}\boldsymbol{P}$ produces a closed-loop system that has exactly the same properties as the faultless closed-loop system.

Situations where the requirement (7.36) is satisfied include the following:

- The fault has changed the sensitivity of the sensor, but the signal is not completely lost. Hence, $\boldsymbol{y}_f = a\,\boldsymbol{y}$ holds for some scalar a.
- A sensors is at fault which has at least one parallel redundant sensor. The matrix $\boldsymbol{P}$ switches the output to the redundant sensor.
- An analytic relation between the faulty output and several other output values exists, which can be reformulated by using the matrix $\boldsymbol{P}$.

The later two cases are only possible if $\boldsymbol{C}$ does not have full rank, which is likely in special applications only.

7.4.4 Model-matching control for actuator failures

In case of an actuator failure, the matrix $\boldsymbol{B}$ is replaced by the matrix $\boldsymbol{B}_f$ with zero column for the failing actuator. The output feedback

[1] The kernel of $\boldsymbol{C}$ is the set of vectors $\boldsymbol{x}$ for which $\boldsymbol{C}\boldsymbol{x}(t) = \boldsymbol{0}$ holds.

$$\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{y}(t)$$

which leads to the closed-loop system

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= (\boldsymbol{A} - \boldsymbol{B}_f\,\boldsymbol{K}\boldsymbol{C})\;\boldsymbol{x}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t). \end{aligned}$$

should be replaced by a new controller

$$\boldsymbol{u}(t) = -\boldsymbol{K}_f\,\boldsymbol{y}_f(t)$$

such that the closed loop

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= (\boldsymbol{A} - \boldsymbol{B}_f\,\boldsymbol{K}_f\,\boldsymbol{C})\,\boldsymbol{x}(t) \\ \boldsymbol{y}_f(t) &= \boldsymbol{C}\boldsymbol{x}(t) \end{aligned}$$

has the same dynamics as the nominal loop.

The controller has to satisfy the simplified version of Eq. (7.32)

$$\boldsymbol{B}_f\,\boldsymbol{K}_f = \boldsymbol{B}\boldsymbol{K}. \tag{7.40}$$

A solution $\boldsymbol{K}_f$ to this equation exists only if the condition

$$\text{Im}\;(\boldsymbol{B}_f) \supseteq \text{Im}(\boldsymbol{B}) \tag{7.41}$$

holds, where Im denotes the image[2] of a matrix. An equivalent formulation of the condition (7.41) is given by

$$\text{rank}\;\boldsymbol{B}_f = \text{rank}\;(\boldsymbol{B}\boldsymbol{B}_f)\,.$$

Lemma 7.2 *In case of actuator failures, exact model-matching can be reached if Eq. (7.41) holds. Then the reconfigured controller is given by*

$$\boldsymbol{u}(t) = -\boldsymbol{N}\boldsymbol{K}\boldsymbol{y}(t), \tag{7.42}$$

where

$$\boldsymbol{N} = \boldsymbol{B}_f^{+}\boldsymbol{B} = \left(\boldsymbol{B}_f^{'}\boldsymbol{B}_f\right)^{-1}\boldsymbol{B}_f^{'}\boldsymbol{B} \tag{7.43}$$

is a matrix satisfying the relation

$$\boldsymbol{B}_f\boldsymbol{N} = \boldsymbol{B}\,. \tag{7.44}$$

The new controller $\boldsymbol{K}_f = \boldsymbol{N}\boldsymbol{K}$ yields a closed-loop system with exactly the same properties as the nominal closed loop.

Example 7.5 *Model-matching for actuator failures*

This example demonstrates the model-matching approach for actuator failures and shows the main idea and a situation in which this approach fails.

Consider the tank system shown in Fig. 7.7 which has two input pipes. Obviously, for the level control, only one pipe is necessary as control input and the redundant input can be used in case of an actuator failure.

[2] The image of $\boldsymbol{C}$ is the set of vetors $\boldsymbol{y}$, for which a vector $\boldsymbol{x}$ exists such that $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$ holds.

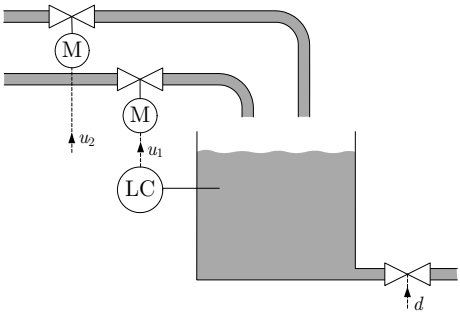


Fig. 7.7. Example demonstrating the model-matching reconfiguration strategy

Assume first, that the valve position is used as the control inputs. Then the system can be described by a state-space model (7.28), (7.29) where the matrix

$$\boldsymbol{B} = (\boldsymbol{b} \;\; k\boldsymbol{b})$$

has two linearly depending columns because the two input influence the system in the same way and the effects of the two actuators distinguish only with respect to some constant factor k. In the nominal system, the first control input is used:

$$u_1 = u_C = -\boldsymbol{K}\boldsymbol{y}$$

for some controller $\boldsymbol{K}$ and some output $\boldsymbol{y}$ of the tank system.

If the corresponding actuator fails, the controller should be switched to the second input, where

$$\boldsymbol{B}_f = (\boldsymbol{0} \;\; k\boldsymbol{b})$$

holds. The model-matching solutions yields the (21)-element of the matrix $\boldsymbol{N}$

$$\boldsymbol{N}_{21} = (k^2\boldsymbol{b}'\boldsymbol{b})^{-1}k\boldsymbol{b}'\boldsymbol{b} = \frac{1}{k},$$

which means that the output u_C of the nominal controller is transformed into the input

$$u_2 = \frac{1}{k}u_C$$

to the second actuator. This is a obvious solution: As the gain of the new actuator is k-times the gain of the old one, the old input u_C is multiplied by $\frac{1}{k}$. A perfect reconfiguration results.

Now change the situation by including the motors for the valves as shown in Fig. 7.7. As these motors have integral dynamics, two additional states have to be added to the state

$$\tilde{\boldsymbol{x}} = \begin{pmatrix} x_{a1} \\ x_{a2} \\ \boldsymbol{x} \end{pmatrix}$$

such that the model now reads as

$$\begin{aligned} \frac{d}{dt}\tilde{\boldsymbol{x}} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \boldsymbol{b} & k\boldsymbol{b} & \boldsymbol{A} \end{pmatrix}\tilde{\boldsymbol{x}} + \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \\ \boldsymbol{y} &= (\boldsymbol{O} \;\; \boldsymbol{O} \;\; \boldsymbol{C}) \end{aligned}$$

In principle, the same solution as before is possible. However, the model-matching approach yields for

$$\boldsymbol{B} = \begin{pmatrix} 1 \\ 0 \\ \mathbf{0} \end{pmatrix}$$

$$\boldsymbol{B}_f = \begin{pmatrix} 0 \\ 1 \\ \mathbf{0} \end{pmatrix}$$

the solution

$$\boldsymbol{N}_{21} = \left(\begin{pmatrix} 0 \\ 1 \\ \mathbf{0} \end{pmatrix}' \begin{pmatrix} 0 \\ 1 \\ \mathbf{0} \end{pmatrix} \right)^{-1} \begin{pmatrix} 0 \\ 1 \\ \mathbf{0} \end{pmatrix}' \begin{pmatrix} 1 \\ 0 \\ \mathbf{0} \end{pmatrix} = 0.$$

Hence, there is no control input at all. The model-matching approach fails.

The reason for this result lies in the fact that the model-matching idea tries to reproduce the effect $\boldsymbol{B}u$ of the nominal controller by the reconfigured controller $\boldsymbol{B}_f\boldsymbol{N}u$. This is impossible in this example, because the nominal controller has a direct effect only on the state variable x_{a1} an no effect at all on the state variable x_{a2} whereas the redundant input leads to the reverse situation. Hence, no choice of $\boldsymbol{N}$ can reproduce any of the effects of the nominal input. The failure of the model-matching approach lies in this idea and can be circumvented by extending the model-matching aim to the whole plant as described below. □

7.4.5 Markov parameter approach to control reconfiguration for actuator failures

The previous application of the model-matching approach using the pseudo-inverse on the input matrix shows that the reason of its failure lies in the concentration on the forcing action, point Ⓟ in Fig. 7.8. In the approach shown in this Section, the goal refers to the I/O-behaviour of the plant. By this formulation, analytical redundancies become amenable which are based on internal couplings via the system matrix on the one hand and the selection of relevant states via the output matrix on the other hand, see point Ⓠ in the figure. Such redundancies are hidden from a forcing action perspective.

The *Markov parameters*

$$\boldsymbol{G}_i = \boldsymbol{C}\boldsymbol{A}^{(i-1)}\boldsymbol{B}, \quad i = 1, \ldots, n \tag{7.45}$$

completely describe the input/output-behaviour of a linear system (7.28), (7.29) in terms of its transfer function

$$\boldsymbol{P}(s) = \sum_{i=0}^{\infty} \boldsymbol{G}_i s^{-i}. \tag{7.46}$$

The Markov parameter-based approach tries to recover the nominal plant Markov parameters after an actuator failure by using the static reconfiguration block

$$\boldsymbol{u}_c(t) = \boldsymbol{N}\boldsymbol{u}_f(t). \tag{7.47}$$

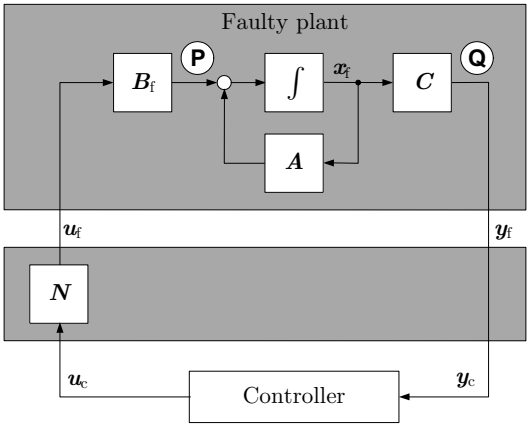


Fig. 7.8. Input/output-based reconfiguration after actuator failures

If the Markov parameters of a reconfigured plant match those of the nominal plant (7.28), (7.29) exactly, the dynamic I/O-behaviour is recovered exactly, which is both necessary and sufficient for successful static I/O-reconfiguration.

With the observability matrix

$$\boldsymbol{S}_O = \begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \vdots \\ \mathbf{CA}^{(n-1)} \end{pmatrix} \in \mathbb{R}^{n\cdot m\times n} \tag{7.48}$$

the design approach to Markov parameter recovery is

$$\boldsymbol{N} = \arg\min_{\boldsymbol{N}} \|\boldsymbol{S}_O\boldsymbol{B}_f\boldsymbol{N} - \boldsymbol{S}_O\boldsymbol{B}\| \tag{7.49}$$

with the solution

$$\boldsymbol{N} = (\boldsymbol{S}_O\boldsymbol{B}_f)^+\boldsymbol{S}_O\boldsymbol{B}. \tag{7.50}$$

If the condition

$$\mathrm{Im}\,(\boldsymbol{S}_O\boldsymbol{B}_f) \supseteq \mathrm{Im}\,(\boldsymbol{S}_O\boldsymbol{B}) \tag{7.51}$$

holds, then perfect I/O-reconfiguration results in the sense that all Markov parameters are exactly recovered. This condition is equivalent to

$$\mathrm{rank}\,(\boldsymbol{S}_O\boldsymbol{B}_f) = \mathrm{rank}\,(\boldsymbol{S}_O\boldsymbol{B}_f \;\; \boldsymbol{S}_O\boldsymbol{B}). \tag{7.52}$$

In the over-constrained case, an approximate solution is obtained in this way which matches the original Markov parameters as closely as possible.

Lemma 7.3 *In case of actuator failures, exact model-matching with respect to the input/output-behaviour can be reached if the condition (7.51) holds. Then the reconfigured controller is given by*

$$\boldsymbol{u}(t) = -\boldsymbol{N}\boldsymbol{K}\boldsymbol{y}(t), \tag{7.53}$$

where

$$\boldsymbol{N} = (\boldsymbol{S}_O\boldsymbol{B}_f)^+\boldsymbol{S}_O\boldsymbol{B} \tag{7.54}$$

is a matrix satisfying the relation

$$\boldsymbol{C}\boldsymbol{A}^{(i-1)}\boldsymbol{B}_f\boldsymbol{N} = \boldsymbol{C}\boldsymbol{A}^{(i-1)}\boldsymbol{B}, \quad i = 1, \ldots, n. \tag{7.55}$$

The new controller yields a closed-loop system with exactly the same input/output-behaviour as the nominal loop.

Remark 7.1 *Generality of the method*

The approach is valid in connection with any controller, since the plant I/O-response is recovered and the fault is hidden from the controller.

If the nominal loop was internally stable, this property is preserved under reconfiguration if condition (7.51) holds, as an analysis using the Kalman decomposition reveals. □

Example 7.5 (cont.) *Model-matching for actuator failures: Markov approach*

The example is now solved using the Markov approach. It is shown that the problems of the model-matching approach using the pseudo-inverse method are overcome.

The construction of the observability matrix (7.48) yields

$$\boldsymbol{S}_O\boldsymbol{B} = (\gamma \;\; k\gamma)\boldsymbol{b} \;\text{ with }\; \gamma = (0 \;\; \boldsymbol{C} \;\; \boldsymbol{C}\boldsymbol{A}...)', \tag{7.56}$$

whereas after the fault

$$\boldsymbol{S}_O\boldsymbol{B}_f = (\boldsymbol{0} \;\; k\gamma)\boldsymbol{b}. \tag{7.57}$$

Condition (7.51) is met and the admissible solution to the problem

$$\boldsymbol{S}_O\boldsymbol{B}_f\boldsymbol{N} = \boldsymbol{S}_O\boldsymbol{B} \tag{7.58}$$

is found using Eq. (7.50) as

$$\boldsymbol{N} = \begin{pmatrix} 0 & 0 \\ \frac{1}{k} & 1 \end{pmatrix}. \tag{7.59}$$

As expected, the control input meant for the first valve is redirected to the second valve with the correct gain adjustment. □

Example 7.6 *Markov approach applied to the two-tank example*

The plant consists of the two tanks T_1 and T_2 interconnected by valves u_L, u_H, where T_1 is filled via pump u_P as shown in Fig. 7.9. Valves are electromechanically driven with the motor states v_L, v_H. The controlled quantities are the levels h_1 and h_2. With the state $\boldsymbol{x} = (v_L, \; v_H, \; h_1, \; h_2)'$, the tank system is described by the linear model (7.28), (7.29) with

$$\boldsymbol{A} \;=\; 10^3 \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -3.2 & -3.4 & -7.1 & 3.6 \\ 3.2 & 3.4 & 7.1 & -18 \end{pmatrix}$$

$$\boldsymbol{B} = 10^3 \begin{pmatrix} 0 & 10^{-3} & 0 \\ 0 & 0 & 10^{-3} \\ 8.1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \qquad \boldsymbol{B}_{\mathrm{f}} = 10^3 \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 10^{-3} \\ 8.1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{C} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and controlled by two decentralised proportional controllers LC.

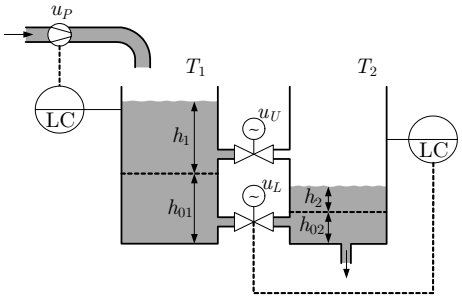


Fig. 7.9. Reconfiguration of a two-tank system

After a blocking lower valve at fault time t_{f}, which yields $u_L(t) = 0$ for $t \geq t_{\mathrm{f}}$, the plant is statically I/O-reconfigurable according to the condition (7.52). The reconfiguration (7.54) yields

$$\boldsymbol{N} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0.9167 & 1 \end{pmatrix}.$$

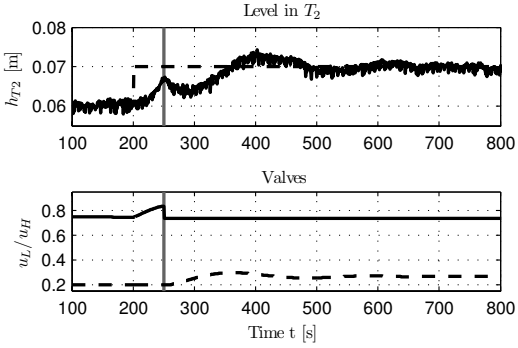


Fig. 7.10. Experimental results with the reconfigured tank system: After the failure of the lower valve (u_L, solid) the controller acts at the upper valve (u_H, dashed)

The behaviour of the successfully reconfigured plant with fault f occurring at $t_f = 250s$ is shown in Fig. 7.10. After the fault at $t_f = 250s$ and reconfiguration at $t = 260s$, the control action is redirected from the lower to the upper valve. This action appears logical, but it is not found by the well-known pseudo-inverse method. □

7.5 Control reconfiguration for actuator or sensor failures

7.5.1 The idea of virtual sensors and virtual actuators

Severe faults such as the complete failure of actuators or sensors break the control loops brought about by the nominal controllers. In order to hold the system in operation, it is necessary to use a different set of input or output signals to achieve the control task. Once the new control configuration is selected, new controller parameters have to be found. The goal of the reconfiguration is to stabilise the faulty process and to keep it operational with sufficient performance.

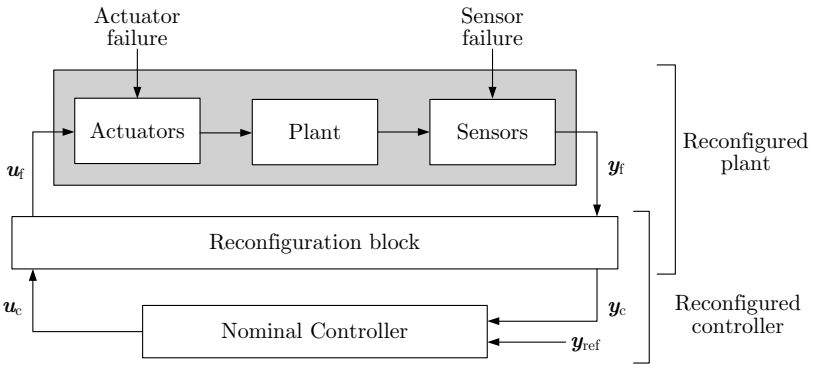


Fig. 7.11. Principle of control reconfiguration for actuator or sensor faults

Figure 7.11 shows the main idea of the methods explained in this section. Instead of adapting the controller to the faulty plant, a reconfiguration block is used to adapt the faulty plant to the nominal controller. The faulty plant together with the reconfiguration block should produce, for a given input $\boldsymbol{u}_c$, the same (or approximately the same) output $\boldsymbol{y}_c$ as the nominal plant. Hence, the controller "sees" the same plant as before and reacts in the same way as before.

This solution of the reconfiguration problem tries to apply a minimal change to the control loop. In particular, the nominal controller remains an unchanged block of the control loop. The rationale for keeping the controller as before is given by the fact that the existing control low includes valuable implicit knowledge about the process and the possible performance of the closed-loop system. This knowledge was acquired during the design cycle and is not represented in the process model.

For example, during the design it became obvious, which control objectives (like overshoot, band width, settling time) can be met with reasonable control effort and which not. The trade-off between the different control objectives is represented by the nominal controller.

In case of a sensor break-down, the reconfiguration block results from the application of a Luenberger observer to reconstruct the immeasurable output. It is called a "virtual sensor", because it reconstructs that element y_i of the output vector $\boldsymbol{y}_c$ from the other measured output signals that the faulty sensor does no longer measure. If an actuator becomes faulty, the reconfiguration block is obtained in a dual way. The reconfiguration block is called a "virtual actuator", because it acts like the faulty actuator but replaces the effect of this actuator by using the control input of the other actuators appropriately. The reconfigured controller, which is to be applied to the faulty plant, consists of the nominal controller and the reconfiguration block (Fig. 7.11).

The way to find appropriate reconfiguration blocks, which will be described in this section, uses an alternative interpretation of the reconfigured control loop: The faulty process and the reconfiguration block together are called the reconfigured plant, which is connected to the nominal controller. If the reconfigured plant behaves like the nominal plant, the loop consisting of the reconfigured plant and the controller behaves like the nominal closed-loop system. This is true for an arbitrary nominal controller.

Example 7.7 *Two-tank reconfiguration problem*

The reconfiguration is illustrated by the two coupled tanks depicted in Fig. 7.12.

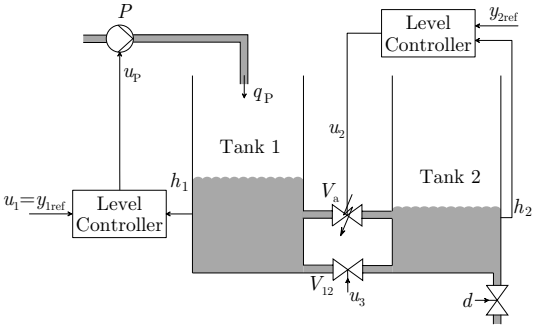


Fig. 7.12. Reconfiguration problem for the tank example

The main aim of the system is to store water at a certain level in the right tank for some consumer. During the nominal operation there exist two level controllers, with the set-points y_{1ref} and y_{2ref}. The right controller uses the upper valve, whose position is given by the input u_2. A redundant control input is provided by the lower valve with input signal u_3. In the nominal case, the valve V_{12} is closed. The right controller has to attenuate the disturbance d and to hold the tank level at a given value y_{2ref}. The control specifications include the stability, the set-point following requirement and the specification that the command step response should not have a large overshoot.

For the reconfiguration problem, three actuator faults are considered:

- Valve V_a is closed and blocked.
- Valve V_a is open and blocked.
- A level sensor is faulty.

In these cases, one of the two control loops does no longer work. The reconfiguration task consists in finding a new control structure by selecting appropriate actuators, new control laws and new set-points for the control loops such that the control aims described above are reached.

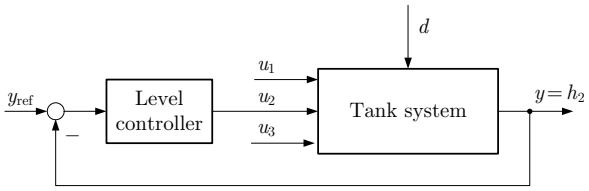


Fig. 7.13. Block diagram of the reconfiguration problem

Obviously, the reconfiguration task cannot be solved by simply changing the parameters of the given controllers, but a structural change of the control configuration is necessary:

- If Valve V_a is closed and blocked, the level controller of the right tank has to use the lower valve V_{12} as control input. In this case, the controller of the left tank can remain unchanged.
- If Valve V_a is open and blocked, in addition to the change of the level controller of the right tank as before, the set-point of the level controller of the left tank has to be set to a value which is lower then the position of Valve V_a. Another possibility is to use the set-point of the level controller of the left tank as control input of the level controller of the right tank.
- In case of the sensor fault, the missing sensor reading has to be reconstructed by means of the remaining output measurements.

All these solutions, which for this simple example seem to be obvious, have to be found automatically by a fault-tolerant control algorithm. □

7.5.2 Reconfiguration problem

Before explaining the reconfiguration method, the problem to be solved is formally stated. The model of the nominal process is given in state-space form:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{E}\boldsymbol{d}(t)\,, \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{7.60}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t). \tag{7.61}$$

The standard model is extended by the disturbance $\boldsymbol{d} \in \mathbb{R}^p$.

It is important that the process model includes all available input and output signals including those that are not used by the nominal controller. Unlike in the traditional design problem, $\boldsymbol{B}$ and $\boldsymbol{C}$ may not have full rank.

The nominal process is stabilised by a nominal controller with output $\boldsymbol{u}(t)$ and inputs $\boldsymbol{y}(t)$ and $\boldsymbol{y}_{\mathrm{ref}}(t)$. The reconfiguration method explained here can be applied without further assumptions on the controller, which may have an arbitrary dynamics and even be nonlinear. However, for the analysis of the resulting control loop a linear feedback controller

$$\boldsymbol{u}_c(t) = -\boldsymbol{K}\boldsymbol{y}_c(t) + \boldsymbol{V}\boldsymbol{y}_{\mathrm{ref}}(t) \tag{7.62}$$

is used.

Process and controller form the nominal control loop for $\boldsymbol{u}(t) = \boldsymbol{u}_c(t)$ and $\boldsymbol{y}(t) = \boldsymbol{y}_c(t)$:

$$\dot{\boldsymbol{x}}(t) = (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}\boldsymbol{C})\,\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{V}\boldsymbol{y}_{\mathrm{ref}}(t) + \boldsymbol{E}\boldsymbol{d}(t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{7.63}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t). \tag{7.64}$$

This control loop is assumed to be stable and to satisfy the performance requirements concerning set-point tracking and disturbance rejection.

Fault cases. In case of a loss of sensor i, the i-th row of the matrix $\boldsymbol{C}$ is changed into zeros and the new matrix is denoted by $\boldsymbol{C}_f$. If the j-th actuator fails, the j-th column of the matrix $\boldsymbol{B}$ is set to zero and the resulting matrix denoted by $\boldsymbol{B}_f$. In this way, the number of input signals, output signals and state variables is not changed in the model, though some of them may have lost their function. It is assumed that the faulty process is still controllable and observable. This implies that a stabilising controller exists. The input and output of the faulty plant are denoted by $\boldsymbol{u}_f$ or $\boldsymbol{y}_f$, respectively.

Reconfiguration task. The aim is to find a reconfigured controller that makes the closed-loop system satisfy the following conditions, which, depending on the control task, refer to the autonomous behaviour, reference tracking and disturbance rejection:

- **Strong reconfiguration goal:**
  The controller should make the reconfigured control loop behave in exactly the same way as the nominal control loop, i.e. the relation

  $$\boldsymbol{y}_f(t) = \boldsymbol{y}(t)$$

  should hold for any $\boldsymbol{d}(t)$, $\boldsymbol{y}_{\mathrm{ref}}(t)$ and $\boldsymbol{x}_0$.

It will be demonstrated that this strong goal is only feasible in very special cases. Therefore, a weaker goal is defined in terms of the dynamical and the static behaviour of the reconfigured loop.

- **Weak reconfiguration goal:**
  The weak goal consists of a static and a dynamical part. Considering the static behaviour, the output $\boldsymbol{y}_f$ of the reconfigured loop should have the same value as for the nominal system. This means that for constant values of $\boldsymbol{y}_{\mathrm{ref}}$ and $\boldsymbol{d}$, the relation

$$\boldsymbol{y}_f(t) \to \boldsymbol{y}(t) \text{ for } t \to \infty$$

should hold. The transient behaviour is determined by the poles and zeros of the system which should not differ significantly in the nominal and the reconfigured control loop. This requirement applies for the autonomous, the disturbance, and the command following behaviour of the reconfigured loop. Additional poles (and zeros) are allowed only if they are fast enough not to dominate the system behaviour.

7.5.3 Virtual sensor

This section describes a reconfiguration block that reconstructs a measurement y_i from the remaining sensor signals after the i-th sensor is no longer available. The main idea is to use an observer for the faulty system, which represents the main part of the reconfiguration block to be built. This block is called *virtual sensor* due to its function of replacing a broken sensor.

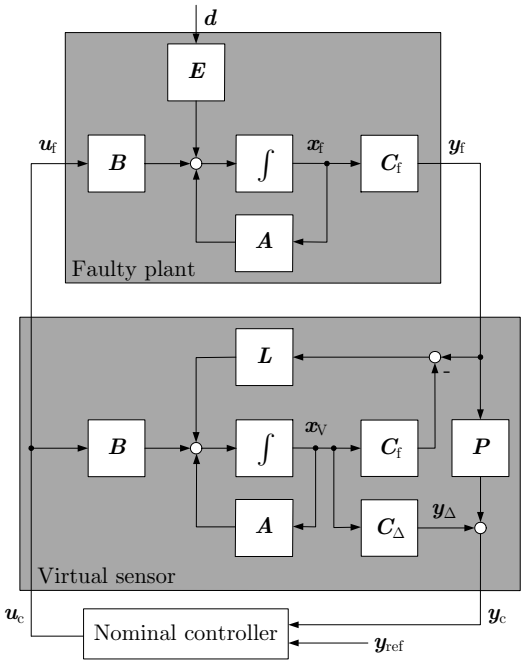


Fig. 7.14. Reconfiguration by using a virtual sensor

The system under consideration is described by the state-space model

$$\dot{\boldsymbol{x}}_f(t) = \boldsymbol{A}\boldsymbol{x}_f(t) + \boldsymbol{B}\boldsymbol{u}_f(t) + \boldsymbol{E}\boldsymbol{d}(t)\,, \quad \boldsymbol{x}_f(0) = \boldsymbol{x}_{f0} \tag{7.65}$$

$$\boldsymbol{y}_f(t) = \boldsymbol{C}_f\boldsymbol{x}_f(t), \tag{7.66}$$

where the sensor failure is reflected by the matrix $\boldsymbol{C}_f$. If the condition (7.36) is satisfied, the complete output vector $\boldsymbol{y}$ can be reconstructed from $\boldsymbol{y}_f$ and the reconfigured controller (7.37) can be used. This new control structure can be interpreted as consisting of a reconfiguration block

$$\begin{aligned} \boldsymbol{y}_c(t) &= \boldsymbol{P}\boldsymbol{y}_f(t) + \boldsymbol{y}_\Delta \\ \boldsymbol{u}_f(t) &= \boldsymbol{u}_c(t) \end{aligned}$$

and the nominal controller. That is, under the condition (7.36) the virtual sensor is a static reconfiguration block.

In the following, the general case is considered, where the condition (7.36) is violated. Then, the reconfiguration block includes a state observer and a direct feedthrough:

Definition 7.6 Virtual sensor
Consider the faulty plant (7.65), (7.66). The virtual sensor is defined as the system

$$\dot{\boldsymbol{x}}_V(t) = \boldsymbol{A}_V\boldsymbol{x}_V(t) + \boldsymbol{B}_V\boldsymbol{u}_c(t) + \boldsymbol{L}\boldsymbol{y}_f(t), \qquad \boldsymbol{x}_V(0) = \boldsymbol{x}_{V0} \tag{7.67}$$

$$\boldsymbol{u}_f(t) = \boldsymbol{u}_c(t) \tag{7.68}$$

$$\boldsymbol{y}_c(t) = \boldsymbol{C}_\Delta\boldsymbol{x}_V(t) + \boldsymbol{P}\boldsymbol{y}_f(t) \tag{7.69}$$

with the state $\boldsymbol{x}_V \in \mathbb{R}^n$ *and matrices*

$$\boldsymbol{A}_V = \boldsymbol{A} - \boldsymbol{L}\boldsymbol{C}_f \tag{7.70}$$

$$\boldsymbol{B}_V = \boldsymbol{B} \tag{7.71}$$

$$\boldsymbol{C}_V = \boldsymbol{C} - \boldsymbol{P}\boldsymbol{C}_f \tag{7.72}$$

$\boldsymbol{P}$ *and* $\boldsymbol{L}$ *denote matrices that can be freely chosen.*

The main part of the virtual sensor is the state observer with the state vector $\boldsymbol{x}_V(t)$. The complete output $\boldsymbol{y}_c(t)$ of the plant can be approximately determined: $\boldsymbol{y}_c(t) \approx \boldsymbol{C}\boldsymbol{x}_V(t)$. This observation result is improved by using the available sensor values and by observing only the difference between the nominal and the faulty output. In a generalised form, this approach is represented by Eq. (7.69) where the matrix $\boldsymbol{P}$ is a design parameter. For $\boldsymbol{P} = \boldsymbol{O}$ only observed values are used.

Model of the reconfigured plant. The plant together with the virtual sensor is described by Eqs. (7.65) – (7.72):

$$\begin{pmatrix} \dot{\boldsymbol{x}}_f(t) \\ \dot{\boldsymbol{x}}_V(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{O} \\ \boldsymbol{L}\boldsymbol{C}_f & \boldsymbol{A} - \boldsymbol{L}\boldsymbol{C}_f \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_V(t) \end{pmatrix} + \begin{pmatrix} \boldsymbol{B} \\ \boldsymbol{B} \end{pmatrix} \boldsymbol{u}_c(t) + \begin{pmatrix} \boldsymbol{E} \\ \boldsymbol{O} \end{pmatrix} \boldsymbol{d}(t) \tag{7.73}$$

$$\boldsymbol{y}_c(t) = \begin{pmatrix} \boldsymbol{P}\boldsymbol{C}_f & \boldsymbol{C}_V \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_V(t) \end{pmatrix} . \tag{7.74}$$

A state transformation is performed in order to introduce the observation error $\boldsymbol{x}_\Delta(t) = \boldsymbol{x}_V(t) - \boldsymbol{x}_f(t)$: Equations (7.73), (7.74) are equivalent to

$$\begin{aligned} \begin{pmatrix} \dot{\boldsymbol{x}}_f(t) \\ \dot{\boldsymbol{x}}_\Delta(t) \end{pmatrix} &= \begin{pmatrix} \boldsymbol{A} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{A} - \boldsymbol{L}\boldsymbol{C}_f \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_\Delta(t) \end{pmatrix} \\ &\quad + \begin{pmatrix} \boldsymbol{B} \\ \boldsymbol{O} \end{pmatrix} \boldsymbol{u}_c(t) + \begin{pmatrix} \boldsymbol{E} \\ -\boldsymbol{E} \end{pmatrix} \boldsymbol{d}(t) \end{aligned} \tag{7.75}$$

$$\boldsymbol{y}_c(t) = \begin{pmatrix} \boldsymbol{C} & \boldsymbol{C}_V \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_\Delta(t) \end{pmatrix} \tag{7.76}$$

$$\begin{pmatrix} \boldsymbol{x}_f(0) \\ \boldsymbol{x}_\Delta(0) \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_{f0} \\ \boldsymbol{x}_{V0} - \boldsymbol{x}_{f0} \end{pmatrix} .$$

Model of the reconfigured loop. For the analysis of the closed-loop behaviour the model of the reconfigured plant is combined with the linear feedback controller (7.62):

$$\begin{aligned} \begin{pmatrix} \dot{\boldsymbol{x}}_f(t) \\ \dot{\boldsymbol{x}}_\Delta(t) \end{pmatrix} &= \begin{pmatrix} \boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}\boldsymbol{C} & -\boldsymbol{B}\boldsymbol{K}\boldsymbol{C}_V \\ \boldsymbol{O} & \boldsymbol{A} - \boldsymbol{L}\boldsymbol{C}_f \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_\Delta(t) \end{pmatrix} + \\ &\quad + \begin{pmatrix} \boldsymbol{E} \\ -\boldsymbol{E} \end{pmatrix} \boldsymbol{d}(t) + \begin{pmatrix} \boldsymbol{B}\boldsymbol{V} \\ \boldsymbol{O} \end{pmatrix} \boldsymbol{y}_{\mathrm{ref}}(t) \end{aligned} \tag{7.77}$$

$$\boldsymbol{y}_f(t) = \begin{pmatrix} \boldsymbol{C}_f & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_\Delta(t) \end{pmatrix} . \tag{7.78}$$

The trajectory of this system depends on the initial state, the reference input $\boldsymbol{y}_{\mathrm{ref}}$ and the disturbance $\boldsymbol{d}$ (Fig. 7.15). As the system is linear, the behaviour can be analysed separately for these three excitations.

Autonomous behaviour. For $\boldsymbol{y}_{\mathrm{ref}}(t) = \boldsymbol{0}$ and $\boldsymbol{d}(t) = \boldsymbol{0}$ the system (7.77), (7.78) simplifies to

$$\begin{pmatrix} \dot{\boldsymbol{x}}_f(t) \\ \dot{\boldsymbol{x}}_\Delta(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{A} - \boldsymbol{B}\boldsymbol{K}\boldsymbol{C} & -\boldsymbol{B}\boldsymbol{K}\boldsymbol{C}_V \\ \boldsymbol{O} & \boldsymbol{A} - \boldsymbol{L}\boldsymbol{C}_f \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_\Delta(t) \end{pmatrix} \tag{7.79}$$

$$\boldsymbol{y}_f(t) = \begin{pmatrix} \boldsymbol{C}_f & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_\Delta(t) \end{pmatrix} \tag{7.80}$$

$$\begin{pmatrix} \boldsymbol{x}_f(0) \\ \boldsymbol{x}_\Delta(0) \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_{f0} \\ \boldsymbol{x}_{V0} - \boldsymbol{x}_{f0} \end{pmatrix} .$$

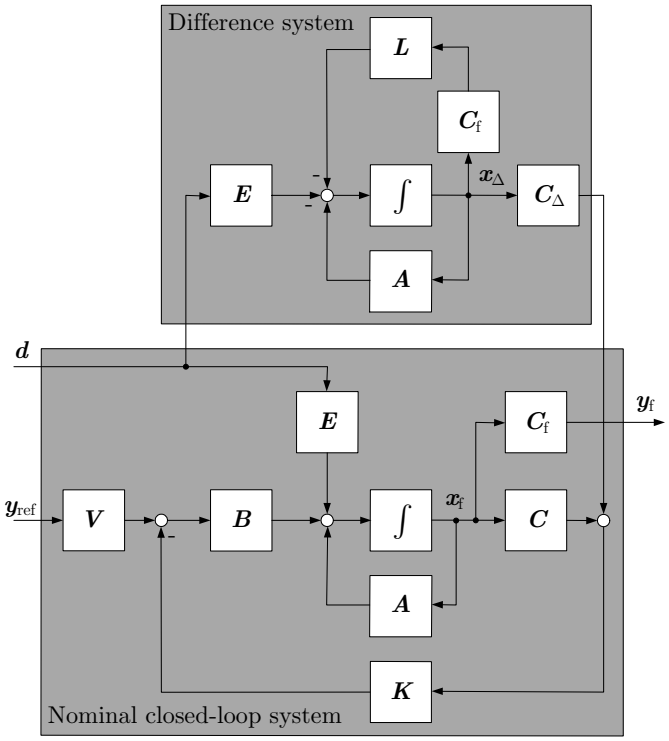


Fig. 7.15. Analysis of the closed-loop system with virtual sensor

The separation principle of state observers applies: The matrix $\boldsymbol{K}$ influences the behaviour of the process state $\boldsymbol{x}_f(t)$ through the submatrix $\boldsymbol{A} - \boldsymbol{BKC}$ (controller design), while $\boldsymbol{L}$ affects the behaviour of the observation error $\boldsymbol{x}_\Delta$ through the submatrix $\boldsymbol{LC}_f$ (observer design). There are cross-couplings in one direction only from $\boldsymbol{x}_\Delta(t)$ to $\boldsymbol{x}_f(t)$. The strength of the cross-coupling and the influence of $\boldsymbol{x}_\Delta(t)$ on the output can be reduced by a suitable choice of the matrix $\boldsymbol{P}$.

Theorem 7.1 Separation principle for the virtual sensor
The set σ of eigenvalues of the reconfigured closed-loop system (7.79), (7.80) consists of the set of eigenvalues of the nominal closed-loop system (7.63), (7.64) and the set of eigenvalues of the virtual sensor (7.67):

$$\sigma = \sigma\{\boldsymbol{A} - \boldsymbol{BKC}\} \cup \sigma\{\boldsymbol{A} - \boldsymbol{LC}_f\}.$$

The stability of the closed-loop is guaranteed if the nominal control loop is stable (depending on $\boldsymbol{K}$) and if the observer is stable (depending on $\boldsymbol{L}$). The second condition can be satisfied by an appropriate choice of $\boldsymbol{L}$ because the pair ($\boldsymbol{A}$,$\boldsymbol{C}_f$) is assumed to be observable. The equilibrium state is zero for both the faulty and the nominal system.

Tracking behaviour. For $\boldsymbol{x}_{f0} = \boldsymbol{x}_{V0} = \boldsymbol{0}$ and $\boldsymbol{d} = \boldsymbol{0}$, the system (7.77), (7.78) simplifies to

$$\begin{aligned} \dot{\boldsymbol{x}}_f(t) &= (\boldsymbol{A} - \boldsymbol{BKC})\,\boldsymbol{x}_f(t) + \boldsymbol{BV}\boldsymbol{y}_{\mathrm{ref}}(t), \qquad \boldsymbol{x}_f(0) = \boldsymbol{O} \\ \boldsymbol{y}_f(t) &= \boldsymbol{C}_f\boldsymbol{x}_f(t), \end{aligned}$$

which is identical to the behaviour of the nominal closed-loop system (7.63), (7.64). Hence, the reference tracking behaviour of the reconfigured control loop is identical to that of the nominal control loop.

Disturbance behaviour. For the disturbance behaviour it is assumed that the initial state and the reference input are zero. This leads to the following closed-loop system:

$$\begin{aligned} \begin{pmatrix} \dot{\boldsymbol{x}}_f(t) \\ \dot{\boldsymbol{x}}_\Delta(t) \end{pmatrix} &= \begin{pmatrix} \boldsymbol{A} - \boldsymbol{BKC} & -\boldsymbol{BKC}_V \\ \boldsymbol{O} & \boldsymbol{A} - \boldsymbol{LC}_f \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_\Delta(t) \end{pmatrix} \\ &\quad + \begin{pmatrix} \boldsymbol{E} \\ -\boldsymbol{E} \end{pmatrix} \boldsymbol{d}(t) \\ \boldsymbol{y}_f(t) &= \begin{pmatrix} \boldsymbol{C}_f & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_\Delta(t) \end{pmatrix} \\ \begin{pmatrix} \boldsymbol{x}_f(0) \\ \boldsymbol{x}_\Delta(0) \end{pmatrix} &= \begin{pmatrix} \boldsymbol{O} \\ \boldsymbol{O} \end{pmatrix}. \end{aligned}$$

It is obvious that the output $\boldsymbol{y}_f$ is different from the output $\boldsymbol{y}$ of the nominal control loop. The dynamical disturbance behaviour is much more complex because the number of states of the reconfigured process is $2n$ instead of n for the nominal process. The poles of the disturbance rejection behaviour depend on $\boldsymbol{K}$ and $\boldsymbol{L}$, while the zeros are affected by $\boldsymbol{P}$.

These results are summarised in the following theorem.

Theorem 7.2 (Virtual sensor)
For sensor faults, the virtual sensor (7.67) – (7.69) solves the reconfiguration problem such that the weak goal is reached provided that the faulty process is observable. The strong goal is reached for the reference tracking behaviour.

The analysis has shown how the virtual sensor works. The direct feedthrough $\boldsymbol{P}$ reconstructs or at least approximates the output $\boldsymbol{y}_c$ of the faultless plant from the remaining output $\boldsymbol{y}_f$. If the condition (7.36) is satisfied and $\boldsymbol{P}$ is chosen according to Eq. (7.38), the virtual sensor shrinks to a static reconfiguration block

$$\boldsymbol{y}_c(t) = \boldsymbol{C}\boldsymbol{C}_f^{'}\left(\boldsymbol{C}_f^{'}\boldsymbol{C}_f\right)^{-1}\boldsymbol{y}_f(t)$$

$$\boldsymbol{u}_f(t) \quad = \quad \boldsymbol{u}_c(t),$$

because $\boldsymbol{C}_V = \boldsymbol{O}$ results. This solution to the reconfiguration problem coincides with the solution obtained by the model-matching approach. The strong reconfiguration goal is satisfied.

If the condition (7.36) is not satisfied, the virtual sensor reconstructs the missing sensor information. Its state $\boldsymbol{x}_V(t)$ approximates the plant state $\boldsymbol{x}_f(t)$. The strong reconfiguration goal is satisfied only for the reference tracking behaviour. The disturbance behaviour of the reconfigured closed-loop system is typically slower compared with the nominal behaviour. The smaller the state $\boldsymbol{x}_\Delta(t)$ in the model of the disturbance behaviour is, the better approximates the reconfigured loop the nominal behaviour.

7.5.4 Virtual actuator

This section develops a solution to the reconfiguration problem for actuator failures. The notion of a virtual actuator is introduced as the dual system to the virtual sensor.

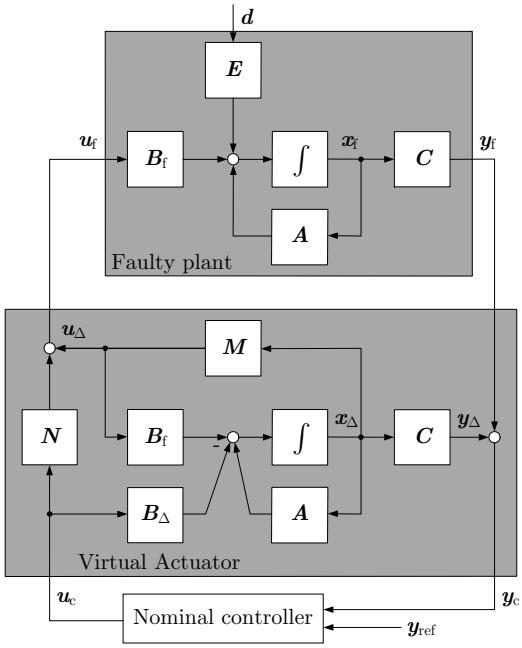


Fig. 7.16. Reconfiguration by means of a virtual actuator

The system under consideration is described by

$$\dot{\boldsymbol{x}}_f(t) \quad = \quad \boldsymbol{A}\boldsymbol{x}_f(t) + \boldsymbol{B}_f\,\boldsymbol{u}_f(t) + \boldsymbol{E}\boldsymbol{d}(t), \quad \boldsymbol{x}_f(0) = \boldsymbol{x}_{f0} \tag{7.81}$$

$$\boldsymbol{y}_f(t) \quad = \quad \boldsymbol{C}\boldsymbol{x}_f(t), \tag{7.82}$$

where zero columns in the matrix $\boldsymbol{B}_f$ reflect the failing actuators. If the condition (7.41) is satisfied, the static reconfiguration block

$$\begin{aligned} \boldsymbol{u}_f(t) &= \boldsymbol{N}\boldsymbol{u}_c(t) \\ \boldsymbol{y}_c(t) &= \boldsymbol{y}_f(t) \end{aligned}$$

can be used. In the following, the more general case is investigated, where this condition is not satisfied.

To explain the structure of the virtual actuator, the dual system of the reconfigured control loop for sensor faults shown in Fig. 7.14 is constructed. The result is shown in Fig. 7.16.

Definition 7.7 Virtual actuator
Consider the faulty plant (7.81), (7.82). The virtual actuator is defined as the system

$$\dot{\boldsymbol{x}}_\Delta(t) = \boldsymbol{A}_\Delta\boldsymbol{x}_\Delta(t) + \boldsymbol{B}_\Delta\boldsymbol{u}_c(t), \qquad \boldsymbol{x}_\Delta(0) = \boldsymbol{x}_{\Delta 0} \tag{7.83}$$

$$\boldsymbol{u}_f(t) = \boldsymbol{C}_\Delta\boldsymbol{x}_\Delta(t) + \boldsymbol{D}_\Delta\boldsymbol{u}_c(t) \tag{7.84}$$

$$\boldsymbol{y}_c(t) = \boldsymbol{C}\boldsymbol{x}_\Delta(t) + \boldsymbol{y}_f(t) \tag{7.85}$$

with the state $\boldsymbol{x}_\Delta \in \mathbb{R}^n$ *and matrices*

$$\boldsymbol{A}_\Delta = \boldsymbol{A} - \boldsymbol{B}_f\boldsymbol{M} \tag{7.86}$$

$$\boldsymbol{B}_\Delta = \boldsymbol{B} - \boldsymbol{B}_f\boldsymbol{N} \tag{7.87}$$

$$\boldsymbol{C}_\Delta = \boldsymbol{M} \tag{7.88}$$

$$\boldsymbol{D}_\Delta = \boldsymbol{N}. \tag{7.89}$$

$\boldsymbol{M}$ *and* $\boldsymbol{N}$ *denote matrices that can be freely chosen.*

Analysis of the reconfigured plant. The plant together with the virtual actuator leads to the following model of the reconfigured plant:

$$\begin{pmatrix} \dot{\boldsymbol{x}}_f(t) \\ \dot{\boldsymbol{x}}_\Delta(t) \end{pmatrix} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B}_f\boldsymbol{M} \\ \boldsymbol{O} & \boldsymbol{A} - \boldsymbol{B}_f\boldsymbol{M} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_\Delta(t) \end{pmatrix} + \begin{pmatrix} \boldsymbol{B}_f\boldsymbol{N} \\ \boldsymbol{B} - \boldsymbol{B}_f\boldsymbol{N} \end{pmatrix} \boldsymbol{u}_c(t) + \begin{pmatrix} \boldsymbol{E} \\ \boldsymbol{O} \end{pmatrix} \boldsymbol{d}(t) \tag{7.90}$$

$$\boldsymbol{y}_c(t) = \begin{pmatrix} \boldsymbol{C} & \boldsymbol{C} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \boldsymbol{x}_\Delta(t) \end{pmatrix}. \tag{7.91}$$

The introduction of the new state $\hat{\boldsymbol{x}}(t) = \boldsymbol{x}_f(t) + \boldsymbol{x}_\Delta(t)$ leads to the following equivalent model:

$$\begin{aligned}
\frac{d}{dt}\begin{pmatrix}\hat{\boldsymbol{x}}(t)\\ \boldsymbol{x}_\Delta(t)\end{pmatrix} &= \begin{pmatrix}\boldsymbol{A} & \boldsymbol{O}\\ \boldsymbol{O} & \boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{M}\end{pmatrix}\begin{pmatrix}\hat{\boldsymbol{x}}(t)\\ \boldsymbol{x}_\Delta(t)\end{pmatrix}\\
&\quad + \begin{pmatrix}\boldsymbol{B}\\ \boldsymbol{B}-\boldsymbol{B}_f\boldsymbol{N}\end{pmatrix}\boldsymbol{u}_c(t) + \begin{pmatrix}\boldsymbol{E}\\ \boldsymbol{O}\end{pmatrix}\boldsymbol{d}(t)\\
\boldsymbol{y}_c(t) &= \begin{pmatrix}\boldsymbol{C} & \boldsymbol{O}\end{pmatrix}\begin{pmatrix}\hat{\boldsymbol{x}}(t)\\ \boldsymbol{x}_\Delta(t)\end{pmatrix}\\
\begin{pmatrix}\hat{\boldsymbol{x}}(0)\\ \boldsymbol{x}_\Delta(0)\end{pmatrix} &= \begin{pmatrix}\boldsymbol{x}_0+\boldsymbol{x}_{\Delta 0}\\ \boldsymbol{x}_{\Delta 0}\end{pmatrix}.
\end{aligned}$$

Note that the state $\boldsymbol{x}_\Delta$ of the second subsystem is not observable by $\boldsymbol{y}_c$. Hence, this state does not influence the input-output behaviour of the reconfigured plant, whose model can be reduced to

$$\begin{aligned}
\dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t)+\boldsymbol{B}\boldsymbol{u}_c(t), \qquad \boldsymbol{x}(0)=\boldsymbol{x}_0+\boldsymbol{x}_{\Delta 0}\\
\boldsymbol{y}_c(t) &= \boldsymbol{C}\boldsymbol{x}(t).
\end{aligned}$$

This model is identical to the nominal plant provided that $\boldsymbol{x}_{\Delta 0}=\boldsymbol{0}$ holds.

Theorem 7.3 *The reconfigured plant (7.81), (7.89) has the same input-output behaviour as the nominal plant (7.63), (7.64) for arbitrary parameter matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ of the virtual actuator.*

Hence, the virtual actuator yields a reconfigured plant that satisfies the fault-hiding goal for arbitrary matrices $\boldsymbol{M}$ and $\boldsymbol{N}$.

Separation principle for the virtual actuator. The reconfigured closed-loop system consists of the reconfigured plant and the controller (7.62), both of which are considered for vanishing disturbance $\boldsymbol{d}$ and command input $\boldsymbol{y}_{\mathrm{ref}}$. If the transformed model is used, the reconfigured closed-loop system is described by

$$\begin{aligned}
\frac{d}{dt}\begin{pmatrix}\hat{\boldsymbol{x}}(t)\\ \boldsymbol{x}_\Delta(t)\end{pmatrix} &= \begin{pmatrix}\boldsymbol{A}-\boldsymbol{B}\boldsymbol{K}\boldsymbol{C} & \boldsymbol{O}\\ -\boldsymbol{B}_\Delta\boldsymbol{K}\boldsymbol{C} & \boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{M}\end{pmatrix}\begin{pmatrix}\hat{\boldsymbol{x}}(t)\\ \boldsymbol{x}_\Delta(t)\end{pmatrix}\\
\begin{pmatrix}\hat{\boldsymbol{x}}(0)\\ \boldsymbol{x}_\Delta(0)\end{pmatrix} &= \begin{pmatrix}\boldsymbol{x}_0+\boldsymbol{x}_{\Delta 0}\\ \boldsymbol{x}_{\Delta 0}\end{pmatrix}.
\end{aligned}$$

As the system matrix is a block diagonal matrix, the following result is obtained:

Theorem 7.4 Separation principle for the virtual actuator
The set σ of eigenvalues of the reconfigured closed-loop system (7.62), (7.81)–(7.89) consists of the set of eigenvalues of the nominal closed-loop system (7.62)–(7.64) and the set of eigenvalues of the virtual actuator (7.83):

$$\sigma = \sigma\{\boldsymbol{A}-\boldsymbol{B}\boldsymbol{K}\boldsymbol{C}\}\cup\sigma\{\boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{M}\}.$$

This theorem holds true for arbitrary matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ of the virtual actuator. Clearly, a corollary of this theorem is that the matrix $\boldsymbol{M}$ has to be chosen so that the matrix $\boldsymbol{A} - \boldsymbol{B}_f\boldsymbol{M}$ has eigenvalues with negative real parts in order to ensure the stability of the reconfigured closed-loop system.

Corollary 7.1 *The stability of the reconfigured closed-loop system can be ensured by appropriately choosing the matrix $\boldsymbol{M}$ of the virtual actuator if and only if the pair $(\boldsymbol{A}, \boldsymbol{B}_f)$ is stabilisable.*

This corollary shows that the stabilisation goal can be satisfied by using the generalised virtual actuator as long as the faults plant is stabilisable.

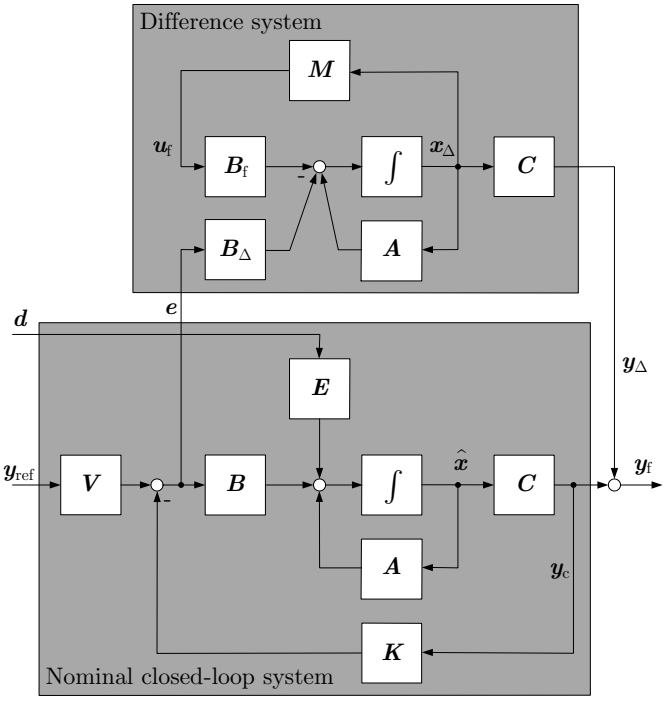


Fig. 7.17. Transformed closed-loop system showing the separation principle

I/O-behaviour of the reconfigured closed-loop system. The following investigates the input-output behaviour of the reconfigured closed-loop system and derives guidelines for choosing the parameter matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ of the virtual actuator. If the models of the faulty plant (7.81), (7.82) is combined with the virtual actuator (7.83) – (7.85) and the controller (7.62), the following model is obtained after the state $\hat{\boldsymbol{x}}$ has been introduced as before:

$$\frac{d}{dt}\begin{pmatrix}\hat{\boldsymbol{x}}(t)\\ \boldsymbol{x}_{\Delta}(t)\end{pmatrix} = \begin{pmatrix}\boldsymbol{A}-\boldsymbol{BKC} & \boldsymbol{O}\\ -\boldsymbol{B}_{\Delta}\boldsymbol{KC} & \boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{M}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}(t)\\ \boldsymbol{x}_{\Delta}(t)\end{pmatrix} + \begin{pmatrix}\boldsymbol{BV}\\ \boldsymbol{B}_{\Delta}\boldsymbol{V}\end{pmatrix}\boldsymbol{y}_{\mathrm{ref}}(t) + \begin{pmatrix}\boldsymbol{E}\\ \boldsymbol{O}\end{pmatrix}\boldsymbol{d}(t) \tag{7.92}$$

$$\begin{pmatrix}\hat{\boldsymbol{x}}(0)\\ \boldsymbol{x}_{\Delta}(0)\end{pmatrix} = \begin{pmatrix}\boldsymbol{x}_0+\boldsymbol{x}_{\Delta 0}\\ \boldsymbol{x}_{\Delta 0}\end{pmatrix}$$

$$\boldsymbol{y}_c(t) = (\boldsymbol{C}\quad \boldsymbol{O})\begin{pmatrix}\hat{\boldsymbol{x}}(t)\\ \boldsymbol{x}_{\Delta}(t)\end{pmatrix} \tag{7.93}$$

$$\boldsymbol{y}_f(t) = (\boldsymbol{C}\quad -\boldsymbol{C})\begin{pmatrix}\hat{\boldsymbol{x}}(t)\\ \boldsymbol{x}_{\Delta}(t)\end{pmatrix}. \tag{7.94}$$

The block diagram that illustrates this model is shown in Fig. 7.17. The lower block represents the nominal closed-loop system. The control error $\boldsymbol{e} = \boldsymbol{V}\boldsymbol{y}_{\mathrm{ref}} - \boldsymbol{y}_c$ is fed into the "difference system"

$$\dot{\boldsymbol{x}}_{\Delta}(t) = (\boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{M})\boldsymbol{x}_{\Delta}(t) + \boldsymbol{B}_{\Delta}\boldsymbol{e}(t), \qquad \boldsymbol{x}_{\Delta}(0) = \boldsymbol{x}_{\Delta 0} \tag{7.95}$$

$$\boldsymbol{y}_{\Delta}(t) = \boldsymbol{C}\boldsymbol{x}_{\Delta}(t), \tag{7.96}$$

whose name results from its output $\boldsymbol{y}_{\Delta}$, which is the difference between the output $\boldsymbol{y}_c$ of the nominal closed-loop system and the output $\boldsymbol{y}_f$ of the reconfigured closed-loop system. Hence, $\boldsymbol{y}_{\Delta}$ shows how the reconfigured closed-loop system differs from the nominal loop.

This model yields two corollaries:

- The input-output behaviour with respect to the disturbance input $\boldsymbol{d}$ or the command input $\boldsymbol{y}_{\mathrm{ref}}$, respectively, and to the output $\boldsymbol{y}_c$ is identical to the corresponding input-output behaviour of the nominal closed-loop system.
- The input-output behaviour with respect to the disturbance input $\boldsymbol{d}$ or the command input $\boldsymbol{y}_{\mathrm{ref}}$, respectively, and to the output $\boldsymbol{y}_f$ differs from that of the nominal closed-loop system due to the influence of the difference system (7.95), (7.96).

To summarise these results, the virtual actuator presents a successful reconfiguration in case of actuator failures. It creates a stable control loop with n placeable additional poles. However, it does not restore the original equilibrium unless the equilibrium is zero.

Theorem 7.5 (Virtual actuator)
For actuator failures, the virtual actuator (7.83) – (7.89) is a solution to the reconfiguration problem such that the weak goal is reached provided that the faulty process is controllable.

This following concerns the question how to choose the matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ of the virtual actuator in order to get a small difference $\boldsymbol{y}_\Delta$ beween the behaviour of the nominal and the reconfigured closed-loop system.

Complete reconfiguration. As Fig. 7.17 and Eqs. (7.95), (7.96) show, a complete reconfiguration is possible if the matrix $\boldsymbol{N}$ can be chosen such that the matrix $\boldsymbol{B}_\Delta$ vanishes.

Corollary 7.2 *If the matrix $\boldsymbol{N}$ can be chosen such that*

$$\boldsymbol{B}_\Delta = \boldsymbol{B} - \boldsymbol{B}_f\boldsymbol{N} = \boldsymbol{O} \tag{7.97}$$

holds, the input-output behaviour of the reconfigured closed-loop system is identical to that of the nominal control loop for both the disturbance input $\mathbf{d}$ and the command input $\boldsymbol{y}_{\mathrm{ref}}$. Furthermore, if

$$\boldsymbol{x}_\Delta(0) = \mathbf{0} \tag{7.98}$$

holds, the reconfigured loop has the same free motion as the nominal loop.

The condition (7.97) can be satisfied for an arbitrary controller (7.62) if and only if the relation (7.41) holds. Then the virtual actuator (7.83), (7.85) reduces to the static reconfiguration block

$$\boldsymbol{u}_f(t) = (\boldsymbol{B}_f^{'}\boldsymbol{B}_f)^{-1}\boldsymbol{B}_f^{'}\boldsymbol{B}\boldsymbol{u}_c(t) \tag{7.99}$$

$$\boldsymbol{y}_c(t) = \boldsymbol{y}_f(t), \tag{7.100}$$

which is identical to the reconfiguration solution described in Section 7.4.4.

If the condition (7.41) is violated, this static reconfiguration block does not solve the reconfiguration problem, $\boldsymbol{B}_\Delta \neq \boldsymbol{O}$ holds and the dynamical part of the virtual actuator becomes active.

Design of the virtual actuator by disturbance decoupling methods. If the transfer function matrix of the difference system vanishes

$$\boldsymbol{G}(s) = \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A} + \boldsymbol{B}_f\boldsymbol{M})^{-1}(\boldsymbol{B} - \boldsymbol{B}_f\boldsymbol{N}) = \boldsymbol{O}, \tag{7.101}$$

the reconfiguration is complete as well. Then the difference model (7.95), (7.96), which can be equivalently written as

$$\dot{\boldsymbol{x}}_\Delta(t) = \boldsymbol{A}\boldsymbol{x}_\Delta(t) + \boldsymbol{B}\boldsymbol{u}_c(t) + \boldsymbol{B}_f\boldsymbol{u}_f(t), \qquad \boldsymbol{x}_\Delta(0) = \boldsymbol{x}_{\Delta 0} \tag{7.102}$$

$$\boldsymbol{u}_\Delta(t) = \boldsymbol{M}\boldsymbol{x}_\Delta(t) + \boldsymbol{N}\boldsymbol{u}_c(t) + \boldsymbol{Q}\tilde{\boldsymbol{u}}(t) \tag{7.103}$$

has a vanishing output. To select the matrices $\boldsymbol{N}$ and $\boldsymbol{M}$ such that the condition (7.101) holds is a disturbance decoupling problem for known disturbance $\boldsymbol{u}_c$. It has been shown in [240] that the solution to this problem yields a complete reconfiguration. This solution exist, however, only under restrictive conditions.

Restoration of the static behaviour. The static behaviour is completely reconstructed if the gain of the difference system vanishes:

$$\boldsymbol{G}(0) = -\boldsymbol{C}(\boldsymbol{A} - \boldsymbol{B}_f\boldsymbol{M})^{-1}(\boldsymbol{B} - \boldsymbol{B}_f\boldsymbol{N}) = \boldsymbol{O} \tag{7.104}$$

Approximate solution. The generalised virtual actuator has the property that the effect of the virtual actuator "disappears" if the matrix $\boldsymbol{B}_\Delta$ can be made very small by choosing the matrix $\boldsymbol{N}$ appropriately.

Corollary 7.3 *For $\|\boldsymbol{B}_\Delta\| \to 0$, the behaviour of the reconfigured closed-loop system approaches that of the nominal loop:*

$$\|\boldsymbol{y}_c(t) - \boldsymbol{y}_f(t)\| \to 0.$$

Hence, if $\|\boldsymbol{B}_\Delta\|$ is sufficiently small it is reasonable to use the static reconfiguration block only.

Example 7.8 *Reconfiguration of the two-tank system*

To illustrate the reconfiguration by means of the virtual actuator, the problem posed in Example 7.7 is considered. The tank system is described by the nonlinear state-space model

$$\begin{aligned}
\dot{h}_1(t) &= \frac{Q_{1max}}{A_1}(-k_I x_r(t) - k_P(h_1(t) - u_1(t))) \\
&\quad - \frac{Q_{1max}}{S}\sqrt{2g(h_1(t) - h_v)}u_2(t) - \frac{Q_{1max}}{S}\sqrt{2gh_1(t)}u_3(t) \\
\dot{x}_r(t) &= h_1(t) - u_1(t) \\
\dot{h}_2(t) &= \frac{1}{A_2}\left(S\sqrt{2g(h_1(t) - h_v)}u_2(t) + S\sqrt{2gh_1(t)}u_3(t) - S\sqrt{2gh_2(t)}d(t)\right) \\
y_c(t) &= h_2(t)
\end{aligned}$$

that includes the controller of the left tank, which is a PI controller

$$\begin{aligned}
\dot{x}_r(t) &= h_1(t) - u_1(t) \\
\tilde{u}_1(t) &= -k_I x_r(t) - k_P(h_1(t) - u_1(t)).
\end{aligned}$$

This model uses the following parameters:

Symbol	Physical meaning
A_1, A_2	Cross section areas of the two tanks
Q_{1max}	Maximum flow through the pump
h_v	Height of the upper pipe above the tank bottom
S	Constant of the valves
g	gravity constant
k_I, k_P	Controller parameters

After the linearisation of the model around the operation point described by $\bar{h}_1$, $\bar{h_2}$, $\bar{u}_1$, $\bar{u_2}$, $\bar{u_3}$, the following model (7.60), (7.61) with

$$\begin{aligned}
\boldsymbol{A} &= \begin{pmatrix} -0.0478 & -0.0004 & 0 \\ 1.0000 & 0 & 0 \\ 0.0058 & 0 & -0.0058 \end{pmatrix} \\
\boldsymbol{B} &= \begin{pmatrix} 0.0406 & -0.0058 & -0.0092 \\ -1.0000 & 0 & 0 \\ 0 & 0.0046 & 0.0073 \end{pmatrix}
\end{aligned}$$

$$\boldsymbol{C} = (0 \quad 0 \quad 1)$$
$$\boldsymbol{E} = \begin{pmatrix} 0 \\ 0 \\ -0.0454 \end{pmatrix}$$

is obtained. Is is assumed that the upper valve fails and is, therefore, completely closed and no longer used as actuator of the right level controller. Then the second column in the matrix $\boldsymbol{B}$ has to be set to zero to obtain the matrix $\boldsymbol{B}_f$:

$$\boldsymbol{B} = \begin{pmatrix} 0.0406 & 0 & -0.0092 \\ -1.0000 & 0 & 0 \\ 0 & 0 & 0.0073 \end{pmatrix}.$$

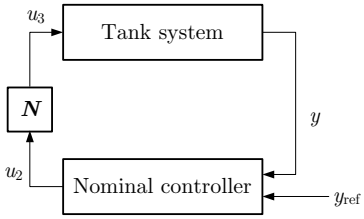


Fig. 7.18. Static reconfiguration of the tank system

Static reconfiguration. A complete reconfiguration of the controller is possible, because the condition (7.41) is satisfied due to the lower valve, which represents a redundant control input with similar effects on the tank system as the upper valve. In fact, the last column of $\boldsymbol{B}$ is linearly dependent upon the second column:

$$0.6325 \begin{pmatrix} -0.0092 \\ 0 \\ 0.0073 \end{pmatrix} = \begin{pmatrix} -0.0058 \\ 0 \\ 0.0046 \end{pmatrix}.$$

Hence, the reconfiguration is possible with a static reconfiguration block (7.99)

$$u_f(t) = \begin{pmatrix} 0 & 0 & -2.7039 \\ 0 & 0 & 0.6325 \\ 0 & 0 & 1 \end{pmatrix} u_c(t)$$

(cf. Fig. 7.18).

Figure 7.19 shows the reference tracking behaviour of the right tank for changing level set-point. The right tank has the same behaviour with the reconfigured controller as in the nominal case. In the lower subplot the control input u_3 used by the reconfigured controller is compared to the input u_2 of the nominal controller, which is shown by the dashed lines. Cleary, the new input has to be smaller than the nominal one, because the lower valve between the tanks has a higher effectiveness than the upper one, which can be seen by comparing the corresponding columns in the matrix $\boldsymbol{B}$.

Reconfiguration by means of the virtual actuator. If the lower valve is not available for the reconfiguration, the right controller has only the input u_1, which is the command signal

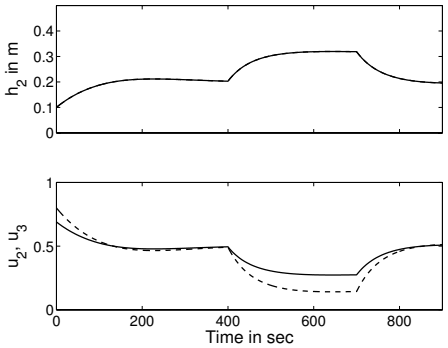


Fig. 7.19. Behaviour of the reconfigured closed-loop system where the reconfigured controller uses the input u_3

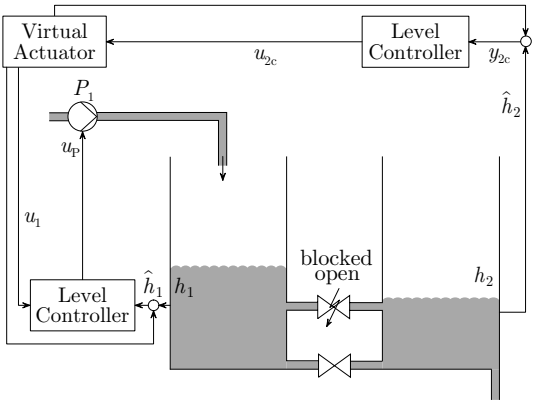


Fig. 7.20. Reconfigured system with virtual actuator

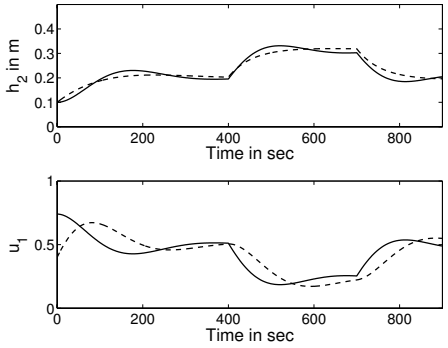


Fig. 7.21. Behaviour of the reconfigured closed-loop system where the reconfigured controller uses the input u_1

of the left controller, as its disposal. With the third columns deleted, the matrices $\boldsymbol{B}$ and $\boldsymbol{B}_f$ do no longer satisfy the condition (7.41). Hence, a dynamical reconfiguration block has to be used. The matrix $\boldsymbol{N}$ of the virtual actuator is chosen according to Eq. (7.43):

$$\boldsymbol{N} = \begin{pmatrix} 1 & -0.0002 & .0.0004 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

The nominal closed-loop system has the eigenvalues -0.0427, $-0.0124 \pm 0.0058i$. Therefore, the matrix $\boldsymbol{M}$ of the virtual actuator is chosen so as to place the eigenvalues of the matrix $\boldsymbol{A} - \boldsymbol{B}_f\boldsymbol{M}$ to the left of these eigenvalues, namely at -0.05, -0.06 and -0.07. As $\boldsymbol{M}$ should use only the first input, its nonzero elements are restricted to the first row:

$$\boldsymbol{M} = \begin{pmatrix} -0.9968 & -0.0048 & -0.0002 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

In summary, the virtual actuator (7.83), (7.84) results:

$$\begin{aligned} \dot{\boldsymbol{x}}_{\Delta}(t) = &\begin{pmatrix} -0.0074 & -0.0002 & 0 \\ 0.0032 & -0.0048 & -0.0002 \\ 0.0058 & 0 & -0.0058 \end{pmatrix} \boldsymbol{x}_{\Delta}(t) \; + \\ &+ \begin{pmatrix} 0 & -0.0058 & -0.0091 \\ 0 & -0.0002 & -0.0004 \\ 0 & 0.0046 & 0.0073 \end{pmatrix} u_{2c}(t) \\ u_1(t) = &(-0.9968 \quad -0.0048 \quad -0.0002)\boldsymbol{x}_{\Delta}(t) \; + \\ &+(1 \quad -0.0002 \quad 0.0004)\boldsymbol{y}_c(t), \end{aligned}$$

where $u_{2c}(t)$ is the control input generated by the nominal level controller of the right tank. This signal is used now as an input of the virtual actuator (cf. Fig. 7.20).

Figure 7.21 shows the disturbance behaviour of the tank system after the controller has been extended by a virtual actuator that uses the input u_1. The response is slower than the nominal response, which is drawn by dashed lines to make a comparison possible. The slower response results from the fact that the controller of the right tank uses now the command input of the controller of the left tank as control input. □

7.5.5 Duality between virtual sensors and virtual actuators

The comparison of the reconfiguration blocks developed in the preceding sections clearly shows the duality of the approaches for sensor and actuator failures. The variables correspond to each other in the following way:

Table 7.2 Duality of the system variables

Virtual sensor	$\boldsymbol{A}$	$\boldsymbol{B}$	$\boldsymbol{C}$	$\boldsymbol{K}$	$\boldsymbol{L}$	$\boldsymbol{P}$	$\hat{\boldsymbol{x}}$	$\boldsymbol{u}$	$\boldsymbol{y}$
Virtual actuator	$\boldsymbol{A}'$	$\boldsymbol{C}'$	$\boldsymbol{B}'$	$\boldsymbol{K}'$	$\boldsymbol{M}'$	$\boldsymbol{N}'$	$\tilde{\boldsymbol{x}}$	$\boldsymbol{y}$	$\boldsymbol{u}$

Note that the duality involves more than just swapping input and output and transposing the matrices. It requires that the directions of the signals be reversed. Summation points become signal knots and vice versa. The diagrams also require mirroring to preserve the clockwise signal direction of the control loop.

A short mathematical demonstration of the duality is given here. If the system (7.73) is transposed, the input and output matrices are exchanged, and all system matrices are transposed, the following model results:

$$\begin{aligned}\begin{pmatrix} \dot{\boldsymbol{x}}_f(t) \\ \dot{\hat{\boldsymbol{x}}}(t) \end{pmatrix} &= \begin{pmatrix} \boldsymbol{A}' & \boldsymbol{C}_f'\boldsymbol{L}' \\ \boldsymbol{O} & \boldsymbol{A}' - \boldsymbol{C}_f'\boldsymbol{L}' \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \hat{\boldsymbol{x}}(t) \end{pmatrix} \\ &\quad + \begin{pmatrix} \boldsymbol{C}_f'\boldsymbol{P}' \\ \boldsymbol{C}' - \boldsymbol{C}_f'\boldsymbol{P}' \end{pmatrix} \boldsymbol{u}_f(t) \\ \boldsymbol{y}_c(t) &= \begin{pmatrix} \boldsymbol{B} & \boldsymbol{B} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_f(t) \\ \hat{\boldsymbol{x}}(t) \end{pmatrix} .\end{aligned}$$

Apart from the different variable names according to Table 7.2, the result is identical to (7.90) – (7.91). The duality holds for most properties, but not for the reference tracking. The reason is that $\boldsymbol{y}_{\mathrm{ref}}$ and $\boldsymbol{y}$ do not have symmetric positions in the system.

7.6 Fault-tolerant $\mathcal{H}_\infty$ design

This section introduces fault-tolerant control strategies that can be applied in a general fault case. It starts with a characterisation of all controllers that stabilise a linear system and maybe also satisfied norms condition like e.g. $\mathcal{H}_2$ or $\mathcal{H}_\infty$ norm conditions. This characterisation makes it possible to evaluate the severity of the fault with respect to the control aims and to find methods for re-design the controller automatically.

The complete description of all stabilising controllers is given by the Youla-Kucera or Q-parametrisation. The Youla parametrisation was originally defined by using co-prime factorisation. However, it is simple to give an equivalent description of the Youla-Kucera parametrisation as a state-space formulation. In the state-space formulation, the parametrisation turns out to be an observer-based controller.

One of the facilities by using this parametrisation is that the closed-loop transfer function turns out to be affine in the controller parameters. This affine structure is very useful in connection with design of controllers using optimisation methods. Therefore the method also fits well to solve the control problem arising when we wish to make a re-design for a controller when a system is in a faulty state.

The salient feature offered by the Youla-Kucera parametrisation is that it offers an elegant and very fast solution to the re-design problem for some classes of faults that leave the system stable with the existing controller but make it unable to meet the required performance.

7.6.1 System description

Consider

$$\begin{aligned} \dot{x} &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}_1\boldsymbol{w} + \boldsymbol{B}_2\boldsymbol{u}(t) \\ \boldsymbol{z} &= \boldsymbol{C}_1\boldsymbol{x}(t) + \boldsymbol{D}_{11}\boldsymbol{w} + \boldsymbol{D}_{12}\boldsymbol{u}(t) \\ \boldsymbol{y} &= \boldsymbol{C}_2\boldsymbol{x}(t) + \boldsymbol{D}_{21}\boldsymbol{w} + \boldsymbol{D}_{22}\boldsymbol{u}. \end{aligned} \tag{7.105}$$

For brevity, it is common to denote this system by the shorter notation

$$\boldsymbol{G}(s) = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{C}_1 & \boldsymbol{C}_2 \\ \boldsymbol{B}_1 & \boldsymbol{D}_{11} & \boldsymbol{D}_{21} \\ \boldsymbol{B}_2 & \boldsymbol{D}_{12} & \boldsymbol{D}_{22} \end{pmatrix},$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the state, $\boldsymbol{u} \in \mathbb{R}^r$, is the control input, $\boldsymbol{w} \in \mathbb{R}^k$ is the external input or disturbance, $\boldsymbol{z} \in \mathbb{R}^l$ is the controlled output and $\boldsymbol{y} \in \mathbb{R}^m$ is the measurement output. It is assumed that $(\boldsymbol{A}, \boldsymbol{B}_2)$ is stabilisable and $(\boldsymbol{C}_2, \boldsymbol{A})$ is detectable.

Described in transfer function form,

$$\begin{pmatrix} \boldsymbol{x}(s) \\ \boldsymbol{y}(s) \end{pmatrix} = \begin{pmatrix} \boldsymbol{G}_{11}(s) & \boldsymbol{G}_{12}(s) \\ \boldsymbol{G}_{21}(s) & \boldsymbol{G}_{22}(s) \end{pmatrix} \begin{pmatrix} \boldsymbol{w}(s) \\ \boldsymbol{u}(s) \end{pmatrix}$$

the transfer function matrix $\boldsymbol{G}(s)$ is defined by

$$\boldsymbol{G}(s) = \begin{pmatrix} \boldsymbol{G}_{11}(s) & \boldsymbol{G}_{12}(s) \\ \boldsymbol{G}_{21}(s) & \boldsymbol{G}_{22}(s) \end{pmatrix}.$$

Further, in order to later study the design of diagnosis in conjunction with closed-loop control, let the system described by (7.105) be controlled by an output feedback controller

$$\boldsymbol{u}(t) = \boldsymbol{K}(s)\boldsymbol{y}.$$

We have now the following definition of stabilisation by using output feedback.

Definition 7.8 *A proper system $\boldsymbol{G}(s)$ is said to be stabilisable through output feedback if there exists a proper controller $\boldsymbol{K}(s)$ internally stabilising $\boldsymbol{G}(s)$. Moreover, a proper controller $\boldsymbol{K}(s)$ is said to be admissible if it internally stabilises $\boldsymbol{G}(s)$.*

Next, we have the following result for the existence of a stabilising controller $\boldsymbol{K}(s)$ for the system $\boldsymbol{G}(s)$, where here and in the following

$$\boldsymbol{K}(s) = \left[\begin{array}{c|c} \boldsymbol{M} & \boldsymbol{N} \\ \hline \boldsymbol{P} & \boldsymbol{Q} \end{array}\right]$$

is used as abbreviation of

$$\boldsymbol{K}(s) = \boldsymbol{P}(s\boldsymbol{I} - \boldsymbol{M})^{-1}\boldsymbol{N} + \boldsymbol{Q}.$$

Lemma 7.4 *There exists a proper $\boldsymbol{K}(s)$ achieving internal stability if and only if $(\boldsymbol{A},\ \boldsymbol{B}_2)$ is stabilisable and $(\boldsymbol{C}_2,\ \boldsymbol{A})$ is detectable. Further, let $\boldsymbol{F}$ and $\boldsymbol{L}$ be such that $\boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F}$ and $\boldsymbol{A}+\boldsymbol{L}\boldsymbol{C}_2$ are stable, then an observer-based stabilising controller is given by*

$$\boldsymbol{K}(s) = \left[\begin{array}{c|c} \boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F}+\boldsymbol{L}\boldsymbol{C}_2+\boldsymbol{L}\boldsymbol{D}_{22}\boldsymbol{F} & \boldsymbol{F} \\ \hline -\boldsymbol{L} & \boldsymbol{O} \end{array}\right].$$

It is important to note that the stabilising controller for $\boldsymbol{G}(s)$ depend only on $\boldsymbol{G}_{22}(s)$. We need therefore only to look at $\boldsymbol{G}_{22}(s)$ when we are looking for stabilising controllers. This is also the case when we are using the Youla-Kucera parametrisation. In the following the argument s will often be omitted.

7.6.2 Youla-Kucera parameterisation in coprime factorisation form

First, let us consider two polynomials $m(s)$ and $n(s)$ with real coefficients. m and n are said to be coprime, if their greatest common divisor is 1 (they have no common zeros). It follows from Euclid's algorithm that f and g are coprime if and only if there exist polynomials $x(s)$ and $y(s)$ such that

$$mx + ny = 1. \tag{7.106}$$

This equation is called a Bezout identity. Similarly, the two stable transfer functions m and n are said to be coprime if there exist stable x and y such that Eq. (7.106) is satisfied.

Generally, two stable matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ are right coprime if they have equal number of columns and there exist stabel matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ such that

$$\begin{pmatrix} \boldsymbol{X} & \boldsymbol{Y} \end{pmatrix} \begin{pmatrix} \boldsymbol{M} \\ \boldsymbol{N} \end{pmatrix} = \boldsymbol{X}\boldsymbol{M} + \boldsymbol{Y}\boldsymbol{N} = \boldsymbol{I}.$$

This is equivalent to saying that the matrix

$$\begin{pmatrix} \boldsymbol{M} \\ \boldsymbol{N} \end{pmatrix}$$

is stable left invertible.

Similarly, two stable matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ are left coprime if they have equal number of rows and there exist stable X and Y such that

$$\begin{pmatrix} \boldsymbol{M} & \boldsymbol{N} \end{pmatrix} \begin{pmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \end{pmatrix} = \boldsymbol{M}\boldsymbol{X} + \boldsymbol{N}\boldsymbol{Y} = \boldsymbol{I}$$

or equivalently,

$$\begin{pmatrix} \boldsymbol{M} & \boldsymbol{N} \end{pmatrix}$$

is stable right invertible.

Now, let $\boldsymbol{G}_{22}(s)$ be a proper real-rational matrix. A right coprime factorisation of $\boldsymbol{G}_{22}(s)$ is a factorisation $\boldsymbol{G}_{22}(s) = \boldsymbol{N}\boldsymbol{M}^{-1}$ where $\boldsymbol{N}$ and $\boldsymbol{M}$ are stable right coprime matrices. Similarly, a left coprime factorisation has the form $\boldsymbol{G}_{22}(s) = \tilde{\boldsymbol{M}}^{-1}\tilde{\boldsymbol{N}}$ where $\tilde{\boldsymbol{N}}$ and $\tilde{\boldsymbol{M}}$ are left coprime. Note that, in these definitions, it is required that $\boldsymbol{M}$ and $\tilde{\boldsymbol{M}}$ are squared and nonsingular.

Based on the above, there exist the following result.

Lemma 7.5 *For each proper real-rational matrix $\boldsymbol{G}_{22}(s)$ there exist eight stable matrices satisfying the equations*

$$\boldsymbol{G}_{22}(s) = \boldsymbol{N}\boldsymbol{M}^{-1} = \tilde{\boldsymbol{M}}^{-1}\tilde{\boldsymbol{N}}$$
$$\begin{pmatrix} \tilde{\boldsymbol{X}} & \tilde{\boldsymbol{Y}} \\ -\tilde{\boldsymbol{N}} & \tilde{\boldsymbol{M}} \end{pmatrix} \begin{pmatrix} \boldsymbol{M} & -\boldsymbol{Y} \\ \boldsymbol{N} & \boldsymbol{X} \end{pmatrix} = \boldsymbol{I}.$$

This lemma defines a double coprime factorisation of $\boldsymbol{G}_{22}(s)$. It should be noted that it is always possible to make a coprime factorisation, if the system is stabilisable and detectable.

Now, let $\tilde{\boldsymbol{K}}$ be a stabilising controller for $\boldsymbol{G}_{22}(s)$ and let $\tilde{\boldsymbol{K}}$ have the following factorisation

$$\tilde{\boldsymbol{K}} = \boldsymbol{U}\boldsymbol{V}^{-1} = \tilde{\boldsymbol{V}}^{-1}\tilde{\boldsymbol{U}}.$$

A feedback system with positive feedback is stable if and only if

$$\begin{pmatrix} \boldsymbol{I} & -\tilde{\boldsymbol{K}} \\ -\boldsymbol{G}_{22} & \boldsymbol{I} \end{pmatrix}^{-1} \text{ is stable.}$$

Using the coprime factorisation of $\tilde{\boldsymbol{K}}$ we get the following conditions for internal stability.

Lemma 7.6 *Let $\boldsymbol{G}_{22}(s)$ be a proper real-rational matrix and*

$$\boldsymbol{G}_{22}(s) = \boldsymbol{N}\boldsymbol{M}^{-1} = \tilde{\boldsymbol{M}}^{-1}\tilde{\boldsymbol{N}}$$

be the stable right and left coprime factorisation. Then there exists a controller

$$\tilde{\boldsymbol{K}}_0 = \boldsymbol{U}_0\boldsymbol{V}_0^{-1} = \tilde{\boldsymbol{V}}_0^{-1}\tilde{\boldsymbol{U}}_0$$

with $\boldsymbol{U}_0, \boldsymbol{V}_0, \tilde{\boldsymbol{U}}_0$ and $\tilde{\boldsymbol{V}}_0$ stable such that

$$\begin{pmatrix} \tilde{\boldsymbol{V}}_0 & -\tilde{\boldsymbol{U}}_0 \\ -\tilde{\boldsymbol{N}} & \tilde{\boldsymbol{M}} \end{pmatrix} \begin{pmatrix} \boldsymbol{M} & \boldsymbol{U}_0 \\ \boldsymbol{N} & \boldsymbol{V}_0 \end{pmatrix} = \boldsymbol{I}.$$

Based on the above results, it is now possible to give a parametrisation of all controllers that stabilise $\boldsymbol{G}_{22}(s)$.

Theorem 7.6 *Let $\boldsymbol{G}_{22}(s)$ be a proper real-rational matrix and*

$$\boldsymbol{G}_{22}(s) = \boldsymbol{N}\boldsymbol{M}^{-1} = \tilde{\boldsymbol{M}}^{-1}\tilde{\boldsymbol{N}}$$

be the stable right and left coprime factorisation. Then the set of all proper controllers achieving internal stability is parameterised either by

$$\begin{aligned} \boldsymbol{K} &= (\boldsymbol{U}_0 + \boldsymbol{M}\boldsymbol{Q}_r)(\boldsymbol{V}_0 + \boldsymbol{N}\boldsymbol{Q}_r)^{-1} \\ &\quad \det(\boldsymbol{I} + \boldsymbol{V}_0^{-1}\boldsymbol{N}\boldsymbol{Q}_r)(\infty) \neq 0 \end{aligned} \tag{7.107}$$

for stable $\boldsymbol{Q}_r$ or by

$$\begin{aligned} \boldsymbol{K} &= (\tilde{\boldsymbol{V}}_0 + \boldsymbol{Q}_l\tilde{\boldsymbol{N}})^{-1}(\tilde{\boldsymbol{U}}_0 + \boldsymbol{Q}_l\tilde{\boldsymbol{M}}) \\ &\quad \det(\boldsymbol{I} + \boldsymbol{Q}_l\tilde{\boldsymbol{N}}\tilde{\boldsymbol{V}}_0^{-1})(\infty) \neq 0 \end{aligned} \tag{7.108}$$

for stable $\boldsymbol{Q}_l$ where $\boldsymbol{U}_0$, $\boldsymbol{V}_0$, $\tilde{\boldsymbol{U}}_0$ and $\tilde{\boldsymbol{V}}_0$ stable satisfied the Bezout identities:

$$\begin{aligned} \tilde{\boldsymbol{V}}_0\boldsymbol{M} - \tilde{\boldsymbol{U}}_0\boldsymbol{N} &= \boldsymbol{I}, \\ \tilde{\boldsymbol{M}}\boldsymbol{V}_0 - \tilde{\boldsymbol{N}}\boldsymbol{U}_0 &= \boldsymbol{I} \end{aligned}$$

Moreover, if $\boldsymbol{U}_0$, $\boldsymbol{V}_0$, $\tilde{\boldsymbol{U}}_0$ and $\tilde{\boldsymbol{V}}_0$ are chosen such that:

$$\begin{pmatrix} \tilde{\boldsymbol{V}}_0 & -\tilde{\boldsymbol{U}}_0 \\ -\tilde{\boldsymbol{N}} & \tilde{\boldsymbol{M}} \end{pmatrix} \begin{pmatrix} \boldsymbol{M} & \boldsymbol{U}_0 \\ \boldsymbol{N} & \boldsymbol{V}_0 \end{pmatrix} = \boldsymbol{I}.$$

Then we have

$$\begin{aligned} \boldsymbol{K} &= (\boldsymbol{U}_0 + \boldsymbol{M}\boldsymbol{Q}_r)(\boldsymbol{V}_0 + \boldsymbol{N}\boldsymbol{Q}_r)^{-1} \\ &= \left(\tilde{\boldsymbol{V}}_0 + \boldsymbol{Q}_r\tilde{\boldsymbol{N}}\right)^{-1}\left(\tilde{\boldsymbol{U}}_0 + \boldsymbol{Q}_r\tilde{\boldsymbol{M}}\right) \\ &= \mathcal{F}_l(\boldsymbol{J}_r, \boldsymbol{Q}_r), \end{aligned} \tag{7.109}$$

where

$$\boldsymbol{J}_r = \begin{pmatrix} \boldsymbol{U}_0\boldsymbol{V}_0^{-1} & \tilde{\boldsymbol{V}}_0^{-1} \\ \boldsymbol{V}_0^{-1} & -\boldsymbol{V}_0^{-1}\boldsymbol{N} \end{pmatrix}$$

and $\boldsymbol{Q}_r$ is stable and satisfies that $(\boldsymbol{I} + \boldsymbol{V}_0^{-1}\boldsymbol{N}\boldsymbol{Q}_y)(\infty)$ is invertible.

The Youla-Kucera parametrisation is shown in Fig. 7.22.

7.6.3 Parametrisation in the state-space form

The Youla-Kucera parametrisation derived in the above section was based on coprime factorisation, which may not be the form in which a particular fault-tolerant control problem is described. Further, popular toolboxes support a state-space description. This includes MATLAB. Therefore, a state-space description will be given in this section together with a description of the closed-loop transfer function as function of the free stable parameter $\boldsymbol{Q}$.

Consider the coprime factorisation in a state-space form. Using state feedback and observers, it is possible in a very simple way by to define a coprime factorisation in state space. The following result is available:

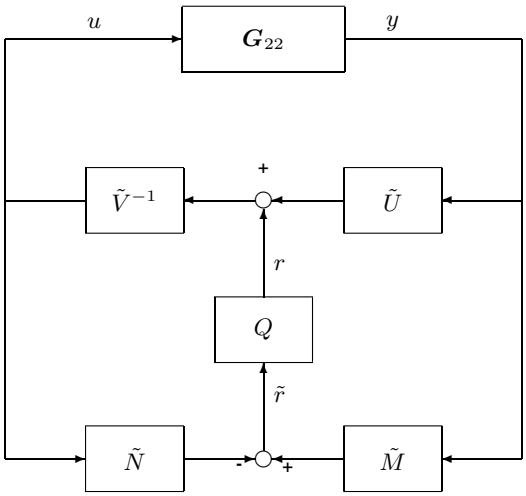


Fig. 7.22: Controller structure for the Youla-Kucera parametrisation

Let two coprime factorisations of $\boldsymbol{G}_{22}(s)$ be chosen as

$$\begin{pmatrix} \boldsymbol{M} & \boldsymbol{U}_0 \\ \boldsymbol{N} & \boldsymbol{V}_0 \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F} & \boldsymbol{F} & \boldsymbol{C}_2+\boldsymbol{D}_{22}\boldsymbol{F} \\ \boldsymbol{B}_2 & \boldsymbol{I} & \boldsymbol{D}_{22} \\ -\boldsymbol{L} & \boldsymbol{O} & \boldsymbol{I} \end{pmatrix}$$

$$\begin{pmatrix} \tilde{\boldsymbol{V}}_0 & -\tilde{\boldsymbol{U}}_0 \\ -\tilde{\boldsymbol{N}} & \tilde{\boldsymbol{M}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}+\boldsymbol{L}\boldsymbol{C}_2 & \boldsymbol{F} & \boldsymbol{C}_2 \\ -(\boldsymbol{B}_2+\boldsymbol{L}\boldsymbol{D}_{22}) & \boldsymbol{I} & -\boldsymbol{D}_{22} \\ \boldsymbol{L} & \boldsymbol{O} & \boldsymbol{I} \end{pmatrix},$$

where $\boldsymbol{F}$ and $\boldsymbol{L}$ are chosen such that $\boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F}$ and $\boldsymbol{A}+\boldsymbol{L}\boldsymbol{C}_2$ are both stable. It is now quite simple to give a state-space realisation of the $\boldsymbol{Q}$-parametrisation of all internal stabilising controllers. From Theorem 7.6 we have the linear fractional transformation formulation of all stabilising controllers. Using the state-space description of the coprime factorisation in $\boldsymbol{J}_y$ we get the following result.

Theorem 7.7 *Let $\boldsymbol{F}$ and $\boldsymbol{L}$ be such that $\boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F}$ and $\boldsymbol{A}+\boldsymbol{L}\boldsymbol{C}_2$ are stable. Then all controllers that internally stabilise $\boldsymbol{G}(s)$ can be parameterised as the transfer matrix from $\boldsymbol{y}$ to $\boldsymbol{u}$ given by $\mathcal{F}_l(\boldsymbol{J}_y, \boldsymbol{Q})$ where*

$$\boldsymbol{J}_y = \begin{pmatrix} \boldsymbol{A}+\boldsymbol{B}_2\boldsymbol{F}+\boldsymbol{L}\boldsymbol{C}_2+\boldsymbol{L}\boldsymbol{D}_{22}\boldsymbol{F} & \boldsymbol{F} & -(\boldsymbol{C}_2+\boldsymbol{D}_{22}\boldsymbol{F}) \\ -\boldsymbol{L} & \boldsymbol{O} & \boldsymbol{I} \\ \boldsymbol{B}_2+\boldsymbol{L}\boldsymbol{D}_{22} & \boldsymbol{I} & -\boldsymbol{D}_{22} \end{pmatrix}$$

with any $\boldsymbol{Q} \in \mathcal{RH}_\infty$ and $\boldsymbol{I}+\boldsymbol{D}_{22}\boldsymbol{Q}(\infty)$ is nonsingular.

The controller given in the theorem is sometimes called the $\boldsymbol{Q}$-observer-based controller. Further, note that when $\boldsymbol{Q} = \boldsymbol{O}$ the nominal controller turn out to be a standard full-order observer-based controller. Moreover, it can be shown that the separation between the design of the state feedback gain $\boldsymbol{F}$ and the observer gain $\boldsymbol{L}$ is still valid and a separation between the nominal controller and the $\boldsymbol{Q}$ parameter. This can be shown by setting up a state-space description of the controller together with the $\boldsymbol{Q}$ parameter and use the state vector $\bar{\boldsymbol{x}} = (\boldsymbol{x} \quad \boldsymbol{x} - \hat{\boldsymbol{x}} \quad \boldsymbol{x}_q)$, where $\boldsymbol{x}_q$ is the state vector for $\boldsymbol{Q}$.

Next, let us look at the closed-loop transfer function when we have applied a $\boldsymbol{Q}$-parameterised controller as given in Theorem 7.7. The closed-loop transfer function is given by the following linear fractional transformation:

$$z = \mathcal{F}_l(\boldsymbol{G},\, \boldsymbol{K})w = \mathcal{F}_l(\boldsymbol{G},\, \mathcal{F}_l(J_y,\, \boldsymbol{Q}))w = \mathcal{F}_l(\boldsymbol{T},\, \boldsymbol{Q})w.$$

We need now just to give a state-space description of T. By using straightforward and tedious algebra, we get the following result.

Theorem 7.8 *Let $\boldsymbol{F}$ and $\boldsymbol{L}$ be such that $\boldsymbol{A} + \boldsymbol{B}_2\boldsymbol{F}$ and $\boldsymbol{A} + \boldsymbol{L}\boldsymbol{C}_2$ are stable. Then the set of a closed-loop transfer matrices from w to z achievable by an stabilising proper controller is equal to*

$$\mathcal{F}_l(\boldsymbol{T},\, \boldsymbol{Q}) = \boldsymbol{T}_{11} + \boldsymbol{T}_{12}\boldsymbol{Q}\boldsymbol{T}_{21},\ \boldsymbol{Q} \in \mathcal{RH}_\infty,\ \boldsymbol{I} + \boldsymbol{D}_{22}\boldsymbol{Q}(\infty),$$

where $\boldsymbol{T}$ is given by

$$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{T}_{11} & \boldsymbol{T}_{12} \\ \boldsymbol{T}_{21} & \boldsymbol{T}_{22} \end{pmatrix} = \left(\begin{array}{cc|cc} \boldsymbol{A} + \boldsymbol{B}_2\boldsymbol{F} & -\boldsymbol{B}_2\boldsymbol{F} & \boldsymbol{B}_1 & \boldsymbol{B}_2 \\ \boldsymbol{O} & \boldsymbol{A} + \boldsymbol{L}\boldsymbol{C}_2 & \boldsymbol{B}_1 + \boldsymbol{L}\boldsymbol{D}_{21} & \boldsymbol{O} \\ \hline \boldsymbol{C}_1 + \boldsymbol{D}_{12}\boldsymbol{F} & -\boldsymbol{D}_{12}\boldsymbol{F} & \boldsymbol{D}_{11} & \boldsymbol{D}_{12} \\ \boldsymbol{O} & \boldsymbol{C}_2 & \boldsymbol{D}_{21} & \boldsymbol{O} \end{array}\right).$$

It is important to note that the closed-loop transfer matrix $\boldsymbol{T}$ is an affine function of the controller parameter matrix $\boldsymbol{Q}$, since $\boldsymbol{T}_{22} = \boldsymbol{O}$. This is the reason why the $\boldsymbol{Q}$-parametrisation is so useful. One example where the parametrisation is very useful is in connection with optimisation of controllers by using numerical tools.

7.6.4 Simultaneous design of the controller and the residual generator

In the closed loop, there is an interaction between the sensitivity of the residual from a fault detection filter and the natural suppression of any fault within a closed loop. The design of closed-loop control and residual generator can therefore be considered an integrated design problem. Consider therefore the simultaneous design of feedback controller and residual generator. The design setup illustrated in Fig. 7.23. The setup uses the standard problem philosophy.

As stated earlier and as detailed in appendix, the standard design provides a controller $\boldsymbol{K}(s)$ designed such that the closed loop is internally stable and a suitable norm of the closed-loop transfer function from $\boldsymbol{w}$ to $\boldsymbol{z}$ is minimised or made smaller than a pre-specified level.

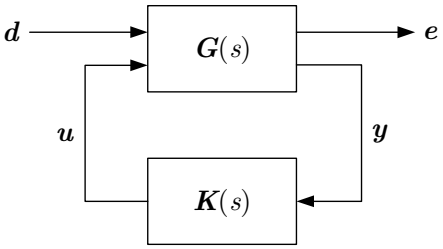


Fig. 7.23. Control system in standard configuration

Instead of using a standard one parameter controller as shown in Fig. 7.23, a two parameter controller can be employed:

$$\begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{a} \end{pmatrix} = \begin{pmatrix} \boldsymbol{K}_1 \\ \boldsymbol{K}_2 \end{pmatrix} \boldsymbol{y}$$

The additional output signal $\boldsymbol{a}$ from the controller is a diagnostic signal. It will be applied to derive an estimate of faults in the controlled system.

Let the open loop transfer function be given by:

$$\begin{pmatrix} \boldsymbol{e} \\ \boldsymbol{y} \end{pmatrix} = \begin{pmatrix} \boldsymbol{G}_{ed} & \boldsymbol{G}_{ef} & \boldsymbol{G}_{eu} \\ \boldsymbol{G}_{yd} & \boldsymbol{G}_{yf} & \boldsymbol{G}_{yu} \end{pmatrix} \begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{f} \\ \boldsymbol{u} \end{pmatrix} \tag{7.110}$$

To obtain a good estimation of the individual faults, fault models are included in the generalised system as frequency weighings on the faults signals

$$\boldsymbol{f} = \boldsymbol{W}_f(s)\boldsymbol{v},$$

where $\boldsymbol{v}$ is a signal that is anticipated to have a flat power spectrum. The generalised setup is shown in Fig. 7.24.

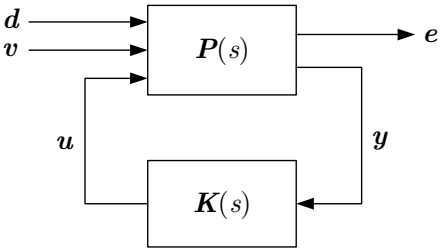


Fig. 7.24. Generalised setup for control and fault detection

Now we need to formulate the design setup in Fig. 7.24 as a standard design problem as illustrated in Fig. 7.23. For doing this, define an additional output $\boldsymbol{r}$ as the fault estimation error:

$$\boldsymbol{r} = \boldsymbol{f} - \boldsymbol{a}. \tag{7.111}$$

This is the standard way of formulating a filter design problem in the standard problem setup. The generalised system $\boldsymbol{P}(s)$ is then given by:

$$\begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{r} \\ \boldsymbol{y} \end{pmatrix} = \boldsymbol{P}(s) \begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{v} \\ \boldsymbol{u} \end{pmatrix} \tag{7.112}$$

with

$$\boldsymbol{P}(s) = \left(\begin{array}{cc|cc} \boldsymbol{G}_{ed} & \boldsymbol{G}_{ef}\boldsymbol{W}_f & \boldsymbol{G}_{eu} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{W}_f & \boldsymbol{O} & -\boldsymbol{I} \\ \hline \boldsymbol{G}_{yd} & \boldsymbol{G}_{yf}\boldsymbol{W}_f & \boldsymbol{G}_{yu} & \boldsymbol{O} \end{array}\right).$$

Using the system setup in (7.112) and apply a two parameter controller

$$\boldsymbol{u} = \boldsymbol{K}(s)\boldsymbol{y}$$

we get the following closed-loop transfer function

$$\begin{pmatrix} \boldsymbol{e} \\ \boldsymbol{r} \end{pmatrix} = \boldsymbol{T}_{cl}(s) \begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{v} \end{pmatrix}$$

with

$$\boldsymbol{T}_{cl}(s) = \begin{pmatrix} \boldsymbol{G}_{ed} & \boldsymbol{G}_{ef}\boldsymbol{W}_f \\ \boldsymbol{O} & \boldsymbol{W}_f \end{pmatrix} +$$

$$\begin{pmatrix} \boldsymbol{G}_{eu} & \boldsymbol{O} \\ \boldsymbol{O} & -\boldsymbol{I} \end{pmatrix} \boldsymbol{K}(s)(\boldsymbol{I} - \begin{pmatrix} \boldsymbol{G}_{yu} & \boldsymbol{O} \end{pmatrix} \boldsymbol{K}(s))^{-1} \begin{pmatrix} \boldsymbol{G}_{yd} & \boldsymbol{G}_{yf}\boldsymbol{W}_f \end{pmatrix}$$

For simplicity, assume that $\boldsymbol{G}(s)$ is open loop stable (the unstable case can be dealt with as well in this methodology, but is computationally more difficult). Then the Youla-Kucera parameterisation of all stabilizing controllers can be obtained by making the substitution:

$$\begin{aligned} \boldsymbol{Q}(s) &= \boldsymbol{K}(s)(\boldsymbol{I} - \begin{pmatrix} \boldsymbol{G}_{yu} & \boldsymbol{O} \end{pmatrix} \boldsymbol{K}(s))^{-1} \\ \boldsymbol{K}(s) &= \boldsymbol{Q}(s)(\boldsymbol{I} + \begin{pmatrix} \boldsymbol{G}_{yu} & \boldsymbol{O} \end{pmatrix} \boldsymbol{Q}(s))^{-1}, \end{aligned} \tag{7.113}$$

where $\boldsymbol{Q}(s)$ is a stable proper transfer function, the Youla parameter. Further, let $\boldsymbol{Q}(s)$ be partitioned as:

$$\boldsymbol{Q}(s) = \begin{pmatrix} \boldsymbol{Q}_1(s) \\ \boldsymbol{Q}_2(s) \end{pmatrix}$$

Then the following equation for the closed-loop transfer function $\boldsymbol{T}_{cl}$ is obtained:

$$\boldsymbol{T}_{cl}(s) = \begin{pmatrix} \boldsymbol{G}_{ed} + \boldsymbol{G}_{eu}\boldsymbol{Q}_1\boldsymbol{G}_{yd} & \boldsymbol{G}_{ef}\boldsymbol{W}_f + \boldsymbol{G}_{eu}\boldsymbol{Q}_1\boldsymbol{G}_{yf}\boldsymbol{W}_f \\ -\boldsymbol{Q}_2\boldsymbol{G}_{yd} & \boldsymbol{W}_f - \boldsymbol{Q}_2\boldsymbol{G}_{yf}\boldsymbol{W}_f \end{pmatrix} \tag{7.114}$$

Note that $\boldsymbol{Q}_1$ only appears in the first row of $\boldsymbol{T}_{cl}$ and $\boldsymbol{Q}_2$ only in the second row of $\boldsymbol{T}_{cl}$. A separation between the design of $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ has therefore been obtained by using a Youla parameterisation. This is a salient feature of this design approach.

Calculating $\boldsymbol{K}(s)$ directly from (7.113) results in the following equation:

$$\begin{aligned} \boldsymbol{K}(s) &= \begin{pmatrix} \boldsymbol{Q}_1(s)(\boldsymbol{I}+\boldsymbol{G}_{yu}\boldsymbol{Q}_1(s))^{-1} \\ \boldsymbol{Q}_2(s)(\boldsymbol{I}+\boldsymbol{G}_{yu}\boldsymbol{Q}_1(s))^{-1} \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{Q}_1(s)(\boldsymbol{I}+\boldsymbol{G}_{yu}\boldsymbol{Q}_1(s))^{-1} \\ \boldsymbol{Q}_2(s)(\boldsymbol{I}-\boldsymbol{G}_{yu}\boldsymbol{K}_1(s)) \end{pmatrix} \end{aligned} \tag{7.115}$$

The result indicates that also the original controller structure is separated in a design of the feedback controller $\boldsymbol{K}_1(s)$ and a design of the fault detection filter $\boldsymbol{K}_2(s)$. Design of the fault detection filter $\boldsymbol{K}_2(s)$ depends of the controller design $\boldsymbol{K}_1(s)$.

This result is essential for proper design of residual generators working in closed loop. It is also important for the re-design problem since the effect that the re-designed controller has on the diagnostic filters cannot be ignored.

7.7 Handling the fault recovery transients

7.7.1 Mastering transient upon switching between controllers

In the previous sections, controller reconfiguration or accommodation often amounts to switching to a newly designed controller or to an element of a bank of controllers which is different from the controller presently in the loop. When the considered control laws are of the state feedback or output feedback type, namely when static laws are used, no precaution is required to switch between controllers with the same reference input. However, the situation is different for dynamic controllers. Indeed, the state of the controller which is not in the loop has to be initialised properly before this controller is introduced in the loop, in order to avoid bumps in the system response. One method to achieve this goal is presented here. It amounts to feeding back to each controller, be it active in the loop or not, the manipulated variable actually applied to the process (namely the process input). This mechanism is similar to an anti-windup strategy, which is normally used to handle actuator saturation in a control loop.

Without loss of generality, a situation with two controllers is considered here, so that one controller, say controller 1, is active in the loop, and controller 2 is the controller towards which switching occurs (see Fig. 7.25). Both controllers are supposed to be described by a linear state-space model of the form

$$\begin{aligned} \dot{\boldsymbol{x}}_{ci}(t) &= \boldsymbol{A}_{ci}\boldsymbol{x}_{ci}(t) + \boldsymbol{B}_{ci}\boldsymbol{y}_{\mathrm{ref}}(t) + \boldsymbol{E}_{ci}\,\boldsymbol{y}(t) \qquad \boldsymbol{x}_{ci}(0) = \boldsymbol{x}_{ci}^0 \\ \boldsymbol{u}_i(t) &= \boldsymbol{C}_{ci}\boldsymbol{x}_{ci}(t) + \boldsymbol{D}_{ci}\boldsymbol{y}_{\mathrm{ref}}(t) + \boldsymbol{F}_{ci}\boldsymbol{y}(t) \qquad i = 1, 2, \end{aligned} \tag{7.116}$$

where $\boldsymbol{x}_{ci}(t)$, $\boldsymbol{u}_i(t)$, $\boldsymbol{y}(t)$ and $\boldsymbol{y}_{\mathrm{ref}}(t)$ are respectively the controller state, the controller output, the measured plant output and the reference.

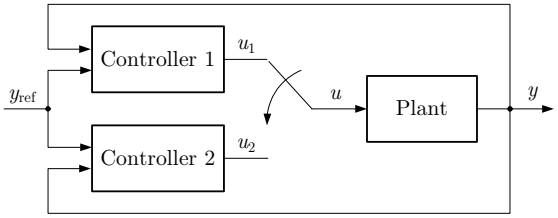


Fig. 7.25. Two-controller scheme

As explained in the previous paragraph, to obtain a smooth switching towards controller 2, the state of this controller must be properly initialised. This can be achieved thanks to an observer-based anti-windup mechanism. It amounts to feeding back the difference, $\boldsymbol{u}(t) - \boldsymbol{u}_2(t)$, between the plant input and the controller output, in the state equation. Hence the state equation for controller 2 becomes

$$\begin{array}{rcl} \dot{\boldsymbol{x}}_{c2}(t) & = & \boldsymbol{A}_{c2}\boldsymbol{x}_{c2}(t) + \boldsymbol{B}_{c2}\boldsymbol{y}_{\text{ref}}(t) + \boldsymbol{E}_{c2}\,\boldsymbol{y}(t) + \boldsymbol{L}_2(\boldsymbol{u}(t) - \boldsymbol{u}_2(t)) \\ \boldsymbol{u}_2(t) & = & \boldsymbol{C}_{c2}\boldsymbol{x}_{c2}(t) + \boldsymbol{D}_{c2}\boldsymbol{y}_{\text{ref}}(t) + \boldsymbol{F}_{c2}\boldsymbol{y}(t). \end{array} \tag{7.117}$$

Substituting the output equation for $\boldsymbol{u}_2(t)$ in the state equation of (7.117) yields

$$\begin{array}{rcl} \dot{\boldsymbol{x}}_{c2}(t) & = & (\boldsymbol{A}_{c2} - \boldsymbol{L}_2\boldsymbol{C}_{c2})\boldsymbol{x}_{c2}(t) + (\boldsymbol{B}_{c2} - \boldsymbol{L}_2\boldsymbol{D}_{c2})\boldsymbol{y}_{\text{ref}}(t) \\ & & +(\boldsymbol{E}_{c2} - \boldsymbol{L}_2\boldsymbol{F}_{c2})\,\boldsymbol{y}(t) + \boldsymbol{L}_2\boldsymbol{u}(t) \\ \boldsymbol{u}_2(t) & = & \boldsymbol{C}_{c2}\boldsymbol{x}_{c2}(t) + \boldsymbol{D}_{c2}\boldsymbol{y}_{\text{ref}}(t) + \boldsymbol{F}_{c2}\boldsymbol{y}(t) \end{array} \tag{7.118}$$

This shows that $\boldsymbol{L}_2$ should be chosen in such a way that $(\boldsymbol{A}_{c2} - \boldsymbol{L}_2\boldsymbol{C}_{c2})$ has all its eigenvalues inside the open left half plane in order for $\boldsymbol{x}_{c2}(t)$ to reach a steady state value when controller 2 is not inserted in the loop, in the absence of change in $\boldsymbol{y}_{\text{ref}}(t)$, $\boldsymbol{y}(t)$ and $\boldsymbol{u}(t)$. Possible options consist in choosing $\boldsymbol{L}_2$ so that all eigenvalues lie at the origin, or so that $\boldsymbol{L}_2 = \boldsymbol{B}_{c2}\boldsymbol{D}_{c2}^{-1}$, which corresponds to the so-called conditioning technique. The latter approach requires a square full rank matrix $\boldsymbol{D}_{c2}$, although this conditions can be weakened. It also requires that the zeros of the controller lie in the open left half plane, since these are precisely the eigenvalues of $\boldsymbol{A}_{c2} - \boldsymbol{B}_{c2}\boldsymbol{D}_{c2}^{-1}\boldsymbol{C}_{c2}$, a condition often fulfilled in practice.

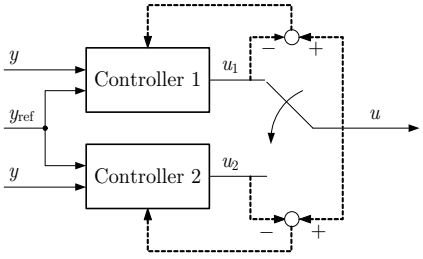


Fig. 7.26. Two-contoller scheme with anti-windup mechanism

Obviously, for reason of symmetry, to allow switching from controller 2 to controller 1, the latter controller must be provided with a similar anti-windup feature. Its state-space equations are thus written like (7.118), with index 1 substituted for 2. The resulting block diagram is given in Fig. 7.26.

7.7.2 Progressive fault accommodation

In the ideal Fault-tolerant linear quadratic problem described above (Section 7.3) the fault detection, isolation and estimation process, which provides the model $(\boldsymbol{A},\ \boldsymbol{B}_f)$ of the post-fault system, and the fault accommodation process, which includes the resolution of the new algebraic Riccati equation and the re-design of the feedback control take no time. However, it has been already noted that in practice, three time periods exist:

Time window	System situation	System	Control
$[0, t_f[$	Nominal operation	$(\boldsymbol{A},\ \boldsymbol{B})$	$\boldsymbol{u} = -\boldsymbol{R}^{-1}\boldsymbol{B}'P\boldsymbol{x}$
$[t_f, t_a[$	Fault detection, isolation and estimation process, fault accommodation process delay	$(\boldsymbol{A},\ \boldsymbol{B}_f)$	$\boldsymbol{u} = -\boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{P}\boldsymbol{x}$
$[t_a, \infty)$	Fault is accommodated	$(\boldsymbol{A},\ \boldsymbol{B}_f)$	$\boldsymbol{u}_f = -\boldsymbol{R}^{-1}\boldsymbol{B}_f'\boldsymbol{P}_f\boldsymbol{x}$

During the time period $[t_f, t_a[$ the faulty system $(\boldsymbol{A},\ \boldsymbol{B}_f)$ is still controlled by the nominal control $\boldsymbol{u} = -\boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{P}\boldsymbol{x}$. This control is optimal for $(\boldsymbol{A},\ \boldsymbol{B})$ and the closed loop $\boldsymbol{A} - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{P}$ is stable, but no guarantee can be given when $\boldsymbol{B}$ is replaced by $\boldsymbol{B}_f$ and the closed-loop $\boldsymbol{A} - \boldsymbol{B}_f\boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{P}$ may be unstable. If $t_a - t_f$ is not small enough, although the new control law $\boldsymbol{u}_f$ will recover the system stability and provide the best possible performance when applied, the system state may violate some physical limits or it may lead to a non-admissible value of the system cost. Note that physical limits are not formalised in the standard LQ problem setting, but they are usually taken into account by an appropriate choice of the weighting matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$.

Therefore, the practical fault accommodation problem is to design a fault detection, isolation and estimation process and fault accommodation process procedure such that $t_a - t_f$ is as small as possible, which needs minimizing both the fault detection, isolation and estimation process delay and the fault accommodation process delay. As far as fault accommodation is concerned, this can be obtained by two complementary strategies:

- design an algorithm that computes the accommodated control $\boldsymbol{u}_f$ (i.e. that solves the algebraic Riccati Eequation) in minimum time,
- design an algorithm that computes a sequence of controls that will eventually converge to $\boldsymbol{u}_f$ and will stabilise the system as soon as possible. Such an algorithm

belongs to the family of "anytime" algorithms, which means that the result of any iteration is acceptable, and it will improve as the number of iterations increases. This is the *progressive accommodation* strategy.

Newton-Raphson iteration scheme for solving the algebraic Riccati equation. . The Newton-Raphson iteration scheme has been proposed in the literature as an effective way of solving the algebraic Riccati equation. Let $\boldsymbol{P}_i$ be the unique solution of the Lyapunov equation

$$\boldsymbol{P}_i(\boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{F}_{i-1})+(\boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{F}_{i-1})'\boldsymbol{P}_i=-\boldsymbol{Q}-\boldsymbol{F}'_{i-1}\boldsymbol{R}\boldsymbol{F}_{i-1}, \quad (7.119)$$

where

$$\boldsymbol{F}_i=\boldsymbol{R}^{-1}\boldsymbol{B}_f\boldsymbol{P}_i \quad (7.120)$$

for all $i=1,2,\cdots$ and the initial $\boldsymbol{F}_0$ is given. If $\boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{F}_0$ is stable, then all matrices $\boldsymbol{K}_i$ are positive definite, and one has the convergence result

$$\begin{array}{ll}(1) & \boldsymbol{P}_0\geq\boldsymbol{P}_1\geq ...\geq\boldsymbol{P}_i\geq\boldsymbol{P}_{i+1}\geq ...\geq\boldsymbol{P}_f, \quad i=1,2,\cdots\\ (2) & \lim_{i\to\infty}\boldsymbol{P}_i=\boldsymbol{P}_f,\end{array} \quad (7.121)$$

where $\boldsymbol{P}_f$ is the solution of the algebraic Riccati equation

$$\boldsymbol{P}_f(\boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{F}_f)+(\boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{F}_f)'\boldsymbol{P}_f=-\boldsymbol{Q}-\boldsymbol{F}'_f\boldsymbol{R}\boldsymbol{F}_f.$$

Progressive Accommodation (PA) scheme. The PA scheme is based on the Newton-Raphson algorithm. Assume that iteration i takes a time Δ_i, and consider the sequence

$$t_i=t_{init}+\sum_{j=1}^{i}\Delta_j, \quad i=1,2,...,$$

where t_{init} is the time at which the Newton-Raphson algorithm is initialised after the fault has been detected, isolated and estimated (note that $t_f<t_{fdi}<t_{init}$). t_i is the time at which the result $\boldsymbol{F}_i$ becomes available (note that the constancy of Δ_i is not necessary, the scheme can therefore be employed whatever the tasks scheduling strategy of the FTC computer). The idea of progressive accomodation is to apply the feedback control law $\boldsymbol{u}_i=-\boldsymbol{F}_i\boldsymbol{x}$ on the time interval $[t_i,t_{i+1}[$. As a result, the system behaviour after the fault occurrence is

$$\dot{\boldsymbol{x}} = (\boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{R}^{-1}\boldsymbol{B}'\boldsymbol{P})\boldsymbol{x}, \quad t\in[t_f,t_{init}[\quad (7.122)$$

$$\dot{\boldsymbol{x}} = (\boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{F}_0)\boldsymbol{x}, \quad t\in[t_{init},t_1[\quad (7.123)$$

$$\dot{\boldsymbol{x}} = (\boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{F}_i)\boldsymbol{x}, \quad t\in[t_i,t_{i+1}[\quad i=1,2,..., \quad (7.124)$$

where $\boldsymbol{F}_0$ is the Newton-Raphson initialisation at time t_{init} . It can be shown from (7.121) that if the system $(\boldsymbol{A},\boldsymbol{B}_f)$ is stabilised by $\boldsymbol{F}_0$, then it is stabilised by all $\boldsymbol{F}_i$, and each $\boldsymbol{F}_i$ is better than the previous one with respect to the LQ cost. Moreover,

Progressive Accommodation results in a lower cost than the one associated with controlling the system by the nominal control until the Newton-Raphson algorithm has converged (which means the accommodated solution is computed) and then applying the accommodated control. Figure 7.27 shows the fault-tolerant system architecture using the PA scheme.

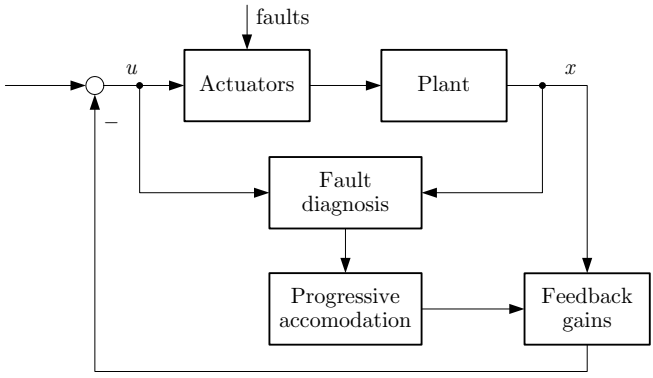


Fig. 7.27. Progressive Accommodation scheme

$\boldsymbol{A} - \boldsymbol{B}_f \boldsymbol{F}_0$ being stable is only a sufficient condition for the PA procedure to converge. Convergence may be obtained in some cases, even when the initial feedback does not stabilise the system. This happens in the following example. □

Example 7.9 *Progressive Accommodation of a first order system*

Consider the following LQ problem

$$\begin{aligned} \dot{\boldsymbol{x}} &= -\boldsymbol{x} + \boldsymbol{u} \qquad \boldsymbol{x}(0) = 4 \\ \boldsymbol{J} &= \int_0^\infty [\boldsymbol{x}^2(t) + \boldsymbol{u}^2(t)]dt \end{aligned}$$

The nominal Algebraic Riccati Equation is $\boldsymbol{P}^2 + 2\boldsymbol{P} - 1 = 0$ leading to the optimal control $\boldsymbol{u} = (1 - \sqrt{2})\boldsymbol{x}$, and closed-loop behaviour $\dot{\boldsymbol{x}} = -\sqrt{2}\boldsymbol{x}$. Let the faulty system be

$$\dot{\boldsymbol{x}} = -\boldsymbol{x} - 2\sqrt{2}\boldsymbol{u} \qquad t \geq 1$$

Under the nominal control, the faulty system behaviour is $\dot{\boldsymbol{x}} = (3 - 2\sqrt{2})\boldsymbol{x}$ which is instable. The new Algebraic Riccati Equation is $8\boldsymbol{P}_f^2 + 2\boldsymbol{P}_f - 1 = 0$ whose stable solution gives the optimal control of the faulty system $\boldsymbol{u}_f = \frac{\sqrt{2}}{2}\boldsymbol{x}$ and the closed-loop behaviour $\dot{\boldsymbol{x}} = -3\boldsymbol{x}$. The Newton-Raphson algorithm results in

$$\boldsymbol{P}_i = \frac{1 + 8\boldsymbol{P}_{i-1}^2}{2(1 + 8\boldsymbol{P}_{i-1})}$$

which converges to the solution of the new Algebraic Riccati Equation. The table below shows the evolution of $\boldsymbol{P}_i$, while Fig. 7.28 shows the evolution of the system state when

fault accommodation is applied after convergence of the Newton-Raphson scheme (which takes 3 iterations, with $\Delta = 1s$) (classical approach, dashed line), and using the progressive accommodation scheme (continuous line).

i	0	1	2	3	4	5
$k_i \times 10^2$	41.42	27.5	25.08	25	25	25

□

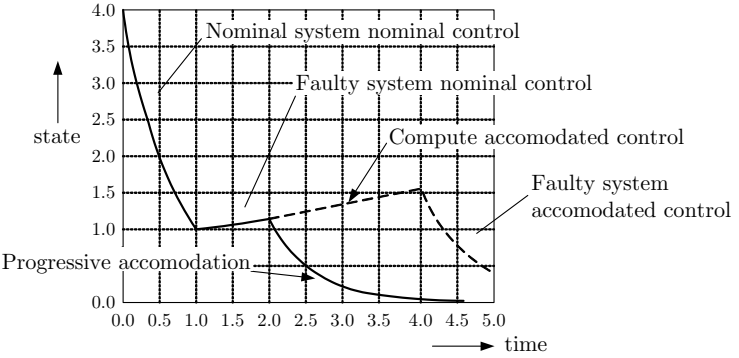


Fig. 7.28. Comparison of the classical and the progressive accommodation schemes

7.8 Exercises

Exercise 7.1 *Reconfiguration by model matching techniques*
Consider the plant

$$\begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{pmatrix} = \begin{pmatrix} -\frac{1}{T_1} & 0 \\ \frac{1}{T_2} & -\frac{1}{T_2} \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix}$$

$$y(t) = (1 \quad 1) \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}$$

and the proportional controller

$$\begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix} = \begin{pmatrix} -k_1 \\ 0 \end{pmatrix} y(t).$$

If the actuator 1 fails, the control loop should be closed with the help of the redundant control input $u_1(t)$. Does the model-matching approach yield a stable closed-loop system? Is the performance of the closed-loop system improved with respect to the nominal loop if the Markov parameter appraoch is used? □

Exercise 7.2 *Virtual actuator*
For the unstable system

$$\begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} 2 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix}$$
$$y(t) = (2 \quad 1) \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}$$

a proportional controller

$$\begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix} = \begin{pmatrix} -k_1 \\ 0 \end{pmatrix} y(t).$$

should be found that stabilises the system. In case of the actuator failure a virtual actuator should be used to stabilise the system with the nominal controller. Find reasonable parameters of the virtual actuator and prove that the reconfigured closed-loop system is stable. □

7.9 Bibliographical notes

A survey on fault accommodation is given in [193] and [206].

[96] is one of the earliest papers on controller reconfiguration by model-matching. [70] describes an improvement of the pseudo-inverse method for the ensurance of stability. A proof of Lemma 7.4 can be found in [279].

Further extensions of the pseudo-inverse method, that result in the so-called admissible model matching approach have been recently given in [226] and [227].

The ideas of the virtual sensor and virtual actuator have been developed in [160], [162]. A thorough treatment can be found in the recent monograph [240]. These concepts have been experimentally tested at a laboratory process [162] and a two-degrees-of-freedom helicopter model [161]. The generalised version of the virtual actuator, which is explained in this chapter, has been proposed in [149].

Model-predictive control has been used in [164] for fault accommodation and reconfiguration. The model-predictive controller uses all available input signals u_i and measurable output signals y_i which comprise the vectors $\boldsymbol{u}$ and $\boldsymbol{y}$ as before rather than only those input and output signals are used in the nominal feedback loop. If on the supervision level a fault f is detected, the inequality constraints included in the optimisation problem can be changed so that the model-predictive controller adapts to the faulty system. This can be done in a very easy way for actuator faults. If the diagnostic algorithm shows the the j-th actuator is faulty, the equality constraint $u_j = 0$ is included in the optimisation problem in order to ensure that the controller does not really use the j-th input. Then the model-predictive controller moves its control activity towards the available actuators, which can be interpreted as an on-line reconfiguration of the control loop.

As model-predictive control necessitates a rather large on-line computing capacity and as its reconfigurability property is, more or less, restricted to actuator faults this method should be used for ensuring fault-tolerance only if the advantages of model-predictive control has to be exploited for the faultless plant as well. For applications, where a fixed (linear) controller is sufficient for satisfying the control requirements for the faultless plant the reconfiguration should be carried out by methods described in the earlier sections, which eventually result in a new fixed control law.

A control mixer approach to deal with actuator faults was pursued by [274], [275]. A wider area of reconfiguration was studied in [273] and [269].

The results on controller re-design are based in the results for the Youla-Kucera parametrisation given in [186], [250] and [119]. See also [183], where the Youla-Kucera parametrisation has been applied in connection with tuning controllers. The exact, the almost exact and the optimal design problems for $\boldsymbol{Q}$ have been considered in detail in [209].

The properties of combined fault diagnosis and control were treated in [193]. The results in Section 7.6.4 are based on [243] and [244] which focus on the use of fault estimation within a reliable control framework [259]. The methods for re-configuration design are comparatively new within the fault-tolerant control domain. A few schemes have come into real application. Predetermined design for accommodation was demonstrated for a satellite in [26], and [25]. Techniques using logic inference on qualitative models were used in [138] and [154].

A related area is the control of discrete-event dynamical systems which have a known structure with pre-determined events that occur with unknown instants and sequence [264]. Diagnosis for discrete-event dynamical system was treated in [211] and [156].

Recoverability is concerned with the possibility either to accommodate the faults or to re-configure the system when faults occur. It has been considered from the point of view of the system structural properties, e.g. observability or controllability, extending the evaluation of these properties to the faulty system. For example, the smallest second order mode, first introduced in [170], has been proposed as a reconfigurability measure in [268], while the determinants of the controllability and observability Gramians were preferably suggested in [21]. In [237] and [224], two reconfigurability measures have been proposed to evaluate the size of the set of fault-tolerant situations, namely the number of recoverable failures (redundancy degrees) and the mean time until a non-recoverable failure occurs.

The recoverability problem can also be considered from the point of view of a specific objective, as analysed in [236], by considering the question: "Can the objective be achieved either through fault accommodation or through system reconfiguration?"

Due to the discrete nature of fault occurrence and reconfiguration, fault-tolerant control systems are hybrid in nature according to [67], [66].

Many different approaches can be used to solve the fault-tolerant control problem $< O, \hat{\mathcal{S}}, \hat{\Theta}, \mathcal{U} >$ [193].

Most commonly, human intervention is needed, using decision support from the diagnosis and overall goals for the plant [131], [236].

Theorem 7.6 has been proved in [279].

The design methods considered in Section 7.6 are based on the same conditions as the methods described in [220]. Further results on using the Youla-Kucera parameterisation for fault-tolerant control in the additive fault, multiplicative fault and parameter fault cases can be found in [178], [185], [182], [245] and [186]. An architecture for fault-tolerant control, based on joint controller and FDI design was presented in [176].

The link between mastering the transient upon controller switching and handling actuator saturation has been recognised for a long time. Indeed, both problems involve the discrepancy between the controller output and the process input, which might lead to performance degradation and even instability of the closed-loop. Anti-windup methods have thus been developed with a view to handle both problems (see [4] for the observer-based approach, and [87], [199], [88] for the conditioning technique). In [85] and [276], they are explicitly introduced in multi-controller schemes such as found in hybrid or switched-mode systems in order to avoid undesirable switching transients. The anti-windup mechanisms used in [85] are high gain feedback loops around each idle controller, which force the controller outputs to track the process input, while in [276] each controller is augmented with a dynamics identical to that of the plant in order to allow the controller state to evolve in an appropriate way when the controller is not connected to the plant input. The latter scheme is cumbersome when the number of controllers is large.

Dedicated methods have also been studied for handling transient in controller switching. In [84], the authors recast the problem in an associated tracking problem, where the standby controller is viewed as a dynamic system of which the output should track the manipulated

signal (plant input) by means of a two-degree-of-freedom controller. In [272], a simple new realization of a set of linear SISO controllers is described that inherently assures bumpless transient upon switching between controllers.

[271] made a rigorous analysis of controller switching and suggested an adaptive method to obtain bounded signals with a nominal controller from the instant a fault is detected until controller reconfiguration is made.

Chapter 8

Diagnosis and reconfigurable control of discrete-event systems

This chapter concerns discrete-event dynamical systems that are described by stochastic automata. Based on the solution to the state observation problem developed in Section 8.3, the fault diagnostic problem will be solved in Section 8.4 by "observing" the unknown state of the fault model. Section 8.7 shows how a supervisor can be reconfigured after a fault has been detected.

8.1 Motivation

This chapter is devoted to discrete-event dynamical systems whose behaviour is described by sequences of input and output events. In contrast to the preceding chapters where the continuous changes of the signals occurring in the system have been investigated, only abrupt changes of the signal values are considered here. The signals are no longer real-valued, but have a discrete value set.

Discrete-event systems occur naturally in the engineering practice. If the actuators like switches or valves can only jump between discrete positions the input signal is binary and the signal values 0 and 1 are associated with the closed and the open position. Sensors may indicate that a physical quantity like the liquid level in a tank or a voltage exceeds a prescribed bound. Alarm sensors are typical examples for such sensors. Besides these discrete-valued inputs and outputs also the internal state of the system may be discrete. For example, a robot gripper is empty or has grasped some part, a production step prescribed by a recipe has been carried out or not.

As the dynamical behaviour of such systems is described by the changes of the discrete signal values, which are called *events*, such systems are called discrete-event systems.

Whether or not a given dynamical system is considered as a discrete-event system depends upon the purpose of the investigations. It is typical for process supervision problems that a rather broad view on the system behaviour can be adopted which is based on a qualitative assessment of the signal values. If the supervisor should make a robot carry out a given sequence of movements or apply a certain recipe, then its decisions depend on discrete signal values like those mentioned in the examples and, hence, a discrete-event view-point is adequate. However, if signals have to remain in a narrow tolerance band, the continuous changes of these signals have to be considered and, thus, the continuous-systems point of view used in the preceding chapters is appropriate. The alternative considerations of the tank system as a continuous system or as a discrete system demonstrates the dependence of the representation level of a dynamical system from the task to be performed. Note that the terms "continuous" and "discrete" refer to the signal spaces and, hence, to the different description levels. If a distinction has to be made concerning the temporal behaviour, the notions of continuous-time and discrete-time systems will be used.

The theory of discrete-event systems has been developed rather separately with respect to the theory of continuous-variable systems. Therefore, the state of the art is different from what is known about continuous systems. The main theoretical results concern the modelling, analysis and supervisory control of discrete systems, but only a few results are available for diagnosis and fault-tolerant control. This is the reason why this chapter concerns mainly the fault diagnostic problem and presents merely preliminary results on fault accomodation and control reconfiguration. Section 8.7 shows how process supervision can be made fault-tolerant. It investigates how the diagnostic algorithm can be reconfigured after a sensor or an actuator fault has occurred. The diagnostic system reacts to such a fault by changing the model used for diagnosis and, hence, remains in operation. This is a preliminary step to the broader problem of discrete controller reconfiguration, which has not yet been considered in depth in the literature and still remains an open research problem.

In Section 8.2 models of discrete-event systems are introduced. The further investigation concerns mainly the stochastic automaton. Section 8.3 gives the solution to the state observation problem of stochastic automata, which provides the basis for the solution of the diagnostic problem given in Section 8.4. The diagnostic result can be used to reconfigure a supervisor of the system in accordance with the changes of the system dynamics brought about by the fault as described in Section 8.7.

8.2 Models of discrete-event systems

8.2.1 Deterministic and non-deterministic systems

This section explains the basic dynamical properties of discrete-event systems and shows how such systems can be described. It extends the brief introduction given in Section 3.6.

Discrete-valued signals. The discrete input, state and output of the system are denoted by the symbols v, z and w. Their discrete value sets are enumerated as follows:

$$\begin{aligned} v &\in \mathcal{N}_v = \{1, 2, ..., M\} \\ z &\in \mathcal{N}_z = \{1, 2, ..., N\} \\ w &\in \mathcal{N}_w = \{1, 2, ..., R\} \end{aligned}$$

(cf. Fig. 3.3 on p. 57). It is assumed that the numbers M, N or R are finite.

In order to use these symbolic values, in many applications the physical variables have to be encoded. For example, if a tank system with 4 on/off valves is considered, the input can be represented by a 4-vector $\boldsymbol{v}$ whose i-th component describes the position of the i-th valve. As each component can assume either the value 0 or 1, which describes the closed or the open valve, $\boldsymbol{v}$ has one of 16 different values. These 16 values are represented in the following by a scalar input symbol v with the value set $\mathcal{N}_v = \{1, 2, ..., 16\}$. To do so, a mapping from the set of the 16 values of $\boldsymbol{v}$ onto the set $\mathcal{N}_v$ has to be defined.

Every change of the symbolic value of v, z or w is called an event. For example, if the state jumps from the value j to the value i, a state event denoted by e_{ij} occurs. In Fig. 3.4 on p. 58 the events e_{13} and e_{32} are marked.

Logical behaviour. The events occur at the time instants t_k which are enumerated ($k = 0, 1, 2, ...$). In the following only the number k of the event is considered but not the actual time t_k. That is, the models used are called untimed or logical. They describe in which order the events occur but they say nothing about the temporal distance of these events.

The motivation for using untimed models is twofold. First, the basic ideas of diagnosis and fault-tolerant control of discrete-event systems can be explained by using untimed models, which are much simpler than timed models. Extensions of the methods to timed models are mentioned in the bibliographical notes. Second, in many practical circumstances, the untimed model yields the wanted results. For example, for many discrete-event systems faults can be detected due to the change of the order in which the events occur and no temporal information is necessary.

For real-time applications, a synchronisation of the results obtained for the untimed model with the given system is necessary. This problem can be easily solved by using recursive algorithms that process every measured event at the time of its occurrence. Then the time instants in which the algorithms have to carry out the next step is prescribed by the time instant in which an event is measured.

A discrete signal can be described either by the sequence of events that it generates or by the sequence of discrete values that it assumes between two successive events. The signal depicted in Fig. 3.4 generates the event sequence

$$e_{32}, e_{13}, e_{31}, e_{23}, e_{32}, e_{13},$$

but it can also be described by the sequence of discrete values

$$2, 3, 1, 3, 2, 3, 1$$

to which it jumps. In the latter representation, the given discrete value is assumed by the signal between two succeeding event times t_k where the k-th value $v(k)$ is valid for the time interval $[t_k, t_{k+1})$.

In the following, the second form of signal representation will be used. The sequence of discrete values that the input, state or output assumes for a given time horizon k_h is denoted by

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

where the sequences are elements of the product spaces of the sets $\mathcal{N}_v$, $\mathcal{N}_z$ or $\mathcal{N}_w$. For example for the input sequence $V(0\,...\,k_h)$ the relation

$$V(0\,...\,k_h) \in \mathcal{N}_v^{k_h+1} = \underbrace{\mathcal{N}_v \times \mathcal{N}_v \times ... \times \mathcal{N}_v}_{(k_h+1)-\text{times}}$$

holds. The state sequence in Fig. 3.4 is given by

$$Z(0...6) = (2, 3, 1, 3, 2, 3, 1) \in \{1, 2, 3\}^7.$$

As the input, state or output events can occur asynchronously, the counter k is defined in such a way that it counts all events independently of whether the event is generated by the input, state or output signal (left part of Fig. 8.1). Therefore, one signal may generate an event while another signal remains constant at this event time. As a consequence, the sequences V, Z and W can include repetitions of signal values like the state sequence $Z(0\,...\,3) = (1, 1, 4, 4)$.

In order to simplify the consideration, it is assumed in this chapter that the events occur synchronously. That is, the state can only change if the input changes, which, at the same time, results in a change of the output. Repetitions of the symbols indicate that no "real" event occurs but a signal remains constant.

Deterministic and non-deterministic systems. At time $k = 0$ the discrete system is in state $z(0)$ and obtains the input $v(0)$ (Fig. 3.3 and 8.1). The state jumps to the next value $z(1)$ and the output $w(1)$ is given. Under the next input $v(1)$ the system changes its state from $z(1)$ to $z(2)$ and so on. In this way, for a given initial state and input sequence $V(0\,...\,k_h)$ the state sequence $Z(0\,...\,k_h)$ and output sequence $W(0\,...\,k_h)$ are generated by the discrete-event system.

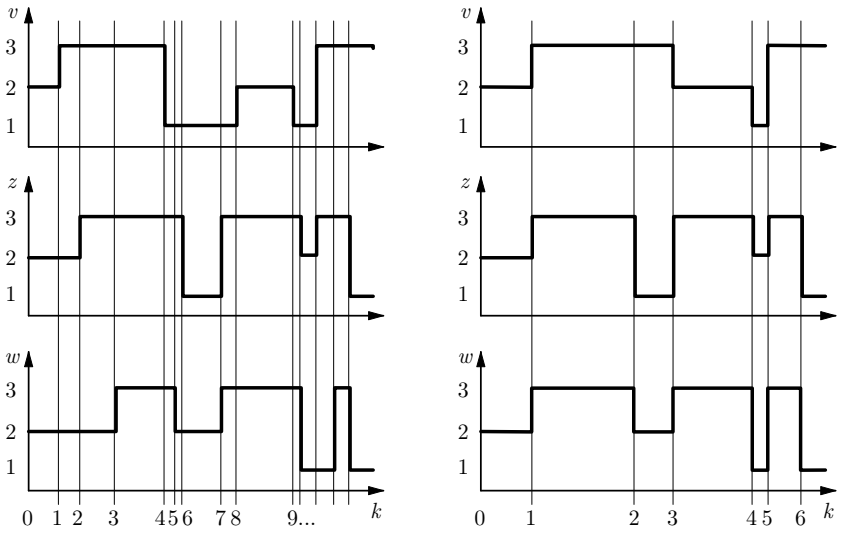


Fig. 8.1. Asynchronous (left) and synchronous (right) input, state and output sequences

For deterministic systems, the generated state and output sequences Z and W are uniquely defined by the initial state $z(0)$ and the input sequence V. A standard form for describing deterministic systems is the automaton, which will be introduced in Section 8.2.2.

For non-deterministic systems these sequences are not unique but the system may generate any sequences of the sets $\mathcal{Z}(z(0), V)$ and $\mathcal{W}(z(0), V)$. This non-determinism has to be understood in the following way. The real system under consideration has a unique performance, because it cannot assume different states at the same time. Hence, for a given initial state and a given input sequence, an unambiguous state sequence and an unambiguous output sequence occur. However, if the system is brought into the given initial state $z(0)$ for several times and gets every time the same input sequence V, then the generated state and output sequences may differ. That is, it is not possible to predict these sequences unambiguously. This is the reason why the system is called non-deterministic.

The model used to analyse this system has to describe the sets $\mathcal{Z}(z(0), V)$ and $\mathcal{W}(z(0), V)$ of possible sequences Z and W from which the real system "selects" one sequence at each time that it is brought into the initial state $z(0)$ and obtains the input sequence V. Such model have the form of, for example, non-deterministic automata or Petri nets, which will be introduced in Section 8.2.2 or of stochastic automata explained in Section 8.2.3.

8.2.2 Non-deterministic automata and Petri nets

This and the next section introduce models which can be used to describe the relation between the initial state $z(0)$ and the input sequence $V(0\,...\,k_h)$ on the one hand and the state sequence $Z(0\,...\,k_h)$ and output sequence $W(0\,...\,k_h)$ that the discrete system generates on the other hand.

Deterministic automata. The deterministic automaton

$$\mathcal{A} = (\mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, G, H, z_0)$$

has the state set $\mathcal{N}_z$, the input set $\mathcal{N}_v$ (also called the input alphabet) and the output set $\mathcal{N}_w$ (also called the output alphabet). G and H are the state transition function and the output function, which determine the successor state or output in the following way:

$$G : \mathcal{N}_z \times \mathcal{N}_v \to \mathcal{N}_z\ , \qquad z' = G(z, v) \tag{8.1}$$

$$H : \mathcal{N}_z \times \mathcal{N}_v \to \mathcal{N}_w\ , \qquad w = H(z, v)\ . \tag{8.2}$$

In this notation, z, v and w are the values of the state, input and output at time k and z' denotes the successor state occurring at time $k+1$. These symbols are used if the time k is not important for the analysis of the movement $z \to z'$ of the system, which occurs at any time k in which the system is in state z and obtains the input v.

For given initial state z_0 and input sequence $V(0\,...\,k_h)$ they yield the state sequence $Z(0...k_h+1)$ and output sequence $W(0\,...\,k_h)$ by recursive application as follows:

$$z(k+1) \;=\; G(z(k), v(k))\ , \qquad z(0) = z_0 \tag{8.3}$$

$$w(k) \;=\; H(z(k), v(k))\ . \tag{8.4}$$

Note that the state sequence has one element more than the input and output sequences. For investigations where this fact is important, the length of the sequences $V(0\,...\,k_h)$, $Z(0...k_h{+}1)$ and $W(0\,...\,k_h)$ are explicitly given, whereas otherwise the sequences are denoted merely by V, Z and W.

A nice graphical interpretation is the *automaton graph*, whose vertices depict the states $z \in \mathcal{N}_z$ and whose edges show how the state of the automaton can change (Fig. 8.2). The labels associated with the arcs describe the input v under which the state change occurs and the output w that the automaton generates while changing the state. The initial state is marked by an arrow not emerging from any other vertex. For a given initial state z_0 and input sequence V, the dynamical behaviour of the automaton is represented by the path through the automaton graph that starts in the vertex z_0 and whose edges are associated with the input prescribed by V.

It is often assumed that the function G and H are completely defined, that is, they describe $z(k+1)$ or $w(k)$, respectively, for all $z(k) \in \mathcal{N}_z$ and all $v(k) \in \mathcal{N}_v$. Then, in the automaton graph there must exist for every state z and every input v an edge from z towards another state (which is denoted by z') that is labelled by v and by some output w. In Fig. 8.2 these labels are given only for some of the edges. Note

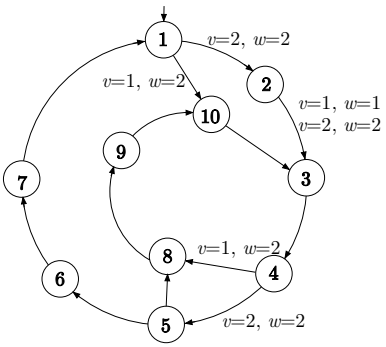


Fig. 8.2. Automaton graph of a deterministic automaton with $N = 10$, $M = 2$ and $R = 2$

that if only one edge leaves a state, this edge must be labelled with all input values $v \in \mathcal{N}_v$ like the edge $2 \rightarrow 3$ in the figure.

The automaton is deterministic because the state and the input unambiguously determine the edge and, hence, the successor state and output. For example, if the automaton in Fig. 8.2 is in state 1, depending on the input $v \in \{1, 2\}$ the automaton goes either towards state 2 or towards 10.

Non-deterministic automaton. In the non-deterministic automaton

$$\mathcal{N} = (\mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, L_n, z_0)$$

instead of the functions G and H the function

$$L_n : \mathcal{N}_z \times \mathcal{N}_w \times \mathcal{N}_z \times \mathcal{N}_v \ \rightarrow \ \{0, 1\}$$

occurs. It describes for every given state z and input v which successor state z' can be assumed while generating the output w. This function describes the behaviour of the automaton in terms of all 4-tuples (z', w, z, v) that are consistent with the automaton and form the set

$$\mathcal{R}(L_n) = \{(z', w, z, v) \ : \ L_n(z', w, z, v) = 1\} \subseteq \mathcal{N}_z \times \mathcal{N}_w \times \mathcal{N}_z \times \mathcal{N}_v \ . \quad (8.5)$$

As the set (8.5) is, mathematically, a *relation*, the function L_n is called *behavioural relation* of the non-deterministic automaton.

The description of the dynamical behaviour of a non-deterministic automaton differs from that of a deterministic automaton given by Eqs. (8.3) and (8.4). Instead of a unique successor state z' and output w, the behavioural relation L_n yields the following sets of possible successor states and output values for the current state $z(k)$ and input $v(k)$:

$$\mathcal{Z}(z(k), v(k)) = \quad \{z' \ : \ \exists w \in \mathcal{N}_w \text{ such that } L_n(z', w, z(k), v(k)) = 1\} \ , \quad (8.6)$$

$$\mathcal{W}(z(k), v(k)) = \quad \{w \ : \ \exists z' \in \mathcal{N}_z \text{ such that } L_n(z', w, z(k), v(k)) = 1\} \ . \quad (8.7)$$

Figure 8.3 depicts a part of the automaton graph of a non-deterministic automaton. It shows that for the input $v = 1$ the state may change from 1 towards 2 or towards

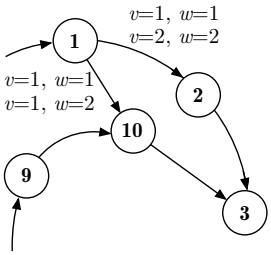


Fig. 8.3. Part of the automaton graph of a non-deterministic automaton

10, whereas for $v = 2$ only the state transition $1 \rightarrow 2$ can occur. The state change from 1 to 10 may either cause the output $w = 1$ or the output $w = 2$. Examples for the sets defined in Eqs. (8.6) and (8.7) are the following:

$$\begin{aligned} \mathcal{Z}(4,\,1) &= \{2,\,10\} \\ \mathcal{W}(1,\,2) &= \{2\}. \end{aligned}$$

If the behavioural relation L_n associates with each pair z, v a unique successor state z' and output w it can be represented by a state transition function G and an output function H and the non-deterministic automaton becomes deterministic.

Markov property. Both the deterministic and the non-deterministic automaton possess an important property, which is referred to as the *Markov property*. This property means that the successor state $z(k+1)$ does only depend upon the current state $z(k)$ and the current input $v(k)$. It does not depend on the whole state sequence $Z(0...k)$ or the whole input sequence $V(0...k)$ that the automaton has generated or obtained until time k, but only on the state and input at time k. The Markov property makes it possible to find the recursive relations (8.3) and (8.6) both of which are a relation between the next state $z(k+1)$ and the current state $z(k)$ and input $v(k)$ only. In case of the non-deterministic automaton, this relation does not determine the future state unambiguously, but it fixes the set of future states. Note that the set $\mathcal{Z}(z(k), v(k))$ given in Eq. (8.6) depends merely upon $z(k)$ and $v(k)$.

In applications, the question whether or not the system under investigation possesses the Markov property depends upon the definition of the state z. Roughly speaking, if the state z includes all the information about the signals up to time k which is necessary to determine the future behaviour of the system, then the future state can be represented only in terms of the current state $z(k)$ and there is no need to refer to earlier states or input values occuring in the sequences $Z(0...k-1)$ or $V(0...k-1)$. To define the state appropriately is an important modelling step.

Other models of non-deterministic systems. Automata are an important means for describing discrete-event systems, but they are not the only form of discrete models. The behavioural relation L_n can be represented in different ways. For example, it is possible to describe the fact

$$(z(k+1)=z',\ w(k)=w,\ z(k)=z,\ v(k)=v) \in \mathcal{R}(L_n)$$

by logic formulae like

$$(z(k)=z) \wedge (v(k)=v) \Rightarrow (w(k)=w) \wedge (z(k+1)=z')\ ,$$

where the expressions in the parentheses are logic assertions. Another possibility is to describe this relation by a Petri net. In the example net given in Fig. 8.4, circular vertices denote places and black boxes transitions. The dynamics of the net is described in terms of the movement of the tokens. In the figure, the places 1, 6 and 7 are marked by one token each, which are represented by black bullets. A transition is enabled to switch if all predecessor places are marked like the transitions 1, 2 and 5 in the figure. If a transition switches, the tokens move from the predecessor places through the transition and mark all successor places. The "firing" of a transition represents an event.

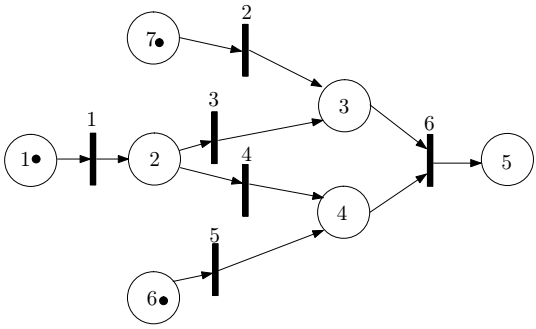


Fig. 8.4. A Petri net

A non-deterministic behaviour occurs if, in the example, place 2 is marked after transition 1 has fired and, hence, both the transitions 3 and 4 are enabled. Only one transition may really switch, but the model does not fix which of the two transitions is selected.

Petri nets provide a more concise representation of non-deterministic systems if the non-determinism refers to the fact that particular events may occur in different order. In the figure the transitions 2 and 5 are both enabled and the Petri net does not describe in which order these transitions will fire. Either transition 2 or transition 5 switches first. This becomes intuitively clear from the graphical representation.

However, Petri nets do not introduce fundamentally new phenomena compared to automata. In fact, they can be transformed into automata with precisely the same dynamical behaviour. Therefore, automata are used in the following to describe discrete-event systems.

8.2.3 Stochastic processes and automata

The stochastic automata introduced in this section extend the description of non-deterministic discrete-event systems in such a way that the frequency of the occurrence of the different events can be assessed. They provide very useful additional information, because non-deterministic systems often produce a large set of different state and output sequences, but in practice these sets do by no means occur with the same frequency. In many applications there are numerous sequences that cannot be completely ignored and have to be described by the non-deterministic automaton in order to meet the requirement that the model can generate all state and output sequences that the system under consideration may generate. However, they occur very seldom and this additional information should also be provided by the model.

The following explains how non-deterministic automata can be associated with the probabilities that the different sequences occur. First, the notion of a stochastic process has to be explained.

Stochastic processes. As discussed in Section 8.2.1, technological systems generate, for a given initial state and input sequence, unambiguous state and output sequences. However, at any time that such a system is brought into the given initial state and obtains the given input sequence (which is also referred to as an "experiment" in probability theory), the generated state and output sequences (also called "realisations") may be different from the sequences generated before. Therefore, any model can only predict some sets $\mathcal{Z}(z_0, V)$ and $\mathcal{W}(z_0, V)$ of possible state and output sequences rather than the actual state and output sequence. This is why the system is called non-deterministic.

A stochastic process is a non-deterministic system for which the state and output sequences are generated with a certain probability. Its non-determinism is not only considered in terms of the sets $\mathcal{Z}(z_0, V)$ and $\mathcal{W}(z_0, V)$ but also in terms of the frequency with which the different elements of these sets are generated. Some of them may occur very often whereas others occur rarely.

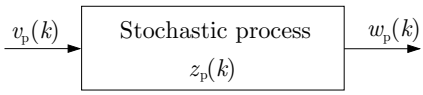


Fig. 8.5. Stochastic process

The symbols v_p, z_p and w_p are used to denote the random variables of the input, state and output of the stochastic process. They are stochastic variables, that is, variables whose values are determined by chance. In every experiment these variables assume values from the sets $\mathcal{N}_v$, $\mathcal{N}_z$ or $\mathcal{N}_w$, respectively (Fig. 8.5). As the ranges of these variables and the time k are discrete, the process is called more precisely a *discrete stochastic process*.

The stochastic automaton should describe the probability with which the different sequences occur for a given stochastic process. To do so, it has to represent

the generation law underlying the stochastic process. Before formally introducing the stochastic automaton, the generation law of the considered class of stochastic processes has to be characterised.

A discrete stochastic process is defined by the probability with which a certain state change appears and an output symbol occurs at time k for a given sequence of input symbols. In general this probability depends on the sequence of states up to time k and is thus described by the conditional probability

$$\mathrm{Prob}\left(\begin{array}{c|l} z_p(k+1)=z(k+1), & z_p(0)=z(0),\ ...,\ z_p(k)=z(k), \\ w_p(k)=w(k) & v_p(0)=v(0),\ ...,\ v_p(k)=v(k) \end{array}\right).$$

In the following only those stochastic processes are considered that pocess the *Markov property*. For such processes the relation

$$\begin{aligned} &\mathrm{Prob}(z_p(k+1)=z(k+1), w_p(k)=w(k) \mid z_p(k)=z(k), v_p(k)=v(k)) \\ &= \mathrm{Prob}\left(\begin{array}{c|l} z_p(k+1)=z(k+1), & z_p(0)=z(0),\ ...,\ z_p(k)=z(k), \\ w_p(k)=w(k) & v_p(0)=v(0),\ ...,\ v_p(k)=v(k) \end{array}\right) \end{aligned} \tag{8.8}$$

holds for all k, $z(k+1), z(k)$, ..., $z(0)$, $w(k)$, $w(k-1)$, ..., $w(0)$ and $v(k)$, $v(k-1)$,...,$v(0)$. It is common to say that $z(k+1)$ and $w(k)$ are conditionally independent of the past variables for given $z(k)$ and $v(k)$. Then, only the transition probability

$$\mathrm{Prob}(z_p(k+1)=z(k+1), w_p(k)=w(k) \mid z_p(k)=z(k), v_p(k)=v(k))$$

has to be known, whose conditional part has the same "length" for all k.

Furthermore, it is assumed that the process is homogeneous which means that the transition probability does not explicitly depend on the time variable k. Whenever the 4-tuple of successor state z', output w, current state z and input v is considered, the transition probability is the same. Hence, the relation

$$\begin{aligned} &\mathrm{Prob}(z_p(k+1)=z', w_p(k)=w \mid z_p(k)=z, v_p(k)=v) \\ &\quad = \mathrm{Prob}(z_p(1)=z', w_p(0)=w \mid z_p(0)=z, v_p(0)=v) \end{aligned} \tag{8.9}$$

holds for all k. A discrete stochastic process whose generation law is described by the state transition probability (8.9) is called *homogenous Markov process* with input and output.

For describing the stochastic process by a stochastic automaton it is assumed that the appearance of a certain input symbol at time k is independent of the states and of the input values that have occurred up to that time, and independent of the time k. Therefore, the relation

$$\begin{aligned} &\mathrm{Prob}\left(v_p(k)=v(k) \,\middle|\, \begin{array}{l} z_p(0)=z(0),\ ...,\ z_p(k)=z(k), \\ v_p(0)=v(0),\ ...,\ v_p(k-1)=v(k-1) \end{array}\right) \\ &\quad = \mathrm{Prob}(v_p=v) \end{aligned} \tag{8.10}$$

holds, where $\mathrm{Prob}(v_p=v)$ describes the probability with which the input symbol v occurs. This assumption is not satisfied, for example, if the input $v(k)$ is prescribed

by a supervisor that acts in dependence upon the measured output $w(k)$. Then this "feedback" needs to be included into the stochastic description of the process in order to satisfy the assumption (8.10) which will be used in the following.

The automaton is set up under the assumption that all input symbols v can occur:

$$\mathrm{Prob}(v_p = v) > 0 \quad \text{for all} \quad v \in \mathcal{N}_v. \tag{8.11}$$

Otherwise input symbols with vanishing probability are removed from the set $\mathcal{N}_v$.

Representation of stochastic processes by stochastic automata. Homogeneous Markov processes with finite sets of input values, output values and states are represented by finite-state stochastic automata. An initialised stochastic automaton is given by the 5-tuple

$$\mathcal{S} = (\mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, L, \mathrm{Prob}(z(0))) \tag{8.12}$$

with $\mathcal{N}_z, \mathcal{N}_v$ and $\mathcal{N}_w$ defined above. $\mathrm{Prob}(z(0))$ denotes the initial state probability distribution which is the set of the N probability values $\mathrm{Prob}(z(0) = z_0)$ for $z_0 \in \mathcal{N}_z$. L is a function

$$L : \mathcal{N}_z \times \mathcal{N}_w \times \mathcal{N}_z \times \mathcal{N}_v \;\longrightarrow\; [0,1] \tag{8.13}$$

that represents the generation law governing the stochastic Markov process:

$$L(z', w, z, v) = \mathrm{Prob}(z_p(1) = z',\ w_p(0) = w \mid z_p(0) = z,\ v_p(0) = v)\ . \tag{8.14}$$

In extension of the terminology used for the non-deterministic automaton, the function L is called *behavioural relation* of the stochastic automaton. In order to indicate that the behavioural relation is a conditional probability distribution, the symbol $L(z', w \mid z, v)$ is used instead of $L(z', w, z, v)$. The notion of the probability distribution refers to the mapping $L(z', w \mid z, v)$ for all $z, z' \in \mathcal{N}_z$, $v \in \mathcal{N}_v$ and $w \in \mathcal{N}_w$.

The behavioural relation has the properties

$$\begin{aligned} 0 \le L(z', w \mid z, v) &\le 1\,, \quad \forall z', z \in \mathcal{N}_z,\ v \in \mathcal{N}_v,\ w \in \mathcal{N}_w \\ \sum_{z' \in \mathcal{N}_z} \sum_{w \in \mathcal{N}_w} L(z', w \mid z, v) &= 1\,, \quad \forall z \in \mathcal{N}_z,\ v \in \mathcal{N}_v \end{aligned} \tag{8.15}$$

and the two boundary distributions

$$G(z' \mid z, v) = \sum_{w \in \mathcal{N}_w} L(z', w \mid z, v) \tag{8.16}$$

$$H(w \mid z, v) = \sum_{z' \in \mathcal{N}_z} L(z', w \mid z, v)\,. \tag{8.17}$$

$G(z'|z, v)$ is called the *state transition relation* and $H(w|z, v)$ the *output relation* of the stochastic automaton. Due to Eq. (8.14) these relations represent the conditional probability distributions

$$G(z' \mid z, v) = \mathrm{Prob}(z_p(1) = z' \mid z_p(0) = z, v_p(0) = v) \tag{8.18}$$

$$H(w \mid z, v) = \mathrm{Prob}(w_p(0) = w \mid z_p(0) = z, v_p(0) = v) \tag{8.19}$$

and possess the properties

$$\sum_{z' \in \mathcal{N}_z} G(z' \mid z, v) = 1 \tag{8.20}$$

$$\sum_{w \in \mathcal{N}_w} H(w \mid z, v) = 1\,. \tag{8.21}$$

These conditional probability distributions replace the state transition function G and the output function H of the deterministic automaton used in Eqs. (8.1) and (8.2).

Note that the functions G and H defined in Eq. (8.18) and (8.19) include less information than the behavioural relation L because L also reflects the stochastic dependence of z' and w. Therefore, the following investigations refer to L rather than G and H. G is useful for problems in which only the state sequence is considered and the output sequence ignored. Stochastic automata for which the behavioural relation L can be reconstructed from G and H are called stochastic *Mealy automata*. For them the relation

$$L(z', w \mid z, v) = G(z' \mid z, v) \cdot H(w \mid z, v)$$

holds for all $z, z' \in \mathcal{N}_z$, $v \in \mathcal{N}_v$ and $w \in \mathcal{N}_w$.

Autonomous stochastic automaton. If the automaton has no input, it is called an autonomous automaton. If, furthermore, the output coincides with the state, this automaton is given by the triple

$$\mathcal{S}_a = (\mathcal{N}_z, G, \mathrm{Prob}(z(0))),$$

where G denotes the state transition relation given by Eq. (8.18) after neglecting the input v:

$$G(z' \mid z) = \mathrm{Prob}(z_p(1) = z' \mid z_p(0) = z).$$

Graphical interpretation of stochastic automata. Consider first the autonomous automaton, which can be interpreted as a directed graph, whose vertices denote the states and whose edges denote the possible state transitions. Figure 8.6 gives an example. The edges are associated with the state transition probability given by the value of the state transition relation G for the pair of states connected by the edges. These probabilities are shown in the figure only for some edges.

Beginning in a state z_0 with $\mathrm{Prob}(z(0) = z_0) > 0$ the automaton moves along the directed edges according to the corresponding probabilities. If more than one edge starts in a state then all these edges can be followed which results in alternative state and output sequences. The frequencies with which the automaton follows these edges is described by the transition probabilities.

As will be described in detail below, the probability that the automaton is at time k in a given state z can be obtained by multiplying the state probabilities at time

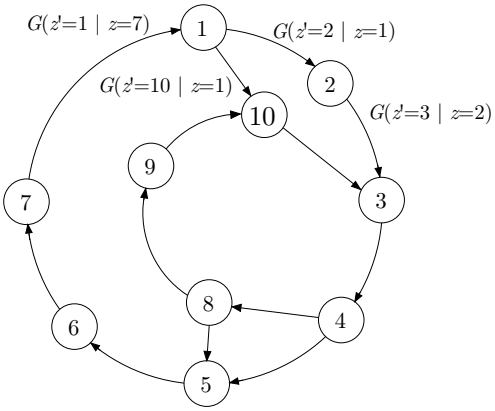


Fig. 8.6. Autonomous stochastic automaton

$k-1$ with the transition probabilities. If more than one "way" leads to the state z, then all these probabilities have to be summed up.

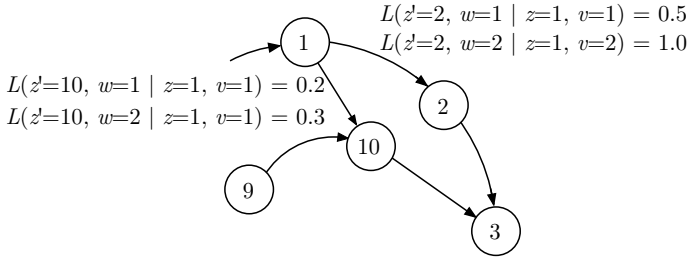


Fig. 8.7. Part of the automaton graph of a stochastic automaton with input and output

For automata with input and output, the state transition probability is replaced by the corresponding value of the behavioural relation L (Fig. 8.7). The movement through the automaton graph depends on the input. Moreover, the probability that a certain output is generated depends on the pair of states involved in the movement and the input obtained.

Example 8.1 *Properties of a stochastic automaton*

In the example shown in Fig. 8.7, the automaton may step from state 1 towards state 2 or 10 if it obtains input $v=1$, but it is known to step towards state 2 if the input $v=2$ is applied. Moreover, the automaton may produce either output $w=1$ or $w=2$. The behavioural relation says that the probability to step from state 1 towards 10 when getting the input $v=1$ is 0.2 if this step is associated with the output $w=1$ and 0.3 if $w=2$ occurs. The sum of 0.5 of both values is the probability that the automaton steps from 1 to 10 under the input $v=1$ while producing *some* output. Hence, $G(10\,|\,1,1) = 0.5$ holds. Alternatively, the automaton may step from 1 towards 2. It definitely does this step if it obtains input $v=2$ and it is known to

produce output $w=2$ during this step. If the automaton gets input $v=1$, then the probability to step to state 2 while generating output $w=1$ is 0.5.

The property (8.15) of the behavioural relation is satisfied, because for $z=1$ and $v=1$ the examples yields

$$\begin{aligned}\sum_{z'}\sum_{w} L(z', w \mid z=1, v=1) = \\ = L(z'=10, w=1 \mid z=1, v=1) + L(z'=10, w=2 \mid z=1, v=1) + \\ + L(z'=2, w=1 \mid z=1, v=1) \\ = 0.2 + 0.3 + 0.5 = 1\end{aligned}$$

and for $z=1$ and $v=2$

$$\sum_{z'\in\mathcal{N}_z}\sum_{w\in\mathcal{N}_w} L(z', w \mid z=1, v=2) \;=\; L(z'=2, w=2 \mid z=1, v=2) \;=\; 1\ .$$

The state transition relation G defined in Eq. (8.16) ignores the output and considers merely the transition between the states in dependence upon the input. For the example the following relations hold:

$$\begin{aligned} G(z'=10 \mid z=1, v=1) &= 0.2 + 0.3 = 0.5 \\ G(z'=2 \mid z=1, v=1) &= 0.5 \\ G(z'=2 \mid z=1, v=2) &= 1. \end{aligned}$$

They say that for the input $v=1$ the automaton goes from state 1 to state 2 or 10 with probability 0.5, but if it obtains input $v=2$ the automaton is known to go towards state 2.

The output relation H defined in Eq. (8.17) says which output is produced independently of the next state that is assumed by the automaton. For the example

$$\begin{aligned} H(w=1 \mid z=1, v=1) &= 0.7 \\ H(w=2 \mid z=1, v=1) &= 0.3 \\ H(w=2 \mid z=1, v=2) &= 1 \end{aligned}$$

hold, which means that the automaton is known to produce the output $w=2$ if it obtains the input $v=2$ in state 1, but for the input $v=1$ it may generate the output $w=1$ with probability 0.7 and $w=2$ with probability 0.3. Note that there is in general no way to reconstruct L from given G and H as mentioned above. □

8.2.4 Behaviour of stochastic automata

This section describes the movement of stochastic automata for given initial state probability distribution and input sequences. The determination of the state and output sequences that the automata generate with positive probability is called *simulation*. The results obtained are the basis for defining and investigating the observability and diagnosability of stochastic automata in the next sections.

State and output sequences generated by the stochastic automaton. Like a nondeterministic automaton, the stochastic automaton is not in a unique state $z(k)$ but the state at time k is given as a set of possible states. This set is described by a discrete probability distribution denoted by $\mathrm{Prob}(z(k))$:

$$\mathrm{Prob}(z(k)) = \{\mathrm{Prob}(z_p(k)=1), \mathrm{Prob}(z_p(k)=2), ..., \mathrm{Prob}(z_p(k)=N)\} \ . \quad (8.22)$$

This notation of the probability distribution $\mathrm{Prob}(z(k))$ means that all N possible values of the random variable $z_p(k)$ are considered and their probability are the elements of a set or, alternatively, of a vector. Hence, the probability distribution $\mathrm{Prob}(z(k))$ is a function associating to all N different states $z \in \mathcal{N}_z$ the probability with which the automaton assumes the state z at time k. Hence, $\mathrm{Prob}(z_p(k) = z)$ has the N elements given on the right-hand side of Eq. (8.22).

The dynamical behaviour depends upon the initial state z_0 and the input sequence V. For the initial state a probability distribution $\mathrm{Prob}(z(0))$ is given, which means, according to the above convention, that the value of $\mathrm{Prob}(z_p(0) = z)$ is available for all N different states $z \in \mathcal{N}_z$. The state at time k_h depends also upon the input sequence with length k_h

$$V(0 \,...\, k_h - 1) = (v(0), ..., v(k_h - 1)) \in \mathcal{N}_v^{k_h} = \underbrace{\mathcal{N}_v \times \mathcal{N}_v \times ... \times \mathcal{N}_v}_{k_h - \text{times}} , \quad (8.23)$$

where k_h is the time horizon. According to Eq. (8.10) the probability with which a certain input sequence occurs is given by the product of the probabilities with which the input values included occur:

$$\begin{aligned} &\mathrm{Prob}(V_p(0 \,...\, k_h - 1) = V(0 \,...\, k_h - 1)) \\ &\quad = \mathrm{Prob}(v_p(0) = v(0), v_p(1) = v(1), ..., v_p(k_h - 1) = v(k_h - 1)) \\ &\quad = \mathrm{Prob}(v_p = v(0)) \cdot \mathrm{Prob}(v_p = v(1)) \cdot ... \cdot \mathrm{Prob}(v_p = v(h_h - 1)) \ . \end{aligned}$$

For given input sequence, the initialised stochastic automaton yields a probability distribution over the set of all state sequences of length $k_h + 1$

$$Z(0 \,...\, k_h) = (z(0),\ z(1),\ ...,\ z(k_h)) \in \mathcal{N}_z^{k_h+1}$$

as follows:

$$\begin{aligned} &\mathrm{Prob}(Z(0 \,...\, k_h) \mid V(0 \,...\, k_h - 1)) \quad (8.24) \\ &\quad = \mathrm{Prob}(z(0),\ z(1),\ ...,\ z(k_h) \mid v(0), ..., v(k_h - 1)) \\ &\quad = G(z(k_h) \mid z(k_h - 1), v(k_h - 1)) \cdot\ G(z(k_h - 1) \mid z(k_h - 2), v(k_h - 2)) \cdot \\ &\qquad \cdot ... \cdot\ G(z(1) \mid z(0), v(0)) \cdot \mathrm{Prob}(z(0)) \\ &\quad = \prod_{k=0}^{k_h - 1} G(z(k+1) \mid z(k), v(k)) \cdot \mathrm{Prob}(z(0)) \ . \end{aligned}$$

This equation describes how the probability distribution

$$\mathrm{Prob}(Z(0 \,...\, k_h) \mid V(0 \,...\, k_h - 1))$$

is obtained for given $V(0 \,...\, k_h)$. Similar to Eq. (8.22), it can be used for all different state and input sequences $Z(0 \,...\, k_h)$ and $V(0 \,...\, k_h - 1)$. The distribution

$$\mathrm{Prob}(Z(0 \,...\, k_h) \mid V(0 \,...\, k_h - 1))$$

consists of $N^{k_h+1} \cdot M^{k_h}$ different values, which are obtained separately by Eq. (8.24).

According to this equation, the probability that a given state sequence

$$Z(0\,...\,k_h) = (z(0),\, z(1),\, ...,\, z(k_h))$$

occurs under the assumption that the automaton obtains the input sequence

$$V(0\,...\,k_h - 1) = (v(0),\, ...,\, v(k_h - 1))$$

is given by the product of the state transition function G whose arguments correspond to the considered state sequence and the input. For each entry of the probability distribution, k_h values of the function G determine together with the value of $\mathrm{Prob}(z(0))$ the resulting entry of

$$\mathrm{Prob}(Z(0\,...\,k_h) \mid V(0\,...\,k_h - 1)).$$

The application of Eq. (8.24) will be referred to as the *simulation* of the stochastic process or of the automaton, because it shows with which probability the system under consideration follows the state sequence $Z(0\,...\,k_h)$ for given input sequence $V(0\,...\,k_h - 1)$.

To simplify the notation the abbreviation

$$G(k) ::= G(z(k+1) \mid z(k), v(k))$$

will be used in the following if it is clear from the context for which argument $z(k+1), z(k)$ and $v(k)$ the function is applied. Then, Eq. (8.24) simplifies to

$$\mathrm{Prob}(Z(0\,...\,k_h) \mid V(0\,...\,k_h - 1)) = \prod_{k=0}^{k_h-1} G(k) \cdot \mathrm{Prob}(z(0))\ . \quad (8.25)$$

The state of the stochastic automaton at time k_h is described by the boundary distribution of (8.25):

$$\begin{aligned} &\mathrm{Prob}(z(k_h) \mid V(0\,...\,k_h - 1)) \qquad (8.26)\\ &= \sum_{z(0)\in\mathcal{N}_z} \sum_{z(1)\in\mathcal{N}_z} \cdots \sum_{z(k_h-1)\in\mathcal{N}_z} \mathrm{Prob}(Z(0\,...\,k_h) \mid V(0\,...\,k_h - 1))\\ &= \sum_{z(0)\in\mathcal{N}_z} \sum_{z(1)\in\mathcal{N}_z} \cdots \sum_{z(k_h-1)\in\mathcal{N}_z} G(k_h-1)\cdot G(k_h-2)\cdot ...\cdot G(0)\cdot \mathrm{Prob}(z(0))\\ &= \sum_{Z(0\,...\,k_h-1)} \prod_{k=0}^{k_h-1} G(k)\cdot \mathrm{Prob}(z(0))\ . \end{aligned}$$

The sum in the last line of the above equation means the sum over all state sequences of the length k_h: $Z(0\,...\,k_h - 1) \in \mathcal{N}_z^{k_h}$. In the following all sums are abbreviated likewise. Eq. (8.26) can be written in the recursive form

$$\begin{aligned} k_h > 1: \qquad &\mathrm{Prob}(z(k_h) \mid V(0\,...\,k_h - 1)) \qquad (8.27)\\ &= \sum_{z(k_h-1)} G(k_h-1)\cdot \mathrm{Prob}(z(k_h-1) \mid V(0\,...\,k_h-2)) \end{aligned}$$

$$k_h = 1: \qquad \mathrm{Prob}(z(1) \mid v(0)) = \sum_{z(0)} G(0)\cdot \mathrm{Prob}(z(0))\ , \qquad (8.28)$$

where the Markov property (8.8) and the independence (8.10) between the current state and the current input have been used. Note that in Eq. (8.28) the simplified notation $V(0\,...\,0) = V(0)$ has been used. Note that

$$\text{Prob}(z(k_h) \mid V(0\,...\,k_h-1)) = \text{Prob}(z(k_h) \mid V(0\,...\,k_h)) \tag{8.29}$$

holds because according to Eq. (8.10) the current state $z(k_h)$ does not depend on the current input $v(k_h)$. Similarly, for the output sequence the relation

$$\begin{aligned} &\text{Prob}(W(0\,...\,k_h) \mid V(0\,...\,k_h)) \qquad (8.30)\\ &= \sum\nolimits_{z(0)} \text{Prob}(W(0\,...\,k_h) \mid V(0\,...\,k_h), z(0)) \cdot \text{Prob}(z(0)) \\ &= \sum\nolimits_{Z(0\ldots k_h+1)} \prod\nolimits_{k=0}^{k_h} L(k) \cdot \text{Prob}(z(0)) \end{aligned}$$

with the abbreviation

$$L(k) := L(z(k+1), w(k) \mid z(k), v(k)) \tag{8.31}$$

is obtained. As above, this equation can be transformed into the recursive relations

$$\begin{aligned} k_h > 0: \quad &\text{Prob}(z(k_h+1), W(0\,...\,k_h) \mid V(0\,...\,k_h)) \qquad (8.32)\\ &= \sum\nolimits_{z(k_h)} L(k_h) \cdot \text{Prob}(z(k_h), W(0\,...\,k_h-1) \mid V(0\,...\,k_h-1)) \end{aligned}$$

$$k_h = 0: \quad \text{Prob}(z(1), w(0) \mid v(0)) = \sum\nolimits_{z(0)} L(0) \cdot \text{Prob}(z(0)) \tag{8.33}$$

which yield

$$\begin{aligned} &\text{Prob}(W(0\,...\,k_h) \mid V(0\,...\,k_h)) \qquad (8.34)\\ &= \sum\nolimits_{z(k_h+1)} \text{Prob}(z(k_h+1), W(0\,...\,k_h) \mid V(0\,...\,k_h)) \,. \end{aligned}$$

Note that in the recursion (8.32), (8.33) the joint probability distribution of the next state $z(k_h+1)$ and the output sequence up to time k_h has to be determined. The probability of the output sequence is then obtained as the boundary probability by Eq. (8.34).

Behaviour of the stochastic automaton. The behaviour of a dynamical system is the set of all input and output sequences that are consistent with the system dynamics. For the stochastic automaton a pair consisting of the input sequence V and output sequence W belongs to the behaviour if there exists a state sequence

$$Z(0...k_h+1) = (z(0),\, z(1),\, z(2),\, ...,\, z(k_h+1))$$

such that the two conditions

$$\text{Prob}(z_p(0) = z(0)) > 0 \quad \text{and} \quad \prod\nolimits_{k=0}^{k_h} L(k) > 0 \tag{8.35}$$

are satisfied:

$$\begin{aligned} \mathcal{B}(k_h) \quad = \quad &\{((v(0),\, v(1),\, ...,\, v(k_h)),\, (w(0),\, w(1),\, ...,\, w(k_h))\ : \qquad (8.36)\\ &\quad \exists Z(0\,...\,k_h+1) = (z(0),\, z(1),\, ...,\, z(k_h+1)) \text{ with} \\ &\quad \text{Prob}(z_p(0) = z(0)) > 0,\ L(k) > 0 \text{ for } k = 0,\, 1,\, ...,\, k_h-1 \ \} \,. \end{aligned}$$

Note that Eq. (8.36) uses the assumption (8.11). In the further investigations the time horizon k_h is used as a steadily increasing variable. The dependence of $\mathcal{B}$ upon k_h is mentioned explicitly. Note that

$$\begin{aligned}\mathrm{Prob}(W(0\,...\,k_h), V(0\,...\,k_h)) \quad &= \quad \mathrm{Prob}(W(0\,...\,k_h) \mid V(0\,...\,k_h)) \cdot \\ &\quad \cdot \mathrm{Prob}(V(0\,...\,k_h)),\end{aligned}$$

where $\mathrm{Prob}(W(0\,...\,k_h) \mid V(0\,...\,k_h))$ given by Eq. (8.30) describes the probability with which the element $(V(0\,...\,k_h), W(0\,...\,k_h))$ of the set $\mathcal{B}(k_h)$ occurs. As $\mathrm{Prob}(V(0\,...\,k_h)) > 0$ holds, the behaviour of the automaton can also be written as

$$\mathcal{B}(k_h) = \{(V(0\,...\,k_h), W(0\,...\,k_h)) : \mathrm{Prob}(W(0\,...\,k_h) \mid (V(0\,...\,k_h)) > 0\}\ . \tag{8.37}$$

Consistency of input/output pairs. Until now, the automaton has been used to predict the behaviour of the stochastic process for given input sequence. In state observation and diagnosis the automaton will be used to check whether measured sequences of input and output values "belong" to the automaton. The pair of input and output sequences $(V(0\,...\,k_h), W(0\,...\,k_h))$ is called an *input/output (I/O) pair*. It is checked whether there exists an initial state such that the automaton generates the given output sequence for the given input sequence with non-vanishing probability. I/O pairs are called *consistent* with the stochastic automaton if they belong to the behaviour $\mathcal{B}(k_h)$ described by Eq. (8.37):

$$(V, W) \in \mathcal{B}.$$

The investigation of whether an I/O pair is consistent or not with a stochastic automaton will be the basis of the next sections.

8.2.5 Model of the faulty automaton

In order to describe the behaviour of a discrete-event system under the influence of faults, the fault $f(k)$ is introduced as an additional input (Fig. 8.8). The fault may change over time. Hence, it is described by the sequence

$$F(0\,...\,k_h) = (f(0), f(1), \ ..., \ f(k_h)).$$

The additional input f extends the stochastic automaton, which now becomes

$$\mathcal{S} = (\mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_f, \mathcal{N}_w, L, \mathrm{Prob}(z(0))) \tag{8.38}$$

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

$$\begin{aligned}&L : \mathcal{N}_z \times \mathcal{N}_w \times \mathcal{N}_z \times \mathcal{N}_f \times \mathcal{N}_v \ \longrightarrow \ [0,1] \\ &L(z', w \mid z, f, v) \\ &\quad = \mathrm{Prob}(z_p(1) = z', w_p(0) = w \mid z_p(0) = z, f_p(0) = f, v_p(0) = v)\ ,\end{aligned} \tag{8.39}$$

where f_p symbolises the stochastic variable for the current fault.

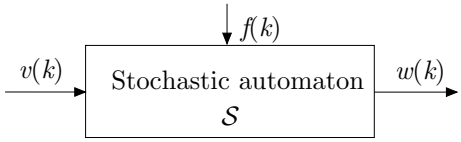


Fig. 8.8. Stochastic automaton subject to some fault

The model (8.38) is applicable for all fault sequences $F(0 \ldots k_h) \in \mathcal{N}_f^{k_h+1}$. Sequences of this kind describe, among others, a fault behaviour in which the fault changes without interruption from any time step to the next one. Such a fault behaviour is, in general, not realistic. In many technical processes the fault is known to be slowly changing and the faults do not switch from one to another without a faultless period in between. This information has to be used during the diagnosis.

In order to include such an information into the diagnosis, the fault is assumed to be the output of another stochastic automaton

$$\mathcal{S}_f = (\mathcal{N}_f, G_f, \mathrm{Prob}(f(0))) \tag{8.40}$$

which is called the *fault model* (Fig. 8.9). G_f is the state transition relation of the fault model, which describes the conditional probability that the fault changes from f towards f' within one time step:

$$\begin{aligned} &G_f : \mathcal{N}_f \times \mathcal{N}_f \longrightarrow [0,1] \\ &G_f(f' \mid f) = \mathrm{Prob}(f_p(1) = f' \mid f_p(0) = f) \ . \end{aligned} \tag{8.41}$$

If the time steps are equidistant (like in discrete-time systems) these probabilities are closely related to the well known measure of mean time before failure. $\mathrm{Prob}(f(0))$ denotes the probability distribution over the initial fault set.

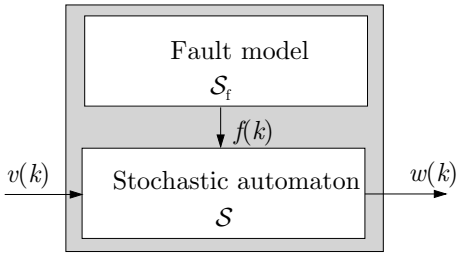


Fig. 8.9. Stochastic automaton with fault model

The combination of the stochastic automaton with the fault model depicted in Fig. 8.9 is a stochastic automaton

$$\tilde{\mathcal{S}} = (\mathcal{N}_{\tilde{z}}, \mathcal{N}_v, \mathcal{N}_w, \tilde{L}, \mathrm{Prob}(\tilde{z}(0))) \tag{8.42}$$

whose state set is given by

$$\mathcal{N}_{\tilde{z}} = \mathcal{N}_z \times \mathcal{N}_f \tag{8.43}$$

and whose behavioural relation $\tilde{L}$ is obtained from L and G_f according to

$$\tilde{L}(z', f', w \mid z, f, v) = L(z', w \mid z, f, v) \cdot G_f(f' \mid f) \tag{8.44}$$

with $z, z' \in \mathcal{N}_z$, $v \in \mathcal{N}_v$, $w \in \mathcal{N}_w$ and $f, f' \in \mathcal{N}_f$. If the elements $\tilde{z} \in \mathcal{N}_{\tilde{z}}$ are written as the vector

$$\tilde{z} = \begin{pmatrix} z \\ f \end{pmatrix}, \tag{8.45}$$

the behavioural relation $\tilde{L}$ in Eq. (8.44) can be rewritten as

$$\tilde{L} : \mathcal{N}_{\tilde{z}} \times \mathcal{N}_w \times \mathcal{N}_{\tilde{z}} \times \mathcal{N}_v \longrightarrow [0,1] \tag{8.46}$$

$$\begin{aligned} &\tilde{L}(\tilde{z}', w \mid \tilde{z}, v) \\ &\quad = \mathrm{Prob}(\tilde{z}_p(1) = \tilde{z}', w_p(0) = w \mid \tilde{z}_p(0) = \tilde{z}, v_p(0) = v) \\ &\quad = \mathrm{Prob}(\begin{pmatrix} z_p(1) = z' \\ f_p(1) = f' \end{pmatrix}, w_p(0) = w \mid \begin{pmatrix} z_p(0) = z \\ f_p(0) = f \end{pmatrix}, v_p(0) = v) \end{aligned}$$

which gives $\tilde{L}$ the standard form.

Remark 8.1 *More general fault model*
It should be mentioned here that the structure depicted in Fig. 8.9 is not the most general one, because it does not allow that the fault depends on the input and the state. In applications the occurrence of a fault may be made more feasible if, for example, a given system is used under maximum load conditions. Then, the probability of the fault occurrence depends also on v and z. For such systems, an information link from the fault model $\mathcal{S}$ towards the fault model has to be inserted into Fig. 8.9 and the behavioural relation $\tilde{L}$ can no longer be decomposed according to Eq. (8.44). However, in the following the simpler scheme depicted in Fig. 8.9 will be considered. □

8.3 State observation of stochastic automata

8.3.1 Preliminary considerations of consistency-based diagnosis

Consistency-based diagnosis is based on the investigation whether an I/O pair

$$(V(0 \ldots k_h), W(0 \ldots k_h))$$

measured for a system under investigation is consistant with a stochastic automaton that describes the faultless or the faulty system. Consistency means that there exists a state sequence $Z(0 \ldots k_h+1)$ that occurs for the measured input sequence $V(0 \ldots k_h)$ with positive probability and for which the output sequence $W(0 \ldots k_h)$ occurs with positive probability.

To define this notion formally it is assumed that both measured sequences

$$V(0 \ldots k_h) \quad \text{and} \quad W(0 \ldots k_h)$$

have the same length k_h.

Definition 8.1 (Consistent I/O pair) *An I/O pair* (V, W) *is called* consistent *with the stochastic automaton* $\mathcal{S}$ *if there exists an initial state* $z(0)$ *with*

$$\mathrm{Prob}(z_p(0) = z(0)) > 0$$

such that the stochastic automaton with given input sequence V *generates the given output sequence* W *with non-vanishing probability:*

$$\mathrm{Prob}(W(0 \dots k_h) \,|\, V(0 \dots k_h), z(0)) > 0. \tag{8.47}$$

This consideration has the important consequence that every diagnostic problem includes a state observation problem unless the state sequence can be measured. Therefore, this section presents a formal statement of the state observation problems for stochastic automata and develops the solution to it. On this basis, the next section describes methods for consistency-based diagnosis of stochastic automata.

8.3.2 Observation problem

The state observation problem concerns the situation in which the I/O pair

$$(V(0 \dots k_h), W(0 \dots k_h))$$

is known and the current state $z(k_h)$ of the stochastic automaton is to be determined.

Problem 8.1 (Observation problem for the stochastic automaton)

Given: *Input sequence* $V(0 \dots k_h)$.
Output sequence $W(0 \dots k_h)$.
Stochastic automaton $\mathcal{S}$.

Find: *Current state* $z(k_h)$.

The solution to the observation problem is, in general, not unique but given by the set

$$\mathcal{Z}(k_h \mid k_h) = \{z(k_h) \,:\, \mathrm{Prob}(z(k_h) \mid W(0 \dots k_h), V(0 \dots k_h)) > 0\}, \tag{8.48}$$

which includes all states $z(k_h)$ to which the automaton may move with non-vanishing probability while accepting the given input sequence V and generating the given output sequence W both of which are known for the time horizon k_h. The set $\mathcal{Z}(k_h \,|\, k_h)$ depends upon the time horizon k_h and upon the I/O pair, but only the dependency upon k_h is indicated explicitly. The two arguments of $\mathcal{Z}$ refer to the time instant for which the state should be reconstructed and to the time horizon of the I/O pair. $\mathcal{Z}(k_h|k_h - 1)$ is the set of states $z(k_h)$ that the automaton can assume after it has obtained the input sequence $V(0 \dots k_h - 1)$ and generated the output sequence $W(0 \dots k_h - 1)$ both of which have the time horizon $k_h - 1$.

The observation starts at $k_h = 0$ with the a-priori initial state probability distribution $\mathrm{Prob}(z(0))$, which is given for the stochastic automaton (8.12) and defines the initial set

$$\mathcal{Z}_0 = \{z(0) \,:\, \mathrm{Prob}(z(0)) > 0\}.$$

This knowledge about the initial state can be improved after the first input $v(0)$ has been applied and the first output $w(0)$ observed. Therefore, the set $\mathcal{Z}_0$ is denoted by $\mathcal{Z}(0\,|\,-1)$

$$\mathcal{Z}(0\,|\,-1) = \{z(0)\,:\,\mathrm{Prob}(z(0)) > 0\}, \tag{8.49}$$

where the time horizon -1 indicates that this set is determined before the first I/O information is available. After the I/O pair $(v(0), w(0))$ has been measured, the set $\mathcal{Z}(0\,|\,0)$ is determined as described below. This set includes all $z(0) \in \mathcal{Z}(0\,|\,-1)$ for which the relation $L(z', w(0)|z(0), v(0)) > 0$ holds for some z'.

The observation problem does not only include the determination of the set $\mathcal{Z}$ but also the determination of the probability distribution

$$\mathrm{Prob}(z(k_h)\,|\,V(0\,...\,k_h), W(0\,...\,k_h)) \quad \text{for all} \;\; z(k_h) \in \mathcal{N}_z,$$

which will be abbreviated by

$$\mathrm{Prob}(z(k_h)\,|\,k_h) := \mathrm{Prob}(z(k_h)\,|\,V(0\,...\,k_h), W(0\,...\,k_h)). \tag{8.50}$$

According to this convention, $\mathrm{Prob}(z(k_h)\,|\,k_h - 1)$ denotes the probability distribution of $z(k_h)$ which is determined from the I/O sequences $V(0\,...\,k_h - 1)$ and $W(0\,...\,k_h - 1)$ obtained for the time horizon $k_h - 1$. With the abbreviation (8.50) the set $\mathcal{Z}(k_h\,|\,k_h)$ is given by

$$\mathcal{Z}(k_h\,|\,k_h) = \{z(k_h)\,:\,\mathrm{Prob}(z(k_h)\,|\,k_h) > 0\} \tag{8.51}$$

8.3.3 Consistent input-output pairs

This section investigates the existence of a solution to the state observation problem. The following lemma describes a condition for this, which follows from the equation

$$\mathrm{Prob}(W(0\,...\,k_h)\,|\,V(0\,...\,k_h), z(0)) = \sum_{Z(1\ldots k_h+1)} \prod\nolimits_{k=0}^{k_h} L(k) \cdot \mathrm{Prob}(z(0))$$

obtained from the second and third line of Eq. (8.30) and the abbreviation (8.31).

Lemma 8.1 *An I/O pair $(V(0\,...\,k_h), W(0\,...\,k_h))$ is consistent with the stochastic automaton $\mathcal{S}$ if and only if the following condition holds:*

$$\begin{aligned} \sum_{Z(1\ldots k_h+1)} & L(z(k_h+1), w(k_h)\,|\,z(k_h), v(k_h)) \cdot \\ & \cdot L\,(z(k_h), w(k_h-1)\,|\,z(k_h-1), v(k_h-1)) \cdot \\ & \cdot L(z(1), w(0)\,|\,z(0), v(0)) > 0\;. \end{aligned} \tag{8.52}$$

If condition (8.52) is satisfied it is guaranteed that there exists an initial state $z(0)$ such that Eq. (8.47) holds.

The connection between the consistency of the pair (V, W) and the existence of solutions to the observation problems is obvious. If the I/O pair is consistent,

there exists at least one state sequence $Z(0\,...\,k_h)$ for which $\mathrm{Prob}(Z \,|\, V, W) > 0$ holds. The final state $z(k_h)$ of this state sequence belongs to $\mathcal{Z}(k_h \,|\, k_h)$. If the I/O pair is not consistent, there is no state sequence for which the stochastic automaton generates the output sequence W. Hence, the following corollary holds:

Corollary 8.1 *A solution to the observation problem exists if and only if the I/O pair (V, W) is consistent with the stochastic automaton.*

8.3.4 Solution to the state observation problem

In this section the solution to the state observation problem is presented and illustrated by means of a simple example. An important preliminary result concerns the probability distribution $\mathrm{Prob}(Z(0\,...\,k_h) \,|\, V(0\,...\,k_h), W(0\,...\,k_h))$ over all state sequence $Z(0\,...\,k_h)$ of the automaton for a given I/O pair.

Theorem 8.1 (Probability distribution of the state sequence)
Consider a stochastic automaton with initial state probability distribution $\mathrm{Prob}(z(0))$ *and a consistent I/O pair* (V, W). *Then*

$$\begin{aligned}&\mathrm{Prob}(Z(0\,...\,k_h) \mid V(0\,...\,k_h), W(0\,...\,k_h)) \\ &= \frac{\sum\limits_{z(k_h+1)} L(k_h) \cdot ... \cdot L(1) \cdot L(0) \cdot \mathrm{Prob}(z(0))}{\sum\limits_{Z(0\,...\,k_h+1)} L(k_h) \cdot ... \cdot L(1) \cdot L(0) \cdot \mathrm{Prob}(z(0))}\end{aligned} \tag{8.53}$$

describes the probability that the stochastic automaton has generated the state sequence $Z(0\,...\,k_h)$.

Note that the denominator in Eq. (8.53) does not vanish because the I/O pair is assumed to be consistent and, hence, Eq. (8.52) holds. In the application of Theorem 8.1 first the inequality (8.52) can be checked to find out whether the I/O pair is consistent with the automaton and if this test is successful Eq. (8.53) is applied to determine

$$\mathrm{Prob}(Z(0\,...\,k_h) \mid V(0\,...\,k_h), W(0\,...\,k_h)).$$

Proof. According to Bayes' formula, the relation

$$\begin{aligned}&\mathrm{Prob}(Z(0\,...\,k_h) \mid V(0\,...\,k_h), W(0\,...\,k_h)) \\ &= \frac{\mathrm{Prob}(Z(0\,...\,k_h), V(0\,...\,k_h), W(0\,...\,k_h))}{\mathrm{Prob}(V(0\,...\,k_h), W(0\,...\,k_h))}\end{aligned} \tag{8.54}$$

holds, where the condition $\mathrm{Prob}(V(0\,...\,k_h), W(0\,...\,k_h)) > 0$ is satisfied because the pair (V, W) is assumed to be consistent with the stochastic automaton. Equation (8.54) can be reformulated as

$$\text{Prob}(Z(0\,...\,k_h) \mid V(0\,...\,k_h), W(0\,...\,k_h)) \tag{8.55}$$
$$= \frac{\sum\limits_{z(k_h+1)} \text{Prob}(z(0), ..., z(k_h+1), v(0), ..., v(k_h), w(0), ..., w(k_h))}{\sum\limits_{Z(0\,...\,k_h+1)} \text{Prob}(z(0), ..., z(k_h+1), v(0), ..., v(k_h), w(0), ..., w(k_h))} .$$

The probability distribution that appears in the numerator and denominator of (8.55) can be simplified as follows:

$$\text{Prob}(z(0), ..., z(k_h+1), v(0),\ ...,\ v(k_h), w(0),\ ...,\ w(k_h)) = \tag{8.56}$$

$$\text{Prob}(z(k_h+1), w(k_h) \mid z(k_h), v(k_h), Z(0\,...\,k_h-1), V(0\,...\,k_h-1), W(0\,...\,k_h-1)) \cdot$$
$$\cdot \text{Prob}(z(k_h), v(k_h), Z(0\,...\,k_h-1), V(0\,...\,k_h-1), W(0\,...\,k_h-1)) = \tag{8.57}$$

$$\text{Prob}(z(k_h+1), w(k_h) \mid z(k_h), v(k_h)) \cdot$$
$$\cdot \text{Prob}(z(0), ..., z(k_h), v(0),\ ...,\ v(k_h), w(0),\ ...,\ w(k_h-1)) \tag{8.58}$$

$$= ... =$$
$$\text{Prob}(z(k_h+1), w(k_h) \mid z(k_h), v(k_h)) \cdot \tag{8.59}$$
$$\cdot \text{Prob}(z(k_h), w(k_h-1) \mid z(k_h-1), v(k_h-1)) \cdot$$
$$\cdot ... \cdot \text{Prob}(z(1), w(0) \mid z(0), v(0)) \cdot \text{Prob}(z(0), v(0), .., v(k_h)) = \tag{8.60}$$

$$L(k_h) \cdot L(k_h-1) \cdot ... \cdot L(0) \cdot \text{Prob}(z(0)) \cdot \text{Prob}(v(0),\ ...,\ v(k_h)) . \tag{8.61}$$

To obtain Eq. (8.57), Bayes' formula was used. The first probability in (8.57) can be reformulated as (8.58) due to the Markov property (8.8). The second probability distribution in (8.57) and (8.58) has a similar form as (8.56). Only the state z and output w with the highest time index have disappeared in the list of arguments. Hence, the same simplification steps can be carried out several times until Eq. (8.60) is obtained. As $z(0)$ is independent of $v(0), .., v(k_h)$ and the relation

$$\text{Prob}(z(k+1), w(k) \mid z(k), v(k)) = L(k) \quad \text{for all } k$$

holds, Eq. (8.61) is obtained. Finally, (8.53) results from inserting (8.61) into (8.55) and simplifying the resulting expression. □

From Theorem 8.1 the solution to the observation problem is obtained by determining the conditional probability distributions

$$\begin{aligned} \text{Prob}(z(k_h) \,|\, k_h) &= \text{Prob}(z(k_h) \mid V(0\,...\,k_h), W(0\,...\,k_h)) \\ &= \sum_{Z(0\,...\,k_h-1)} \text{Prob}(Z(0\,...\,k_h) \mid V(0\,...\,k_h), W(0\,...\,k_h)). \end{aligned} \tag{8.62}$$

The results are summarised in the following theorem, which is a direct consequence of Theorem 8.1:

Theorem 8.2 (Solution to the observation problem)
Consider a stochastic automaton with the initial state probability distribution $\text{Prob}(z(0))$. *If the I/O pair* (V, W) *is consistent with the automaton, the current state probability distribution is given by*

$$= \frac{\sum\limits_{Z(0\,\ldots\,k_h-1)} \; \sum\limits_{z(k_h+1)} L(k_h) \cdot L(k_h-1) \cdot \ldots \cdot L(1) \cdot L(0) \cdot \mathrm{Prob}(z(0))}{\sum\limits_{Z(0\ldots k_h+1)} L(k_h) \cdot L(k_h-1) \cdot \ldots \cdot L(1) \cdot L(0) \cdot \mathrm{Prob}(z(0))} \tag{8.63}$$

and the set of current automaton states by Eq. (8.48).

Note that for time $k_h = 0$ Eq. (8.63) yields the probability distribution $\mathrm{Prob}(z(0)\,|\,0)$ that the automaton is in the initial state $z(0)$ and has generated the output $w(0)$ for the input $v(0)$. This probability is, in general, different from $\mathrm{Prob}(z(0))$, which represents the a-priori probability distribution of the initial state. Hence, the set

$$\mathcal{Z}(0\,|\,0) = \{z(0)\,:\,\mathrm{Prob}(z(0)\,|\,0) > 0\}$$

is different from $\mathcal{Z}(0\,|\,-1)$ given in Eq. (8.49) and satisfies the relation

$$\mathcal{Z}(0\,|\,0) \subseteq \mathcal{Z}(0\,|\,-1).$$

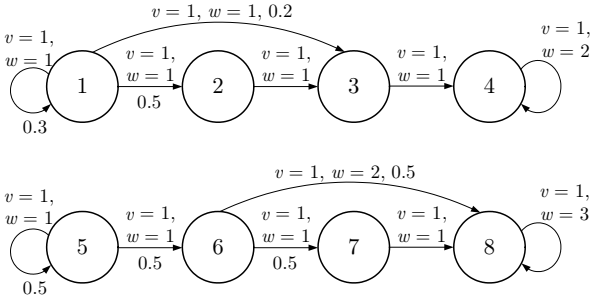


Fig. 8.10. Automaton graph of the example

Example 8.2 *State observation of a stochastic automaton*

Consider the stochastic automaton whose automaton graph is shown in Fig. 8.10 and whose initial state is uniformly distributed:

$$\mathrm{Prob}(z_p(0) = z) = \frac{1}{8} \qquad (z = 1, ..., 8).$$

The value of the behavioural relation L is indicated at the corresponding edges of the automaton graph unless its value is 1.

All state sequences that occur with non-vanishing probability for the input sequence

$$V(0\,...\,4) = (1, 1, 1, 1, 1)$$

and yield the output sequence

$$W(0\,...\,4) = (1, 1, 1, 3, 3)$$

are shown in Table 8.1 together with the probabilities

$$\begin{aligned} &\mathrm{Prob}(z(0) \mid v(0), w(0)), && \mathrm{Prob}\left(Z(0\ldots1) \mid V(0\ldots1), W(0\ldots1)\right), \\ &\qquad\ldots, && \mathrm{Prob}(Z(0\ldots4) \mid V(0\ldots4), W(0\ldots4)), \end{aligned}$$

which are obtained by means of Eq. (8.53). Note that the value of

$$\mathrm{Prob}(Z(0\ldots0) \,|\, V(0\ldots0), W(0\ldots0)) = \mathrm{Prob}(z(0) \,|\, v(0), w(0))$$

which is shown in column #2 differs from the a-priori probability distribution $\mathrm{Prob}(z(0)) = \frac{1}{8}$ because this a-posteriori probability includes the information provided by the I/O pair $(v(0), w(0))$. This is the reason why the states $z = 4$ and $z = 8$, both of which are assumed by the automaton with the a-priori probability $\mathrm{Prob}(z(0)) = \frac{1}{8}$ do not appear in column #1.

Table 8.1 Probability distribution $\mathrm{Prob}(Z \mid V, W)$ of the example automaton

$k_h = 0$ $V(0\ldots0) = (1)$ $W(0\ldots0) = (1)$		$k_h = 1$ $V(0\ldots1) = (1,1)$ $W(0\ldots1) = (1,1)$		$k_h = 2$ $V(0\ldots2) = (1,1,1)$ $W(0\ldots2) = (1,1,1)$	
$Z(0\ldots0)$	$\mathrm{Prob}(Z\|V,W)$	$Z(0\ldots1)$	$\mathrm{Prob}(Z\|V,W)$	$Z(0\ldots2)$	$\mathrm{Prob}(Z\|V,W)$
(1)	0.1818	(1,1)	0.0923	(1,1,1)	0.0632
(2)	0.1818	(1,2)	0.1538	(1,1,2)	0.1053
(3)	0.1818	(1,3)	0.0615	(1,1,3)	0.0421
(5)	0.1818	(2,3)	0.3077	(1,2,3)	0.3509
(6)	0.0909	(5,5)	0.1538	(5,5,5)	0.1754
(7)	0.1818	(5,6)	0.0769	(5,5,6)	0.0877
		(6,7)	0.1538	(5,6,7)	0.1754
#1	#2	#3	#4	#5	#6

$k_h = 3$ $V(0\ldots3) = (1,1,1,1)$ $W(0\ldots3) = (1,1,1,3)$		$k_h = 4$ $V(0\ldots4) = (1,1,1,1,1)$ $W(0\ldots4) = (1,1,1,3,3)$	
$Z(0\ldots3)$	$\mathrm{Prob}(Z \mid V,W)$	$Z(0\ldots4)$	$\mathrm{Prob}(Z \mid V,W)$
(5,6,7,8)	1	(5,6,7,8,8)	1
#7	#8	#9	#10

The state probability distribution obtained by means of Eq. (8.63) is shown in Fig. 8.11. The probabilities are depicted in grey scale. Black rectangles symbolise a probability of one, white rectangles a probability of zero. The set $\mathcal{Z}(k_h \,|\, k_h)$ includes all states $z(k_h)$ with non-zero probability (grey and black boxes):

$$\begin{aligned} \mathcal{Z}(0\,|\,0) &= \{1,2,3,5,6,7\} \\ \mathcal{Z}(1\,|\,1) &= \{1,2,3,5,6,7\} \\ &\vdots \\ \mathcal{Z}(4\,|\,4) &= \{8\}\ .\square \end{aligned}$$

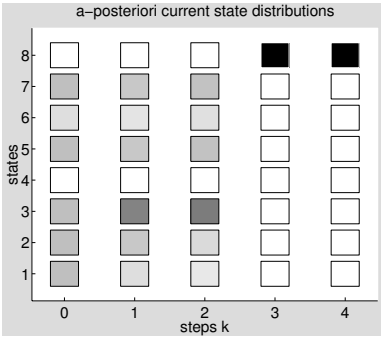


Fig. 8.11. Observation result

8.3.5 Recursive form of the solution

For the application, the elements of the sequences $V(0 \ldots k_h)$ and $W(0 \ldots k_h)$ appear one after the other for $k_h = 0, 1, 2, \ldots$ and should be processed in this way. Therefore, the following recursive form of the solution to the state observation problem is important (for a proof cf. [157]).

Theorem 8.3 (Recursive solution to the state observation problem)
Consider a stochastic automaton with the initial state distribution $\mathrm{Prob}(z(0))$. *If the I/O pair* (V, W) *is consistent with the stochastic automaton, the a-posteriori state probability distribution is given by the recursive relations*

$$\mathrm{Prob}(z(k_h) \,|\, k_h) = \frac{\sum\limits_{z(k_h+1)} L(k_h) \cdot \mathrm{Prob}(z(k_h) \mid k_h - 1)}{\sum\limits_{z(k_h), z(k_h+1)} L(k_h) \cdot \mathrm{Prob}(z(k_h) \mid k_h - 1)}\,, \qquad (k_h \geq 0) \tag{8.64}$$

with

$$\mathrm{Prob}(z(k_h) \mid k_h - 1) = \frac{\sum\limits_{z(k_h-1)} L(k_h - 1) \cdot \mathrm{Prob}(z(k_h - 1) \mid k_h - 2)}{\sum\limits_{z(k_h), z(k_h-1)} L(k_h - 1) \cdot \mathrm{Prob}(z(k_h - 1) \mid k_h - 2)}, \qquad (k_h > 0) \tag{8.65}$$

$$\mathrm{Prob}(z(0) \mid -1) ::= \mathrm{Prob}(z(0)) \qquad (k_h = 0). \tag{8.66}$$

Note that the input $v(k_h)$ and output $w(k_h)$ measured at time k_h appear as arguments in

$$L(k_h) = L\left(z(k_h + 1),\ w(k_h) \mid z(k_h),\ v(k_h)\right).$$

Equation (8.65) is used after the pair $v(k_h-1), w(k_h-1)$ has been measured, which likewise occurs in $L(k_h-1)$. It describes a "prediction" of the state $z(k_h)$ by using the information about the movement of the stochastic automaton until time k_h-1. It is a recursive relation whose initial value is given by Eq. (8.66). Equation (8.65) makes it possible to determine $\mathrm{Prob}(z(k_h) \mid k_h-1)$ for given $\mathrm{Prob}(z(k_h-1) \mid k_h-2)$ and the new measurements $v(k_h-1)$ and $w(k_h-1)$. Hence, only the probability distribution $\mathrm{Prob}(z(k_h-1) \mid k_h-2)$ has to be stored in the computer memory, which consists of N values.

Equation (8.64) describes how the prediction from the previous time point has to be corrected after the new measurements $v(k_h)$ and $w(k_h)$ became available. This step can be interpreted as a "projection" onto the set of those states $z(k_h)$ from which the automaton can have moved when generating the new measurement information $w(k_h)$. The result of the recursion is the a-posteriori probability distribution $\mathrm{Prob}(z(k_h) \mid k_h)$ of the current state $z(k_h)$ for the given measurements until time k_h. With this structure, the recursive solution to the state observation problem shows a remarkable similarity to the Kalman filter, which likewise can be decomposed into a prediction and a projection step.

8.3.6 Discussion of the results

The following remarks concern the application of the observation method. The first two comments deal with problems that may appear if the results presented in Theorems 8.2 and 8.3 are used to implement an observation algorithm. Section 8.3.7 presents such an algorithm, which also refers to the consequences of the following remarks concerning the handling of inconsistent I/O pairs or wrong a-priori knowledge about the initial automaton state.

Inconsistent I/O pairs. An I/O pair that is generated by some stochastic process is consistent with the stochastic automaton that describes this stochastic process. However, in an application, the sequences processed by the observation algorithm may be subjected to some measurement errors, which may cause the I/O pair to become inconsistent. In the application of the observation algorithm for diagnostic purposes, the I/O pair may be generated by the faulty system and, hence, the I/O pair may be inconsistent with the automaton with the behavioural relation of the faultless system. The question arises what happens with the observation result.

The inconsistency of the given I/O pair leads to a violation of condition (8.52), which implies that the denominators in Eqs. (8.64) and (8.65) vanish and no solution to the observation problems is found. The observation algorithm should, therefore, be interrupted if the mentioned denominators

$$\sum_{z(k_h),z(k_h+1)} L(k_h) \cdot \mathrm{Prob}(z(k_h) \mid k_h-1)$$

become zero, because this indicates the inconsistency of the I/O pair. Such a test is included in the algorithm presented in Section 8.3.7.

A-priori knowledge about the initial state. In the solution to the state observation problem, the initial state probability distribution $\mathrm{Prob}(z(0))$ is assumed to be known. Since in applications, this a-priori knowledge is often not available, $\mathrm{Prob}(z(0))$ is measured or "guessed". The question arises what happens if the a-priori knowledge about $z(0)$ is in conflict with the "real" initial state of the stochastic process.

To answer this question, assume that $\mathrm{Prob}(\hat{z}(0))$ denotes the approximate initial state probability distribution and consider the sets

$$\mathcal{Z}_0 = \{z : \mathrm{Prob}(z_p(0)=z) > 0\} \subseteq \mathcal{N}_z \tag{8.67}$$

$$\hat{\mathcal{Z}}_0 = \{z : \mathrm{Prob}(\hat{z}_p(0)=z) > 0\} \subseteq \mathcal{N}_z \,. \tag{8.68}$$

The stochastic process starts from an initial state $z_0 \in \mathcal{Z}_0$, whereas in the observation algorithm it is assumed that the automaton starts from some state $z_0 \in \hat{\mathcal{Z}}_0$. The a-priori knowledge about the initial state is not in conflict with the real system if the relation

$$\hat{\mathcal{Z}}_0 \supseteq \mathcal{Z}_0 \tag{8.69}$$

holds true. Then, the solution to the observation problem ensures that the relation

$$z(k_h) \in \mathcal{Z}(k_h \mid k_h)$$

is valid for all $k_h \geq 0$, i.e. the set of current states determined by the observation algorithm includes the true state of the stochastic process. If the probability distribution $\mathrm{Prob}(\hat{z}(0))$ used in the observation algorithm is different from the "real" distribution $\mathrm{Prob}(z(0))$, the probability distribution $\mathrm{Prob}(z(k_h) \,|\, k_h)$ obtained by the algorithm is wrong, but it is accepted in practice as solution to the observation problem due to the lack of a better a-priori knowledge.

If, however, condition (8.69) is violated, then it is possible that the probability $\mathrm{Prob}(W(0 \,...\, k_h) \,|\, V(0 \,...\, k_h), z(0))$ is zero for all initial states $z(0) \in \hat{\mathcal{Z}}_0$. Consequently, like in the case of an inconsistent I/O pair, the violation of the condition (8.69) makes the denominators in Eqs. (8.64) and (8.65) vanish, which can be used as an indicator to stop the observation algorithm.

This and the preceding remarks show that the observation algorithm cannot distinguish between an inconsistent I/O pair and a wrong initial state probability distribution.

As a consequence, in an application the set $\hat{\mathcal{Z}}_0$ has to be chosen "large enough". A secure way is to choose $\mathrm{Prob}(\hat{z}(0))$ so that $\hat{\mathcal{Z}}_0 = \mathcal{N}_z$ holds, for example, by using the uniform initial state distribution

$$\mathrm{Prob}(\hat{z}_p = z) = \frac{1}{N} \qquad \text{for all } \; z \in \mathcal{N}_z \,. \tag{8.70}$$

Note that in the algorithm presented in Section 8.3.7, $\mathrm{Prob}(z(0))$ is replaced by $\mathrm{Prob}(\hat{z}(0))$ because $\mathrm{Prob}(z(0))$ is usually not known.

Besides the lack of knowledge of $\mathrm{Prob}(z(0))$ another fact makes it reasonable to use the a-priori probability distribution (8.70). For many stochastic automata, the solution $\mathrm{Prob}(z(k_h) \mid k_h)$ of the observation problem is (nearly) independent

of $\mathrm{Prob}(z(0))$ for $k_h \geq \bar{k}$ with very small $\bar{k}$. This is particularly true if the set $Z(k_h \mid k_h)$ has only a few elements compared to the cardinality N of the state set $\mathcal{N}_z$.

Current state observation starting from a known initial state. State observation problems are usually solved if the initial state z_0 is not known. However, for non-deterministic systems the problem of reconstructing the current state makes sense also if z_0 is known because the knowledge of the current state is lost if the automaton makes a non-deterministic state change. This fact will be demonstrated by the following example.

Example 8.3 *State observation for known initial state*

Assume that for the stochastic automaton defined in Example 8.2 the initial state is known to be $z_0 = 1$ (Fig. 8.10). If the automaton was deterministic, for a given input sequence V the current states $z(k_h)$ could be determined for $k_h = 1, 2, ...$ unambiguously. However, since the stochastic automaton is non-deterministic, the state observation problem remains a problem to be solved.

Simulation with the given input sequence $V(0 \ldots 3) = (1, 1, 1, 1)$ yields the probability $\mathrm{Prob}(z(k_h) \,|\, V)$ given by Eq. (8.26) and shown in Table 8.2. For $k = 1$ the results obtained from simulation (column #2) and from observation (column #3) do not distinguish because for all state transitions the same output occurs. However, since the transition from state 4 towards state 4, which is a possible simulation result, generates the output $w = 2$, which contradicts the observed output $w(2) = w(3) = 1$, the simulation and the observation results given in columns #5 and #6 or #8 and #9, respectively, differ. In particular, the state observation yields the result that the automaton cannot be in state 4, which is predicted by the simulation with the probability of 0.2 or 0.76, respectively.

This example shows that due to the non-determinism of the stochastic process, the state observation problem has to be solved even if the initial state is unambiguously known. Observation refines the results obtained from simulation because observation uses also the information provided by the measured output sequence. □

Table 8.2 Comparison of the simulation and the observation results.

$k_h = 1$ $V(0 \ldots 1) = (1,1)$ $W(0 \ldots 1) = (1,1)$			$k_h = 2$ $V(0 \ldots 2) = (1,1,1)$ $W(0 \ldots 2) = (1,1,1)$		
$z(1)$	$\text{Prob}(z\|V)$	$\text{Prob}(z\|V,W)$	$z(2)$	$\text{Prob}(z\|V)$	$\text{Prob}(z\|V,W)$
1	0.3	0.3	1	0.09	0.1125
2	0.5	0.5	2	0.15	0.1875
3	0.2	0.2	3	0.56	0.7000
4	0	0	4	0.20	0
#1	#2	#3	#4	#5	#6

$k_h = 3$ $V(0 \ldots 3) = (1,1,1,1)$ $W(0 \ldots 3) = (1,1,1,1)$		
$z(3)$	$\text{Prob}(z\|V)$	$\text{Prob}(z\|V,W)$
1	0.027	0.1125
2	0.045	0.1875
3	0.168	0.7000
4	0.760	0
#7	#8	#9

8.3.7 Observation algorithm

To show how the observation method developed in this section can be applied on-line, this section presents an observation algorithm, which is based on the recursive solution given in Theorem 8.3. The following abbreviations are used:

$$h(z(k_h)) = \sum_{z(k_h+1)} L(k_h) \cdot \text{Prob}(z(k_h) \mid V(0 \ldots k_h - 1), W(0 \ldots k_h - 1))$$

$$p_k(z(k_h)) = \text{Prob}(z(k_h) \mid V(0 \ldots k_h), W(0 \ldots k_h))$$

$$p_r(z(k_h+1)) = \text{Prob}(z(k_h+1) \mid V(0 \ldots k_h), W(0 \ldots k_h))$$

Algorithm 8.1 *Observation of stochastic automata*

Given: Stochastic automaton $\mathcal{S}$.
Initial state probability distribution $\mathrm{Prob}(\hat{z}(0))$.

Initialisation: $p_r(z) = \mathrm{Prob}(\hat{z}_p(0) = z)$ for all $z \in \mathcal{N}_z$
$k_h = 0$.

Loop:

1. Measure the current input v and output w.
2. For all $z \in \mathcal{N}_z$ determine
$h(z) = \sum_{\bar{z}} L(\bar{z}, w \mid z, v) \cdot p_r(z)$.
3. If $\sum_z h(z) = 0$ holds, stop the algorithm (inconsistent I/O pair or wrong initial state distribution).
4. For all $z \in \mathcal{N}_z$ determine $p_k(z) = \dfrac{h(z)}{\sum\limits_z h(z)}$.
5. For all $z \in \mathcal{N}_z$ determine $p_r(z) = \dfrac{\sum\limits_{\bar{z}} L(z, w \mid \bar{z}, v)\, p_r(\bar{z})}{\sum\limits_z h(z)}$.
6. Determine $\mathcal{Z}(k_h \mid k_h)$ according to Eq. (8.48).
7. $k_h := k_h + 1$
Continue with Step 1.

Result: $p_k(z(k_h)) = \mathrm{Prob}(z(k_h) \mid V(0 \ldots k_h), W(0 \ldots k_h))$ and $\mathcal{Z}(k_h \mid k_h)$ for increasing time horizon k_h .

In Step 3 the consistency of the I/O pair and of the a-priori probability distribution is tested as described in Section 8.3.6. Steps 4 and 5 use Eq. (8.64) and (8.65), respectively, where the last equation is written for $k_h + 1$ instead of k_h. The test in Step 3 ensures that the denominators of both equations do not vanish.

Note that in each step of the algorithm very easy calculations have to be carried out, which makes the algorithm applicable under relatively strong real-time constraints. Since the algorithm is based on the recursive solution to the state observation problem, its complexity does *not* increase with the length of the measurement sequences V and W. Only the N values of the functions $h(z)$, $p_k(z)$ and $p_r(z)$ have to be stored in the memory.

8.3.8 State observation of non-deterministic automata

This section concerns the observation problem for the non-deterministic automaton. The solution can be obtained directly from the solution developed for the stochastic automaton if the probabilistic information is replaced by assertions that simply say whether or not a state belongs to the solution set $\mathcal{Z}(k_h \mid k_h)$. The solution is given here because it illustrates the solution, which has been derived by means of probability theory in the preceding sections, from a different viewpoint.

The observation problem to be solved is the same as that given on p. 390 with the stochastic automaton $\mathcal{S}$ replaced by the non-deterministic automaton

$$\mathcal{N} = (\mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, L_n, z_0).$$

In contrast to the stochastic automaton, the behavioural relation L_n of the non-deterministic automaton says only whether or not the transition from state $z(k)$ towards state $z(k+1)$ is possible for the given input $v(k)$ and output $w(k)$. This transition is possible if

$$L_h(z(k+1), w(k), z(k), v(k)) = 1$$

or, equivalently,

$$(z(k_h+1), w(k), z(k), v(k)) \in R(L_h)$$

holds (cf. Eq. (8.5)), but no probabilistic information about the occurrence of this transition is given. Therefore, all probabilities are replaced by an indicator function "Poss" that describes whether or not a certain input, state or output possibly occurs (The operator name Poss abbreviates *possibility*). Accordingly,

$$\mathrm{Poss}(z(k_h) = z \mid k_h) = 1$$

indicates that the state z is possibly assumed by the non-deterministic automaton at time k_h and

$$\mathrm{Poss}(z(k_h) = z \mid k_h) = 0$$

says that z cannot be assumed.

The initial state z_0 of the non-deterministic automaton is not known. It is assumed that a set $\mathcal{Z}(0) \subseteq \mathcal{N}_z$ is given such that $z_0 \in \mathcal{Z}(0)$ holds. This set represents the a-priori knowledge about z_0.

The diagnostic solution for the non-deterministic automaton can be obtained directly from the solution developed for the stochastic automaton if the following analogies are taken into account:

$$\begin{aligned}
L(z', w \mid z, v) > 0 \quad &\Longleftrightarrow \quad (z', w, z, v) \in \mathcal{R}(L_n)\ , \\
\mathrm{Prob}(z(k_h) \mid k_h) > 0 \quad &\Longleftrightarrow \quad \mathrm{Poss}(z(k_h) \mid k_h) = 1\ , \\
\mathrm{Prob}(z(k_h) \mid k_h) = 0 \quad &\Longleftrightarrow \quad \mathrm{Poss}(z(k_h) \mid k_h) = 0\ , \\
z(k_h) \in \mathcal{Z}(k_h \mid k_h) \quad &\Longleftrightarrow \quad z(k_h) \in \mathcal{Z}_n(k_h \mid k_h)\ .
\end{aligned}$$

These analogies are equivalences if the initial state of the non-deterministic automaton is known to be in the set $\mathcal{Z}(0 \mid -1)$ defined for the stochastic automaton. Note

that for the stochastic automaton the analogies of Eqs. (8.64) – (8.66) yield only yes/no answers instead of probability values:

$$\mathrm{Poss}(z(k_h) \mid k_h) = \begin{cases} 1 & \text{if } \mathrm{Poss}(z(k_h) \mid k_h - 1) = 1 \text{ and} \\ & \exists z(k_h+1) : (z(k_h+1), w(k_h), z(k_h), v(k_h)) \in \mathcal{R}(L_n) \\ 0 & \text{else} \end{cases}$$

$$(k_h \geq 0) \tag{8.71}$$

with

$$\mathrm{Poss}(z(k_h) \mid k_h - 1) = \tag{8.72}$$

$$\begin{cases} 1 & \text{if } \exists z(k_h - 1) : \mathrm{Poss}(z(k_h - 1) \mid k_h - 2) = 1 \text{ and} \\ & \quad (z(k_h), w(k_h - 1), z(k_h - 1), v(k_h - 1)) \in \mathcal{R}(L_n) \quad (k_h > 0) \\ 0 & \text{else} \end{cases}$$

$$\mathrm{Poss}(z(0) \mid -1) = \mathrm{Poss}(z(0)) \qquad (k_h = 0). \tag{8.73}$$

In Eqs. (8.71) – (8.73) a positive value for the left-hand side results whenever in Eqs. (8.64) – (8.66) the numerator is positive. This result can be reformulated in a shorter way if the sets

$$\begin{aligned} \mathcal{Z}(k_h \mid k_h) &= \{z : \mathrm{Poss}(z(k_h) \mid k_h) = 1\} \\ \mathcal{Z}(k_h \mid k_h - 1) &= \{z : \mathrm{Poss}(z(k_h) \mid k_h - 1) = 1\} \end{aligned}$$

are used:

Theorem 8.4 (Recursive solution to the state observation problem for the non-deterministic automaton)
Consider a non-deterministic automaton with the initial state set $\mathcal{Z}(0)$. If the I/O pair (V, W) is consistent with the non-deterministic automaton, then the set of possible current states is given by

$$\mathcal{Z}(k_h \mid k_h) = \{z \in \mathcal{Z}(k_h \mid k_h - 1) \quad : \quad (\tilde{z}, w(k_h), z, v(k_h)) \in \mathcal{R}(L_n) \quad \text{for some } \tilde{z} \in \mathcal{N}_z \} \qquad (k_h \geq 0), \tag{8.74}$$

where

$$\mathcal{Z}(k_h \mid k_h - 1) = \{z : (z, w(k_h - 1), \tilde{z}, v(k_h - 1)) \in \mathcal{R}(L_n) \quad \text{for some } \tilde{z} \in \mathcal{Z}(k_h - 1 \mid k_h - 1)\} \quad (k_h > 0) \tag{8.75}$$

$$\mathcal{Z}(0 \mid -1) = \mathcal{Z}(0) \qquad (k_h = 0) \tag{8.76}$$

holds.

An I/O pair is consistent with the no-deterministic automaton if for some initial state in the given initial state set the automaton may generate a state sequence for the given input sequence V such that the output sequence W is obtained. After the I/O pair $(V(0 \ldots k_h - 1), W(0 \ldots k_h - 1))$ has been obtained, Eq. (8.75) predicts all

future states $z(k_h)$ that the non-deterministic automaton can assume. These states are given by the behavioural relation L_n for the last input $v(k_h-1)$, last output $w(k_h-1)$ and the states $z(k_h-1) \in \mathcal{Z}(k_h-1 \,|\, k_h-1)$. This step corresponds to the "prediction" step discussed in Section 8.3.5 for the stochastic automaton. For $k_h = 0$ the result is given by Eq. (8.76).

Equation (8.74) determines the set of current states $z(k_h)$ by selecting all those states from the set $\mathcal{Z}(k_h \,|\, k_h-1)$ from which the automaton can go to another state for the measured input $v(k_h)$ while generating the measured output $w(k_h)$. This is the "projection" step discussed in Section 8.3.5 for the stochastic automaton.

The theorem leads to the following observation algorithm, in which the sets

$$\begin{aligned} \mathcal{Z}_k &= \mathcal{Z}(k_h \mid k_h) \\ \mathcal{Z}_r &= \mathcal{Z}(k_h \mid k_h-1) \end{aligned}$$

are recursively updated after a new measurement information has been obtained.

Algorithm 8.2 *Observation of non-deterministic automata*

Given: Non-deterministic automaton $\mathcal{A}$.
Initial state set $\mathcal{Z}(0)$.

Initialisation: $\mathcal{Z}_r = \mathcal{Z}(0)$
$k_h = 0$.

Loop:

1. Measure the current input v and output w.
2. Determine
   $\mathcal{Z}_k := \{z \in \mathcal{Z}_r \,:\, (\tilde{z}, w, z, v) \in \mathcal{R}(L_n) \text{ for some } \tilde{z} \in \mathcal{N}_z\}$.
3. If $\mathcal{Z}_k = \emptyset$, stop the algorithm (inconsistent I/O pair or wrong initial state set).
4. Determine
   $\mathcal{Z}_r := \{z \,:\, (z,\, w,\, \tilde{z},\, v) \in \mathcal{R}(L_n) \text{ for some } \tilde{z} \in \mathcal{Z}_k\}$.
5. $k := k+1$
   Continue with Step 1.

Result: $\mathcal{Z}_k = \mathcal{Z}(k_h \mid k_h)$ for increasing time horizon k_h.

Example 8.4 *State observation of a non-deterministic automaton*

Consider the automaton used in Example 8.2 whose automaton graph is depicted in Fig. 8.10 on page 394. The non-deterministic automaton which will be investigated now is obtained if the state transition probabilities associated with the edges of the graph are ignored.

The initial state is assumed to belong to the set

$$\mathcal{Z}(0 \mid -1) = \{1, 2, 3, 4, 5, 6, 7, 8\},$$

which yields the initial value of $\mathcal{Z}_r$ in the initialisation step of the algorithm. After the first measurement

$$v(0) = 1 \quad \text{and} \quad w(0) = 1$$

has been obtained, the projection step yields

$$\mathcal{Z}_k = \{1, 2, 3, 5, 6, 7\}$$

which is identical to the set $\mathcal{Z}(0\,|\,0)$ of all initial states $z(0)$ for which a state transition exists such that the non-deterministic automaton generates the output $w(0) = 1$ for the input $v(0) = 1$ (Step 2 of the algorithm). In the automaton graph this set can be found by looking for all states $z \in \mathcal{Z}(0\,|\,-1)$ in which an edge starts that is labelled with $v = 1$ and $w = 1$. In Step 4 the algorithm determines the set

$$\mathcal{Z}_r = \{1, 2, 3, 4, 5, 6, 7, 8\} = \mathcal{Z}(1 \mid 0)$$

of states $z(1)$ that the automaton can assume after the state changes that have been considered just now. This prediction provides the set of states in which the automaton can be after it has generated the output $w(0) = 1$ for the input $v(0) = 1$.

For the next measurement

$$v(1) = 1 \quad \text{and} \quad w(1) = 1$$

the Steps 2 and 4 of the algorithm yield

$$\begin{aligned} \mathcal{Z}_k &= \{1, 2, 3, 5, 6, 7\} = \mathcal{Z}(1 \mid 1) \\ \mathcal{Z}_r &= \{1, 2, 3, 4, 5, 6, 7, 8\} = \mathcal{Z}(2 \mid 1) \end{aligned}$$

and for

$$v(2) = 1 \quad \text{and} \quad w(2) = 1$$

the sets

$$\begin{aligned} \mathcal{Z}_k &= \{1, 2, 3, 5, 6, 7\} = \mathcal{Z}(2 \mid 2) \\ \mathcal{Z}_r &= \{1, 2, 3, 4, 5, 6, 7, 8\} = \mathcal{Z}(3 \mid 2)\,. \end{aligned}$$

An improvement of the observation result is obtained after the measurement

$$v(3) = 1 \quad \text{and} \quad w(3) = 3$$

has been obtained, because then the singletons

$$\begin{aligned} \mathcal{Z}_k &= \{8\} = \mathcal{Z}(3 \mid 3) \\ \mathcal{Z}_r &= \{8\} = \mathcal{Z}(4 \mid 3) \end{aligned}$$

result. Hence, the state is found unambiguously to be 8.

Step 3 tests whether the set $\mathcal{Z}_k$ is nonempty, which indicates whether the I/O pair is consistent with the automaton. For $\mathcal{Z}_k = \emptyset$ no state sequence $Z(0 \ldots k_h)$ exists for which the automaton generates the measured output sequence for the given input sequence. □

The state observation algorithm for non-deterministic automata clearly demonstrates how state observation can be done for discrete-event systems. As the example has shown, the algorithm looks for state transitions that can occur under the measured input and output $v(k_h)$ and $w(k_h)$. This can be done recursively, because the information obtained from earlier measurements is stored in the set $\mathcal{Z}(k_h \mid k_h - 1)$. After the k_h-th measurement has been obtained, only those states have to be considered that are elements of $\mathcal{Z}(k_h \mid k_h - 1)$.

The I/O pair is inconsistent if no state change exists that starts in the set $\mathcal{Z}(k_h \mid k_h - 1)$ and yield for the input $v(k_h)$ the given output $w(k_h)$.

Comparison with the state observation of stochastic automata. The difference between the observation algorithms for the non-deterministic and the stochastic automaton lies only in the fact that for the stochastic automaton a probability distribution over the sets $\mathcal{Z}(k_h \mid k_h - 1)$ and $\mathcal{Z}(k_h \mid k_h)$ can be determined. For the stochastic automaton considered in Example 8.2 and for the non-deterministic automaton considered in Example 8.4 the resulting sets $\mathcal{Z}(k_h \mid k_h)$ are the same. However, for the stochastic automaton additional probabilistic information is obtained, which leads to the result presented in Fig. 8.11. This is an additional information, which in practice is very useful, because the state observation result is usually not a unique state but a set of several states. Then, the probability of the different states in the solution set helps to evaluate the observation result concerning their practical relevance. For example, if the automaton is found to be in a dangerous state then the probability with which this result is relevant can be used to decide about supervisory control actions.

8.3.9 Observability of stochastic automata

In this section, a notion of observability is introduced, which takes into account that for stochastic automata the state generally cannot be determined unambiguously. Even if the initial state is known, the automaton may produce an I/O pair that does not allow to track the state trajectory with certainty.

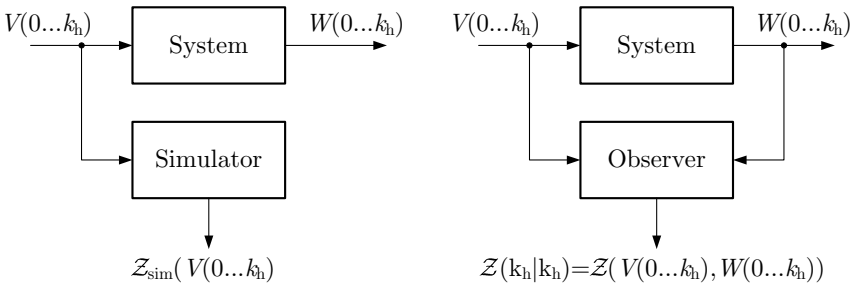


Fig. 8.12. Comparison of simulation and observation

The observability definition is based on a comparison of the results that are obtained by means of simulation and of observation (cf. Fig. 8.12). Roughly speaking, the stochastic automaton is called observable if it is possible to determine the state more precisely by state observation than by simulation. Before defining the observability in this way, simulation and observation have to be compared in more detail.

For both simulation and observation, the initial state distribution $\mathrm{Prob}(z(0))$ and the input sequence V have to be known.

- In **simulation**, the current state probability distribution is determined by Eq. (8.26) which propagates the initial state probability distribution according to the state transition relation G of the stochastic automaton and yields the current state probability distribution $\mathrm{Prob}(z(k_h) \mid V(0 \ldots k_h - 1))$. According to Eq. (8.29) the set of states in which the stochastic automaton recides at time k_h with non-vanishing probability for the given the input sequence $V(0 \ldots k_h)$ is

$$\mathcal{Z}_{\mathrm{sim}}(V(0 \ldots k_h)) = \{z(k_h) \,:\, \mathrm{Prob}(z(k_h) \mid V(0 \ldots k_h)) > 0\}\,. \tag{8.77}$$

- In **state observation** described by Theorem 8.3 the additional information included in the output sequence W is used to determine the probability

  $$\mathrm{Prob}(z(k_h) | V(0 \ldots k_h), W(0 \ldots k_h))$$

  of the state $z(k_h)$ provided that the input and the output sequences V and W are known according to Eq. (8.63). The set of states $\mathcal{Z}(k_h \mid k_h)$ is obtained by Eq. (8.48).

As the state observation uses the generated output sequence as additional constraint when determining the set of possible states $z(k_h)$, the relation

$$\mathcal{Z}_{sim}(V(0 \ldots k_h)) \supseteq \mathcal{Z}(k_h \mid k_h)$$

holds.

Stochastic unobservability. An automaton is called unobservable if for all input sequences V the I/O pair (V, W) does not include more information than the input V alone. This can be stated more formally as follows:

Definition 8.2 (Stochastic unobservability)
A stochastic automaton is called stochastically unobservable *if the behavioural relation L can be represented as the product of two functions*

$$\begin{aligned} G &: \mathcal{N}_z \times \mathcal{N}_z \times \mathcal{N}_v \to [0,1] \\ H &: \mathcal{N}_w \times \mathcal{N}_v \to [0,1] \end{aligned}$$

such that

$$L(z', w \mid z, v) = G(z' \mid z, v) \cdot H(w \mid v) \tag{8.78}$$

holds for all $z', z \in \mathcal{N}_z$, $v \in \mathcal{N}_v$ and $w \in \mathcal{N}_w$.

As before, the functions $G(z', z, v)$ and $H(w, v)$ are denoted by $G(z' \mid z, v)$ or $H(w \mid v)$, respectively, because they turn out to be conditional probabilities.

Definition 8.2 has an obvious interpretation. Equation (8.78) says that the output w does not depend on z and, hence, does not provide any information about the current automaton state. Furthermore Eq. (8.78) implies that w does not depend on

z' and, hence, the output does not provide any information about the successor state either.

Observability test. Before the consequences of Eq. (8.78) will be discussed it should be mentioned how this definition can be used to test whether a stochastic automaton is stochastically unobservable. It is easy to see that the "arbitrary" function G and H used in Eq. (8.78) are the conditional probabilities defined by Eqs. (8.16) and (8.17). From this fact it is clear that the decomposition of L in the form (8.78) is found (if it exists) by determining G and H according to Eqs. (8.16) and (8.17) and by testing whether Eq. (8.78) holds. Note that the function H obtained from Eq. (8.17) is not allowed to depend upon z, but $H(w \mid z_i, v) = H(w \mid z_j, v)$ has to hold for all $z_i, z_j \in \mathcal{N}_z$, $v \in \mathcal{N}_v$ and $w \in \mathcal{N}_w$.

Property of unobservable automata. The following lemma says that the definition of unobservability satisfies the aim to call an automaton observable if the current state can be determined more precisely by state observation than by simulation.

Lemma 8.2 *If the stochastic automaton is stochastically unobservable, then for all consistent I/O pairs (V, W) the results of state observation problem and of simulation are identical for all $k_h = 0, 1, 2, ...$:*

$$\begin{aligned} \mathrm{Prob}(z(k_h) \mid V(0 \ldots k_h), W(0 \ldots k_h)) &= \mathrm{Prob}(z(k_h) \mid V(0 \ldots k_h)) \quad (8.79) \\ \mathcal{Z}(k_h \mid k_h) &= \mathcal{Z}_{sim}(V(0 \ldots k_h)). \quad (8.80) \end{aligned}$$

Hence, the output sequence W does not include any information about the state trajectory of the automaton that cannot be obtained from the initial state probability $\mathrm{Prob}(z(0))$ and the input sequence V. In this case the observers described by Theorems 8.2 and 8.3 work as simulators.

Proof. Due to Eq. (8.78),

$$\begin{aligned} L(k_h) &= L(z(k_h+1), w(k_h) \mid z(k_h), v(k_h)) \\ &= G(z(k_h+1) \mid z(k_h), v(k_h)) \cdot H(w(k_h) \mid v(k_h)) \end{aligned}$$

holds and Eq. (8.53) yields

$$\begin{aligned} &\mathrm{Prob}(Z(0 \ldots k_h) \mid V(0 \ldots k_h), W(0 \ldots k_h)) \\ &= \frac{H(w(k_h) \mid v(k_h)) \cdot \ldots \cdot H(w(0) \mid v(0)) \cdot \sum_{z(k_h+1)} G(k_h) \cdot \ldots \cdot G(0) \cdot \mathrm{Prob}(z(0))}{H(w(k_h) \mid v(k_h)) \cdot \ldots \cdot H(w(0) \mid v(0)) \cdot \sum_{Z(0..k+1)} G(k_h) \cdot \ldots \cdot G(0) \cdot \mathrm{Prob}(z(0))} . \end{aligned}$$

As Eq. (8.20) holds, the sum in the denominator is 1 and the relation

$$\mathrm{Prob}(Z(0 \ldots k_h) \mid V(0 \ldots k_h), W(0 \ldots k_h)) = G(k_h - 1) \cdots G(0) \cdot \mathrm{Prob}(z(0))$$

is obtained, which is identical to the simulation result described by Eq. (8.24). □

Stochastic unobservability of a state set. The stochastic automaton may not satisfy the condition (8.78) for all $z \in \mathcal{N}_z$, but only for a subset $\mathcal{G}_z \subseteq \mathcal{N}_z$, where $\mathcal{G}_z$ should have at least two elements ($|\mathcal{G}_z| \geq 2$). Then a stochastic automaton is called *stochastically unobservable within a set* $\mathcal{G}_z$ if the relation

$$L_{\mathcal{G}_z}(z', w \mid z, v) = G_{\mathcal{G}_z}(z' \mid z, v) \cdot H_{\mathcal{G}_z}(w \mid v) \,, \tag{8.81}$$

holds for all $z, z' \in \mathcal{G}_z$, $v \in \mathcal{N}_v$, and $w \in \mathcal{N}_w$, with

$$L_{\mathcal{G}_z}(z', w \mid z, v) = \frac{L(z', w \mid z, v)}{\sum\limits_{z' \in \mathcal{G}_z} \sum\limits_{w} L(z', w \mid z, v)} \tag{8.82}$$

$$G_{\mathcal{G}_z}(z' \mid z, v) = \sum_{w} L_{\mathcal{G}_z}(z', w \mid z, v) \tag{8.83}$$

$$H_{\mathcal{G}_z}(w \mid v) = \sum_{z' \in \mathcal{G}_z} L_{\mathcal{G}_z}(z', w \mid z, v) \,. \tag{8.84}$$

In this definition the functions L, G and H appearing in Definition 8.2 are replaced by $L_{\mathcal{G}_z}$, $G_{\mathcal{G}_z}$ or $H_{\mathcal{G}_z}$, respectively. These functions are normalised as shown in Eqs. (8.82) – (8.84) so as to define conditional probability distributions. For these new functions, Eq. (8.81) is, in principle, the same as Eq. (8.78), but it has only to be satisfied for the states z, z' that belong to the set $\mathcal{G}_z$. Lemma 8.2 holds as long as the automaton remains within the set $\mathcal{G}_z$:

$$z(k) \in G_z \qquad \text{for } k = 0, 1, \ ..., \ k_h.$$

If the automaton is stochastically unobservable within the set $\mathcal{G}_z$ then the state observer acts as a simulator as long as the state remains in the set $\mathcal{G}_z$.

Stochastic observability. The stochastic observability can now be defined as the property of an automaton not to possess sets of unobservable states:

Definition 8.3 (Stochastic observability)
A stochastic automaton is called stochastically observable *if it does not possess any set $\mathcal{G}_z$ of stochastically unobservable states.*

Consequently, the stochastic automaton is called observable if it is possible to determine every state more precisely by state observation than by simulation. This fact is represented by the following corollary, which follows directly from Lemma 8.2 and Definition 8.3.

Corollary 8.2 *If the stochastic automaton is stochastically observable, the following relation holds:*

$$\mathcal{Z}(k_h \mid k_h) \subseteq \mathcal{Z}_{sim}(V(0...k_h)), \quad (k_h \geq 0). \tag{8.85}$$

Note that the equality sign may hold for some input sequence even if the automaton is observable. The reason for this is given in Section 8.3.10.

Remark 8.2 *Observability test*
The following remarks concern the test of a given stochastic automaton concerning observability. The definition of unobservable state sets implies that if the set $\mathcal{G}_z$ is stochastically unobservable then any subset $\tilde{\mathcal{G}}_z \subset \mathcal{G}_z$ is stochastically unobservable as well, because Eq. (8.81) holds not only for all $z, z' \in \mathcal{G}_z$ but also for all $z, z' \in \tilde{\mathcal{G}}_z$ and the normalisation carried out in Eqs. (8.82) – (8.84) does not influence this result. Hence, the test of a stochastic automaton starts with testing all pairs $z, \bar{z}$ whether or not they are unobservable sets. If no such pair is found, the stochastic automaton does not possess any unobservable set and is, therefore, stochastically observable. If such pairs exist, larger unobservable state sets can be obtained (if they exist) from combinations of such unobservable pairs according to the following corollary. □

Corollary 8.3 *A stochastic automaton is stochastically unobservable within a set $\mathcal{G}_z$ of at least three states ($|\mathcal{G}_z| \geq 3$) if and only if the stochastic automaton is stochastically unobservable within all subsets $\mathcal{G}_z^i \subset \mathcal{G}_z$ of two states ($|\mathcal{G}_z^i| = 2$).*

Therefore, the search for stochastically unobservable sets of states can be reduced to the test of all pairs of states. The stochastic automaton is stochastically observable if no unobservable pair of states is found.

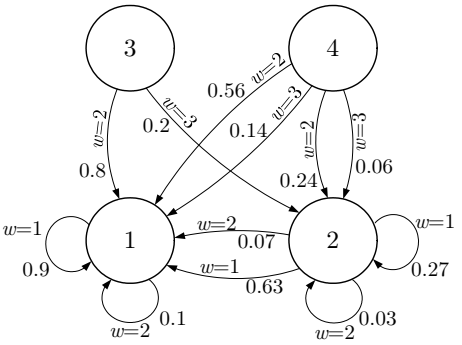


Fig. 8.13. Stochastic automaton with stochastically unobservable set $\{1, 2\}$

Example 8.5 *Observability of stochastic automata*

Consider the stochastic automaton shown in Fig. 8.13, which has the only input $v = 1$. The automaton is not stochastically unobservable according to Definition 8.2. However, the state set $\mathcal{G}_z = \{1, 2\}$ is stochastically unobservable. This can be verified by means of Eq. (8.78) as shown in Table 8.3. Note that $G_{\mathcal{G}_z} \cdot H_{\mathcal{G}_z}$ is identical to $L_{\mathcal{G}_z}$ for all z, z', w and v. □

8.3.10 Distinguishing inputs

In this section, it is investigated under what conditions the observer improves its result by processing the k_h-th I/O pair $(v(k_h), w(k_h))$ in comparison with the result

Table 8.3 Test for stochastic unobservability of the state set $\mathcal{G}_z = \{1, 2\}$

		$L_{\mathcal{G}_z}$		$G_{\mathcal{G}_z}$		$H_{\mathcal{G}_z}$		$G_{\mathcal{G}_z} \cdot H_{\mathcal{G}_z}$	
		$z=1$	$z=2$	$z=1$	$z=2$	$z=1$	$z=2$	$z=1$	$z=2$
$z'=1$	$w=1$	0.9	0.63	1.0	0.7	0.9	0.9	0.9	0.63
	$w=2$	0.1	0.07			0.1	0.1	0.1	0.07
	$w=3$	0	0			0	0	0	0
$z'=2$	$w=1$	0	0.27	0	0.3	0.9	0.9	0	0.27
	$w=2$	0	0.03			0.1	0.1	0	0.03
	$w=3$	0	0			0	0	0	0

obtained for the $(k_h - 1)$-st step. This analysis uses the recursive formulation of the observer given in Theorem 8.3 and compares the observation result with the simulation results obtained by the recursion (8.27). It will be shown that even if the automaton is observable there exist input values v for which the next recursion step of the observer yields the same result as the next recursion step of the simulator. Hence, the movement of the automaton under this input cannot contribute to an improvement of the knowledge about the current state.

To start the analysis, it is assumed that both simulation and observation have obtained the same state probability distribution at time k_h

$$\begin{aligned}&\mathrm{Prob}(z(k_h)=z \mid V(0\,...\,k_h-1), W(0\,...\,k_h-1)) = \\ &\quad = \mathrm{Prob}(z(k_h)=z \mid V(0\,...\,k_h-1)) \\ &\quad =: p(z \mid k_h-1)\,,\end{aligned}$$

where both observation and simulation have used the information available until time $k_h - 1$. Consequently, both methods yield the same state set at time k_h:

$$\begin{aligned}\mathcal{Z}(k_h-1 \mid k_h-1)) &= \mathcal{Z}_{\mathrm{sim}}(V(0\,...\,k_h-1)) \\ &= \{z \,:\, p(z \mid k_h-1) > 0\}\,.\end{aligned}$$

The set of successor states that the automaton can reach at time k_h+1 if it is subjected to input $v(k_h) = \bar{v}$ is given as follows:

$$\mathcal{Z}_{\mathrm{sim}}(V(0\,...\,k_h)) = \{z \mid G(z \mid \tilde{z}, \bar{v}) > 0 \text{ for some } \tilde{z} \in \mathcal{Z}_{\mathrm{sim}}(V(0\,...\,k_h-1))\}\,.$$

In the following the question should be answered under what condition the k_h-th observation step yields a set $\mathcal{Z}(k_h \mid k_h)$ which is a proper subset of $\mathcal{Z}_{\mathrm{sim}}(V(0\,...\,k_h))$. The answer can be obtained from the considerations made in Section 8.3.9. Lemma 8.2 has shown that observation yields the same result as simulation if the automaton is unobservable and, hence, the behavioural relation can be decomposed according to Eq. (8.78). This result is applied here for the fixed input $\bar{v}$. If the decomposition (8.78) is possible for the input $v(k_h) = \bar{v}$, simulation and observation lead to the same set of states $z(k_h$+1). This result is summarised in the following corollary. It shows that the equality sign can hold in Eq. (8.85).

Corollary 8.4 *Assume that the state probability distribution* $p(z \mid k_h - 1)$ *is known and the stochastic automaton generates the output* $w(k_h) = \bar{w}$ *for the input* $v(k_h) = \bar{v}$*. Then simulation and state observation yield the same state probability distribution*

$$\mathrm{Prob}(z(k_h{+}1) \mid V(0 \ldots k_h), W(0 \ldots k_h)) = \mathrm{Prob}(z(k_h{+}1) \mid V(0 \ldots k_h))$$

for the $(k_h{+}1)$*-st automaton state if and only if the relation*

$$L(z', \bar{w} \mid z, \bar{v}) = G(z' \mid z, \bar{v}) \cdot H(\bar{w} \mid \bar{v}) \tag{8.86}$$

holds with some constant $H(\bar{w} \mid \bar{v})$ *for all* $z \in \mathcal{Z}(k_h - 1 \mid k_h - 1)$ *and* $z' \in \mathcal{Z}_{sim}(V(0 \ldots k_h))$*.*

The condition (8.86) can be tested in each recursion step of the observation algorithm in order to indicate whether the observer works really as an observer or merely as a simulator. If the condition is satisfied, the observer yields the same state probability distribution $\mathrm{Prob}(z(k_h{+}1) \mid V(0 \ldots k_h))$ as a simulation step described by Eq. (8.27) (with k_h replacing $k_h - 1$).

This result has interesting consequences concerning the choice of the input. Even if the stochastic automaton is observable according to Definition 8.3, not all individual I/O pairs $(\bar{v}, \bar{w})$ lead to an improvement of the observation result compared to the corresponding simulation step. The reason for this is given by the fact that the observability definition claims that the decomposition of the behavioural relation L according to Eq. (8.78) is impossible for all states $z, z' \in \mathcal{N}_z$, all input values $v \in \mathcal{N}_v$ and all output values $w \in \mathcal{N}_w$. However, such a decomposition may be possible for the set $\tilde{\mathcal{N}}_v \subset \mathcal{N}_v$ of input values even if it is impossible for all $v \in \mathcal{N}_v$. If the stochastic automaton gets, for some reason, only input values from $\tilde{\mathcal{N}}_v$ the solution of the observation problem is not better than the simulation result (cf. Lemma 8.2), although the stochastic automaton is observable. Therefore, it is important to know, which input should be used in order to get an improved observation result. These input values are called *distinguishing*. For such input values $\bar{v}$ the decomposition (8.78) is impossible for $v = \bar{v}$ and all z', z and w.

As a consequence, for every given state set $\mathcal{G}_z$ it is possible to partition the input set $\mathcal{N}_v$ into two sets $\tilde{\mathcal{N}}_v(\mathcal{G}_z)$ and $\bar{\mathcal{N}}_v(\mathcal{G}_z)$ such that the decomposition of L is possible for all $v \in \tilde{\mathcal{N}}_v(\mathcal{G}_z)$:

$$\begin{aligned} L_{\mathcal{G}_z}(z', w \mid z, v) \quad &= \quad G_{\mathcal{G}_z}(z' \mid z, v) \cdot H_{\mathcal{G}_z}(w \mid v) \\ &\qquad \text{for all } \; z', z \in \mathcal{G}_z, \; v \in \tilde{\mathcal{N}}_v(z), \; w \in \mathcal{N}_w, \end{aligned} \tag{8.87}$$

whereas such a decomposition of L is impossible for all $v \in \bar{\mathcal{N}}_v(\mathcal{G}_z)$. From Lemma 8.2 it is obvious that the state observer cannot improve its result at time k_h for states $z(k_h), z(k_h + 1) \in \mathcal{G}_z$ if the stochastic automaton obtains the input $v(k_h) \in \tilde{\mathcal{N}}_v(\mathcal{G}_z)$. Hence, $\bar{\mathcal{N}}_v(\mathcal{G}_z)$ is the set of distinguishing inputs. Only if an input $v \in \bar{\mathcal{N}}_v(\mathcal{G}_z)$ is applied, the state observer gets enough information for determining the state probability distribution better than it can be determined by simulation.

In an application where the state of the system should be found as quickly as possible, the input has to be chosen at every time instant from the current set of

distinguishing inputs. This set has to be determined at time k_h as follows. Select the set $\mathcal{G}_z$ to include all possible current states $z(k_h) \in \mathcal{Z}(k_h|k_h-1)$ and all possible successor states $z(k_h+1)$, for which $L(z(k_h+1), w|z(k_h), v)$ holds for some v and w. Then $\bar{\mathcal{N}}_v(\mathcal{G}_z)$, which is obtained as described above, is the set of distinguishing inputs form which the current input $V(k_h)$ should be selected. Note that the set $\bar{\mathcal{N}}_v(\mathcal{G}_z)$ changes from one time step to the next because the set of current state changes. It depends on the practical circumstances whether the input can really be chosen with the aim of state observation or whether the selection of the input has to satisfy other control aims. In any case, the set of distinguishing inputs is the set of preferred input signals as far as state observation is concerned.

If the stochastic automaton is stochastically observable, for every state z there exists at least one distinguishing input v. If, in particular, the decomposition (8.78) is impossible for all $v \in \mathcal{N}_v$, the stochastic automaton is called *uniformly stochastically observable*. Then, every input is distinguishing and the observation result improves in every recursion step.

Example 8.6 *Distinguishing input values*

Figure 8.14 shows the automaton graph of a stochastic automaton with two input symbols $v \in \{1, 2\}$ and two output symbols $w \in \{1, 2\}$. Only one output symbol is distinguishing. The edges in Fig. 8.14 are labelled by the I/O pair v/w for which they occur and by the corresponding probability. It can be seen that for $v = 1$ the relation (8.86) holds with

$$H(\bar{w}=1 \mid \bar{v}=1) = 0.5 \quad \text{and} \quad H(\bar{w}=2 \mid \bar{v}=1) = 0.5$$

and

$$G(z'=1 \mid z=1, \bar{v}=1) = 0.8, \qquad G(z'=1 \mid z=2, \bar{v}=1) = 0.2$$
$$G(z'=2 \mid z=1, \bar{v}=1) = 0.2, \qquad G(z'=2 \mid z=2, \bar{v}=1) = 0.8\,.$$

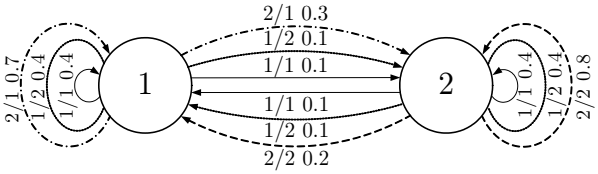


Fig. 8.14. Automaton graph of the example

Figures 8.15 and 8.16 show the effect of the non-distinguishing input on the observation result. Starting with a uniform initial state distribution over the whole state set the observation result is shown in Fig. 8.16. It is identical to the simulation result as long as the input is $v = 1$ (time steps $k = 0, 1, \ldots, 7$). Because of the symmetry of $G(z' \mid z, \bar{v})$ the simulation yields uniform distributions for these times steps. Between time steps $k = 8$ and $k = 13$ the distinguishing input $v = 2$ is applied and, hence, the observation result improves immediately. Instead of a uniform probability distribution, Fig. 8.16 shows that the observation results in high probability for the state $z = 2$ at $k = 8, 9$ and for the state $z = 1$ at $k = 10...13$. When the input returns to the non-distinguishing input symbol $v = 1$ at time step $k = 14$ the observation falls back into a simulation, quickly loosing all state information. □

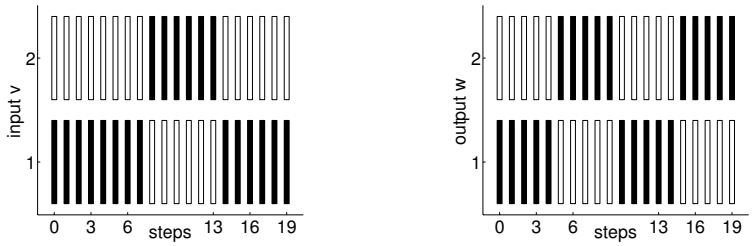


Fig. 8.15. Sequences of input (left) and output symbols (right)

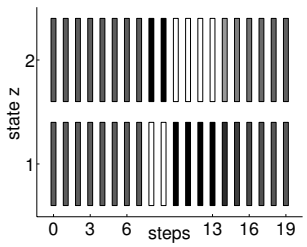


Fig. 8.16. Observation result

8.4 Diagnosis of stochastic automata

8.4.1 Principle of consistency-based diagnosis

This section shows how the diagnostic problem can be solved for a stochastic automaton. To do so, it has to be known how the fault f changes the behaviour of the stochastic automaton. Therefore, in this section the dependence of the automaton and its behaviour upon the fault is denoted by $\mathcal{S}(f)$ and $\mathcal{B}(f, k_h)$. Both symbols will be explained in more detail in Section 8.2.5.

Problem 8.2 (Diagnostic problem for the stochastic automaton)

Given: *Input sequence* $V(0 \dots k_h)$.
Output sequence $W(0 \dots k_h)$.
Stochastic automaton $\mathcal{S}(f)$.

Find: *Fault* f.

The idea of consistency-based diagnosis is to ask whether the measured I/O pair is consistent with the automaton $\mathcal{S}(f)$. If the answer is in the affirmative, the fault f

may have occurred. In case of a negative answer the conclusion is that the automaton is not subjected to the fault f.

Since the behaviour $\mathcal{B}$ of the stochastic automaton is the set of all I/O pairs that are consistent with the automaton, the automaton may be subjected to the fault f whenever the measured I/O pair belongs to its behaviour $\mathcal{B}(f, k_h)$:

$$(V(0 \ldots k_h), W(0 \ldots k_h)) \in \mathcal{B}(f, k_h). \tag{8.88}$$

The fault f is known not to have occurred if the relation

$$(V(0 \ldots k_h), W(0 \ldots k_h)) \notin \mathcal{B}(f, k_h) \tag{8.89}$$

holds because then the I/O pair is inconsistent with $\mathcal{S}(f)$. In particular, for the automaton $\mathcal{S}(f_0)$ of the faultless system, the inconsistency (8.89) for $f = f_0$ implies that some fault $f \neq f_0$ has occurred.

In summary, diagnosing the stochastic automaton means to answer the question:

> Does there exist a fault f such that the I/O pair $(V(0 \ldots k_h), W(0 \ldots k_h))$ is consistent with the stochastic automaton?

The result is a set $\mathcal{F}(k_h)$ of *fault candidates*, which are those faults $f \in \mathcal{N}_f$ for which the I/O pair with time horizon k_h is consistent with the model $\mathcal{S}(f)$.

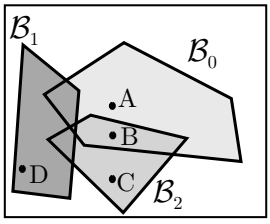


Fig. 8.17. Consistency-based diagnosis

In the graphical interpretation given in Fig. 8.17, which applies the ideas of Fig. 1.2 on p. 4 to the diagnostic problem for the stochastic automaton, the three behaviours

$$\begin{aligned} \mathcal{B}_0 &= \mathcal{B}(f_0, k_h) \\ \mathcal{B}_1 &= \mathcal{B}(f_1, k_h) \\ \mathcal{B}_2 &= \mathcal{B}(f_2, k_h) \end{aligned}$$

are considered which are relevant for the faultless system ($f = f_0$) and for the two faults f_1 and f_2. The points A - D represent measured I/O pairs. Obviously, A is consistent only with $\mathcal{B}_0$, which means that if A is measured the system is known not to be subjected to the faults f_1 and f_2. If it is additionally known that the fault is restricted to the set $\mathcal{N}_f = \{f_0, f_1, f_2\}$, a further conclusion is that the automaton is faultless. In this situation, f_0 is the only fault candidate:

$$\mathcal{F}(k_h) = \{f_0\}.$$

Since B is consistent with $\mathcal{B}_0$ and with $\mathcal{B}_2$, it cannot be decided whether the system is faultless or whether fault f_2 has occurred. However, f_1 can be excluded as a possible fault. Hence, the set of fault candidates is

$$\mathcal{F}(k_h) = \{f_0, f_2\}.$$

The I/O pairs C and D show that the system is faulty with the possible fault f_2 or f_1, respectively:

$$\mathcal{F}(k_h) = \{f_1, f2\}.$$

There is often no unique solution to the diagnostic problem but the set of fault candidates includes more than one element $f \in \mathcal{N}_f$.

In order to distinguish between highly probable faults and faults with low probability all of which are elements of the set $\mathcal{F}(k_h)$, the solution to the diagnostic problem will not only be an answer to the test (8.88) but it should additionally give the probability distribution

$$\mathrm{Prob}(f \mid V(0 \ldots k_h), W(0 \ldots k_h))$$

for all $f \in \mathcal{N}_f$, where $\mathcal{N}_f$ denotes the set of possible faults. Like for the observation problem, this probability will be abbreviated by

$$\mathrm{Prob}(f \mid k_h) ::= \mathrm{Prob}(f \mid V(0 \ldots k_h), W(0 \ldots k_h)) \,.$$

Obviously,

$$(V(0 \ldots k_h), W(0 \ldots k_h)) \in \mathcal{B}(f, k_h)$$

holds if and only if $\mathrm{Prob}(f \,|\, k_h) > 0$ is valid.

8.4.2 Consistency-based diagnosis of stochastic automata

Figure 8.9 shows that the fault $f(k)$ is an immeasurable internal state of the automaton $\tilde{\mathcal{S}}$. Consequently, the diagnostic problem can be solved by observing the internal state $\tilde{z}(k_h)$ and by selecting all those faults $f(k_h)$ that are part of the states $\tilde{z}(k_h) = (z(k_h),\ f(k_h))'$ that occur with a positive probability (cf. Eq. (8.45)). A direct application of Theorem 8.3 yields the following observation result:

$$\mathrm{Prob}(\tilde{z}(k_h) \,|\, k_h) = \frac{\sum\limits_{\tilde{z}(k_h+1)} \tilde{L}(k_h) \cdot \mathrm{Prob}(\tilde{z}(k_h) \mid k_h - 1)}{\sum\limits_{\tilde{z}(k_h), \tilde{z}(k_h+1)} \tilde{L}(k_h) \cdot \mathrm{Prob}(\tilde{z}(k_h) \mid k_h - 1)}, \qquad (k_h \geq 0)$$

with

$$\mathrm{Prob}(\tilde{z}(k_h) \mid k_h - 1) = \frac{\sum\limits_{\tilde{z}(k_h-1)} \tilde{L}(k_h - 1) \cdot \mathrm{Prob}(\tilde{z}(k_h - 1) \mid k_h - 2)}{\sum\limits_{\tilde{z}(k_h), \tilde{z}(k_h-1)} \tilde{L}(k_h - 1) \cdot \mathrm{Prob}(\tilde{z}(k_h - 1) \mid k_h - 2)} \qquad (k_h > 0)$$

$$\mathrm{Prob}(\tilde{z}(0) \mid -1) := \mathrm{Prob}(\tilde{z}(0)) \qquad (k_h = 0),$$

where, as before, the abbreviation

$$\tilde{L}(k_h) = \tilde{L}(\tilde{z}(k_h+1), w(k_h) \mid \tilde{z}(k_h), v(k_h))$$

is used. The diagnostic result follows from

$$\begin{aligned} \mathrm{Prob}(f(k_h) \,|\, k_h) &= \sum_{z(k_h) \in \mathcal{N}_z} \mathrm{Prob}(\tilde{z}(k_h) \,|\, k_h) \\ &= \sum_{z(k_h) \in \mathcal{N}_z} \mathrm{Prob}\left(\begin{pmatrix} z(k_h) \\ f(k_h) \end{pmatrix} \,|\, k_h \right). \end{aligned}$$

This result can be simplified because $\tilde{L}(k_h)$ can be decomposed into $L(k_h)$ and

$$G_f(k_h) := G_f(f(k_h+1) \mid f(k_h))$$

as described by Eq. (8.44):

$$\tilde{L}(k_h) = L(k_h) \cdot G_f(k_h).$$

Furthermore, by assuming the stochastic independence of the a-priori values of both components of $\tilde{z}$ in Eq. (8.45), the initial state probability distribution $\mathrm{Prob}(\tilde{z}(0))$ can be represented as the product of the initial state probability distributions of $z(0)$ and $f(0)$:

$$\mathrm{Prob}(\tilde{z}(0)) = \mathrm{Prob}(z(0)) \cdot \mathrm{Prob}(f(0)).$$

With this reformulation the following result is obtained where the abbreviation

$$\begin{aligned} &\mathrm{Prob}(f(k_h), z(k_h) \,|\, k_h - 1) := \\ &\mathrm{Prob}(f(k_h), z(k_h) \,|\, V(0 \ldots k_h - 1), W(0 \ldots k_h - 1)) \end{aligned}$$

is used.

Theorem 8.5 (Solution to the diagnostic problem)
Consider a stochastic automaton with the initial state probability distribution $\mathrm{Prob}(z(0))$ *and the a-priori fault distribution* $\mathrm{Prob}(f(0))$. *If the relation*

$$\sum_{f(k_h-1),z(k_h-1)} L(k_h-1)\cdot G_f(k_h-1)\cdot \mathrm{Prob}(f(k_h-1),z(k_h-1)\,|\,k_h-2)>0 \tag{8.90}$$

holds, the solution to the diagnostic problem is given by

$$\mathrm{Prob}(f(k_h)\mid k_h)=\frac{\sum\limits_{\substack{f(k_h+1)\\ z(k_h+1),z(k_h)}} L(k_h)\cdot G_f(k_h)\cdot \mathrm{Prob}(f(k_h),z(k_h)\mid k_h-1)}{\sum\limits_{\substack{f(k_h),f(k_h+1)\\ z(k_h),z(k_h+1)}} L(k_h)\cdot G_f(k_h)\cdot \mathrm{Prob}(f(k_h),z(k_h)\mid k_h-1)} \qquad (k_h\geq 0) \tag{8.91}$$

with

$$\mathrm{Prob}(f(k_h),z(k_h)\mid k_h-1)= \frac{\sum\limits_{f(k_h-1),z(k_h-1)} L(k_h-1)\cdot G_f(k_h-1)\cdot \mathrm{Prob}(f(k_h-1),z(k_h-1)\mid k_h-2)}{\sum\limits_{\substack{f(k_h),f(k_h-1)\\ z(k_h),z(k_h-1)}} L(k_h-1)\cdot G_f(k_h-1)\cdot \mathrm{Prob}(f(k_h-1),z(k_h-1)\mid k_h-2)} \qquad (k_h>0) \tag{8.92}$$

$$\mathrm{Prob}(z(0)\mid -1):=\mathrm{Prob}(f(0))\cdot \mathrm{Prob}(z(0)) \qquad (k_h=0). \tag{8.93}$$

Hence, the set of possible faults is given by

$$\mathcal{F}(k_h)=\{f(k_h)\,:\,\mathrm{Prob}(f(k_h)\mid k_h)>0\}. \tag{8.94}$$

Due to the main idea of consistency-based diagnosis, the following corollary holds:

Corollary 8.5 (Diagnostic result for the stochastic automaton)

- *The automaton is known to be subjected to some fault $f(k_h) \in \mathcal{F}(k_h)$ with $\mathcal{F}(k_h)$ given by Eq. (8.94).*
- **Fault detection:** *If $f_0 \notin \mathcal{F}(k_h)$ holds where f_0 symbolises the faultless automaton and the automaton is known to be subjected to some fault $f(k_h) \in \mathcal{N}_f, \quad f \neq f_0$.*
- **Fault identification:** *If $\mathcal{F}(k_h) = \{f_i\}$ holds, the automaton is known to be subjected to the fault $f(k_h) = f_i$ provided that the set $\mathcal{N}_f$ includes all possible faults.*

The first assertion holds under the assumption that the current fault belongs to the set $\mathcal{N}_f$. A fault is diagnosed according to the second assertion, if f_0 is not a fault candidate. However, as long as $\mathcal{F}(k_h)$ has more than one element, the fault cannot be unambiguously determined. The probability $\mathrm{Prob}(f(k_h) \,|\, k_h)$ describes with which probability the faults $f \in \mathcal{F}(k_h)$ occur. The third assertion concerns the case that $\mathcal{F}(k_h)$ is a singleton. Then the fault is unambiguously determined provided that $f \in \mathcal{N}_f$ holds.

If the condition (8.90) is not satisfied, the I/O pair is inconsistent with the stochastic automaton for all faults $f(k_h) \in \mathcal{N}_f$. Then, an unmodelled fault occurred or the a-priori information about the state and fault distributions is wrong.

8.4.3 Diagnostic algorithm

Theorem 8.5 leads to the following diagnostic algorithm, which is developed in close analogy to the observation algorithm presented in Section 8.3.7. It uses the following abbreviations:

$$h(f(k_h), z(k_h)) := \sum_{f(k_h+1), z(k_h+1)} L(k_h) \cdot G_f(k_h) \cdot \mathrm{Prob}(z(k_h) \mid V(0 \ldots k_h - 1), W(0 \ldots k_h - 1))$$

$$p_k(f(k_h), z(k_h)) := \mathrm{Prob}(f(k_h), z(k_h) \mid V(0 \ldots k_h), W(0 \ldots k_h))$$

$$p_r(f(k_h+1), z(k_h+1)) := \mathrm{Prob}(f(k_h+1), z(k_h+1) \mid V(0 \ldots k_h), W(0 \ldots k_h))$$

Algorithm 8.3 *Diagnosis of stochastic automata*

Given: Stochastic automaton $\mathcal{S}$ and fault model $\mathcal{S}_f$.
Initial state probability distribution $\mathrm{Prob}(\hat{z}(0))$.
Initial fault probability distribution $\mathrm{Prob}(\hat{f}(0))$.

Initialisation: $p_r(f,z) = \mathrm{Prob}(\hat{f}_p(0) = f) \cdot \mathrm{Prob}(\hat{z}_p(0) = z)$ for all $f \in \mathcal{N}_f$ and $z \in \mathcal{N}_z$
$k_h = 0$.

Loop:

1. Measure the current input v and output w.
2. For all $f \in \mathcal{N}_f$ and $z \in \mathcal{N}_z$ determine $h(f,z) = \sum_{\bar{f},\bar{z}} L(\bar{z}, w \mid z, f, v) \cdot G_f(\bar{f} \mid f) \cdot p_r(f,z)$.
3. If $\sum_{f,z} h(f,z) = 0$ holds, stop the algorithm (inconsistent I/O pair or wrong initial distributions).
4. For all $f \in \mathcal{N}_z$ and $z \in \mathcal{N}_z$ determine
$$p_k(f,z) = \frac{h(f,z)}{\sum\limits_{f,z} h(f,z)}.$$
5. For all $f \in \mathcal{N}_f$ and $z \in \mathcal{N}_z$ determine
$$p_r(f,z) = \frac{\sum\limits_{\bar{f},\bar{z}} L(z, w \mid \bar{z}, v) \cdot G_f(f \mid \bar{f}) \cdot p_r(\bar{f},\bar{z})}{\sum\limits_{f,z} h(f,z)}.$$
6. Determine $Prob(f(k_h) = f \mid k_h) = \sum_z p_k(f,z)$.
7. Determine $\mathcal{F}(k_h)$ according to Eq. (8.94).
8. $k_h := k_h + 1$
Continue with Step 1.

Result: $\mathrm{Prob}(f(k_h) \mid k_h)$ and $\mathcal{F}(k_h)$ for increasing time horizon k_h.

The algorithm uses the initial fault probability distribution $\hat{p}(f(0))$ which replaces the true distribution $\mathrm{Prob}(f(0))$ if this distribution is not known (cf. Section 8.3.6).

Example 8.7 *Diagnosis of a stochastic automaton*

Consider the stochastic automaton shown in Fig. 8.18 with two faults $f = 1$ and $f = 2$. The automaton states $\tilde{z}$ are composed of the automaton state and the fault according to Eq. (8.45). The initial state $z_0 = 1$ is known, which gives the initial state probability distribution

$$\mathrm{Prob}(z_p(0) = z) = \begin{cases} 1 & \text{for } z = 1 \\ 0 & \text{else.} \end{cases}$$

For the unknown faults, a uniform distribution

$$\mathrm{Prob}(f_p(0) = f) = 0.5 \quad \text{for } f = 1, 2$$

is assumed. This means that the initial state/fault distribution is given by

$$\mathrm{Prob}(\tilde{z}_p(0) = \tilde{z}) = \mathrm{Prob}\begin{pmatrix} z_p(0) = z \\ f_p(0) = f \end{pmatrix} = \begin{cases} 0.5 & \text{for } z = 1 \text{ and } f \in \{1, 2\} \\ 0 & \text{else .} \end{cases}$$

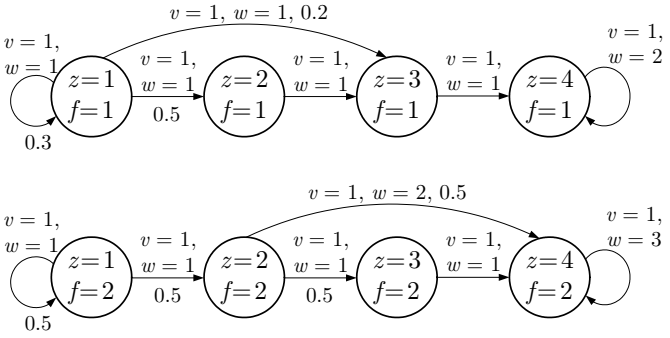


Fig. 8.18. Automaton graph for diagnosis

Table 8.4 shows the diagnostic result for the I/O pair

$$V(0 \ldots 3) = (1, 1, 1, 1, 1)$$
$$W(0 \ldots 3) = (1, 1, 1, 3, 3).$$

For the initial state $z_0 = 1$, no information about the fault can be obtained at time $k_h = 0$, because for both faults the automaton yields the output $w = 1$ for the given input $v = 1$ with certainty. For $k_h = 1$ the system can be in the states

$$\begin{pmatrix} z \\ f \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \begin{pmatrix} 3 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \end{pmatrix} \text{ or } \begin{pmatrix} 2 \\ 2 \end{pmatrix}.$$

The I/O pair $(v(1) = 1, w(1) = 1)$ excludes the state transition

$$\begin{pmatrix} 2 \\ 2 \end{pmatrix} \rightarrow \begin{pmatrix} 4 \\ 2 \end{pmatrix}$$

and makes the fault $f = 2$ less probable than the fault f_1 (cf. Fig. 8.18). From Table 8.4 it can be seen, that, therefore, at $k = 1$ some information about the fault is obtained by the I/O pair and the diagnostic result differs from a simulation of the initial fault distribution. After

Table 8.4 Probability distribution $\mathrm{Prob}(f(k_h) \mid k_h)$

	$k_h = 0$ $V(0 \ldots 0) = (1)$ $W(0 \ldots 0) = (1)$	$k_h = 1$ $V(0 \ldots 1) = (1,1)$ $W(0 \ldots 1) = (1,1)$	$k_h = 2$ $V(0 \ldots 2) = (1,1,1)$ $W(0 \ldots 2) = (1,1,1)$
f	$\mathrm{Prob}(f(0)\|0)$	$\mathrm{Prob}(f(1)\|1)$	$\mathrm{Prob}(f(2)\|2)$
1	0.5	0.5714	0.5614
2	0.5	0.4286	0.4386

	$k_h = 3$ $V(0 \ldots 3) = (1,1,1,1)$ $W(0 \ldots 3) = (1,1,1,3)$	$k_h = 4$ $V(0 \ldots 3) = (1,1,1,1,1)$ $W(0 \ldots 3) = (1,1,1,3,3)$
f	$\mathrm{Prob}(f(3)\|3)$	$\mathrm{Prob}(f(4)\|4)$
1	0.0	0.0
2	1.0	1.0

$w(3) = 3$ has been measured, the fault $f = 2$ is unambiguously found because this output is generated only if the fault $f = 2$ occurs. □

Discussion of the diagnostic results. The diagnostic algorithm yields the set $\mathcal{F}(k_h)$ of fault candidates for time k_h. Each of the fault separately "explains" why the measured I/O pair occurs. In addition to this set, the probability distribution $\mathrm{Prob}(f(k_h) \mid k_h)$ evaluates with which probability each fault candidate represents the real fault.

Under practical circumstances, several heuristic extensions can be made. For example, a threshold s can be fixed and only those faults that occur with a probability higher than s are announced to the human operator in the control room. Then the threshold can be used to adapt the result of the diagnostic algorithm to the certainty with which the behavioural relation of the automaton is known and to the degree of danger that the different faults may have on the system performance. This adaptation of the diagnostic results is analogous to the use of thresholds in the residual evaluation of quantitative diagnostic methods described in Chapter 6.

Automata with fixed faults. In applications, the diagnostic algorithm may have to use the same computing resources as other algorithms. Then it is only temporarily invoked. Under this condition it may be reasonable to assume that the fault does not change during the run of the diagnostic algorithm. Either the system is in the normal operation mode ($f = f_0$) or the fault happened to exist before the algorithm is started.

This situation simplifies the diagnostic problem in two respects. First, no fault model $\mathcal{S}_f$ has to be set up. As the fault is assumed not to change, the state transition relation

$$G_f(f' \mid f) = \begin{cases} 1 & \text{for } f' = f \\ 0 & \text{else} \end{cases} \tag{8.95}$$

is used for all faults $f \in \mathcal{N}_f$. Second, the algorithm simplifies because instead of a fault sequence $F(0 \ldots k_h)$ only a constant fault f has to be considered.

Diagnosis of non-deterministic automata. If no probabilistic information about the movement of the automaton is available, the diagnostic algorithm can also be applied to the non-deterministic automaton. Like for the observation problem, the predicate Prob has to be replaced by Poss in all formulas and, consequently, instead of the equations for determining of probability distributions formulas for determining sets of automaton states and faults are obtained. This reformulation is similar to what has been done in Section 8.3.8 for the observation problem.

8.4.4 Diagnosability of stochastic automata

When solving practical problems, the diagnostic algorithm should provide a set $\mathcal{F}(k_h)$ which is a singleton for a possibly small time horizon k_h. Whether or not this is possible, depends on the diagnosability of the stochastic automaton, which is investigated in this section. Note that the diagnosablity is a system property, which depends upon the system dynamics described by the behavioural relation of the automaton and by the measured signals v and w, but does not refer to the diagnostic method applied. Diagnosability claims that the fault f has to be found by *appropriately* using all the information available.

It is not easy to find conditions under which the system is diagnosable, because the question whether a fault can be detected does not only depend on the system dynamics but also on the initial state and on the input sequence. However, the frequent discussions among theoreticans and people from different application fields on "hidden faults" that do not influence the measurement sequence and, thus, cannot be found by any diagnostic algorithm, and discussions on the fact that different faults have to bring about different effects on the system behaviour if they should be discriminated, show that diagnosability is an important practical issue.

In this section the results on the observability of stochastic automata presented in Section 8.3.9 will be used to define and analyse the diagnosability of automata. Like in Section 8.3.9 the starting point is the investigation under what conditions the automaton is not diagnosable.

Definition 8.4 (Stochastic undiagnosability)
A stochastic automaton $\tilde{S}$ with behavioural relation

$$\tilde{L}(z', f', w \mid z, v, f) = L(z', w \mid z, v, f) \cdot G_f(f' \mid f)$$

is called stochastically undiagnosable *if it satisfies the property*

$$L(z', w \mid z, v, f) = L(z', w \mid z, v) \tag{8.96}$$

for all $z', z \in \mathcal{N}_z$, $w \in \mathcal{N}_w$, $v \in \mathcal{N}_v$ and $f \in \mathcal{N}_f$.

Clearly, under the condition (8.96) the state and output sequences of the stochastic automaton are independent of the fault f, because the fault does no longer appear in L and, hence, the fault cannot be "seen" from the measured I/O pair. In analogy to Lemma 8.2 it can be proved that for undiagnosable automata the diagnostic result coincides with the result obtained by simulation of the faulty behaviour:

Lemma 8.3 *If the stochastic automaton is stochastically undiagnosable, then for all input sequences V and for all output sequences W the diagnostic result is identical to the simulation result:*

$$\mathrm{Prob}(f(k_h) \mid V(0 \ldots k_h), W(0 \ldots k_h)) = \mathrm{Prob}(f(k_h) \mid V(0 \ldots k_h)) \,. \tag{8.97}$$

The left-hand side of Eq. (8.97) is the result obtained from the diagnostic algorithm. As the fault f does not depend on the input v, the right-hand side of Eq. (8.97) is given by the relation

$$\begin{aligned}
&\mathrm{Prob}(f(k_h) \mid V(0 \ldots k_h)) \\
&= \mathrm{Prob}(f(k_h)) \\
&= \sum_{F(0 \ldots k_h - 1)} G_f(f(k_h) \mid f(k_h - 1)) \cdot G_f(f(k_h - 1) \mid f(k_h - 2)) \cdot \ldots \\
&\qquad\qquad \cdot G_f(f(1) \mid f(0)) \cdot \mathrm{Prob}(f(0)) \,,
\end{aligned}$$

which predicts the state of the autonomous fault model $\mathcal{S}_f$. The fault changes with increasing time horizon k_h and so does the probability distribution

$$\mathrm{Prob}(f(k_h) \mid V(0 \ldots k_h)).$$

However, the only information used for simulation is the state transition relation G_f of the fault model. As the diagnostic algorithm uses further information given by the output sequence W it is reasonable to expect that the diagnostic result is better than the simulation result. The lemma says that for stochastically undiagnosable automata this expectation is not met. The diagnostic algorithm cannot improve the simulation result.

A given stochastic automaton is generally not completely stochastically undiagnosable according to Eq. (8.96), but there may exist one or more state sets $\mathcal{G}_z \subset \mathcal{N}_z$ and one or more fault sets $\mathcal{G}_f \subset \mathcal{N}_f$ such that the behaviour within the set $\mathcal{G}_z$ does not depend on the faults $f \in \mathcal{G}_f$. Then Eq. (8.96) does not hold for all z, z' and f, but for all $z, z' \in \mathcal{G}_z$ and all $f \in \mathcal{G}_f$. If a nonempty fault set $\mathcal{F}$ can be found such that Eq. (8.96) is satisfied for all $z, z' \in \mathcal{G}_z$, the set $\mathcal{G}_z$ is called a *stochastically*

undiagnosable state set. If the stochastic automaton does not possess such a state set, it is called diagnosable.

Definition 8.5 (Stochastic diagnosability)
A stochastic automaton is called stochastically diagnosable *if it does not possess any stochastically undiagnosable state set.*

It is obvious from the investigations above that for stochastically diagnosable systems the diagnostic algorithm yields better results than a simulation of the behaviour of the fault model.

Example 8.8 *Diagnosability of stochastic automata*

The stochastic automaton depicted in Fig. 8.18 is not stochastically undiagnosable. Nevertheless, the set of faults $\mathcal{F} = \mathcal{N}_f$ is stochastically undiagnosable within the set of states $\mathcal{G}_z = \{1, 2, 3\}$. Hence, as long as the system is not in state $z = 4$ nor has the possibility to go to this state within one time step, no information about the fault can be obtained. This result can be seen from Example 8.7. The fault $f = 1$ is proved not to exist at time $k_h = 3$ when the output $w = 3$ occurs, which proves that the automaton is in the state $\tilde{z} = (4, 2)'$. □

Example 8.9 *Diagnosis of a stochastic automata*

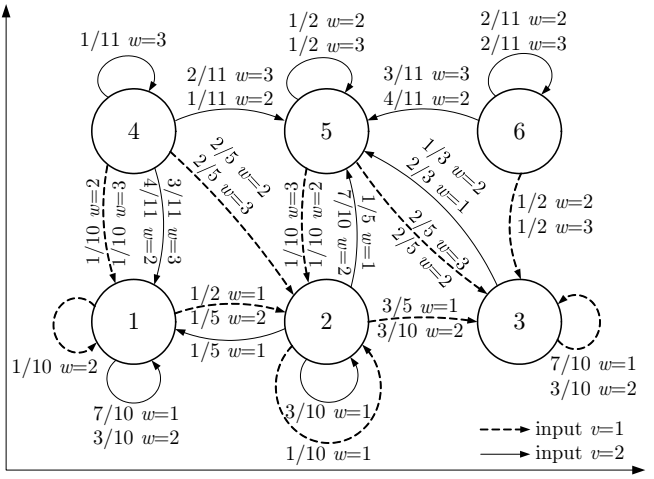


Fig. 8.19. Automaton graph for fault $f = 1$

As an example consider the task to diagnose a fault by means of the automata shown in Fig. 8.19 and 8.20. Both automata together describe the behavioural relation $L(z', w \mid z, v, f)$. The fault is assumed to be constant. Hence, the fault model (8.95) is used.

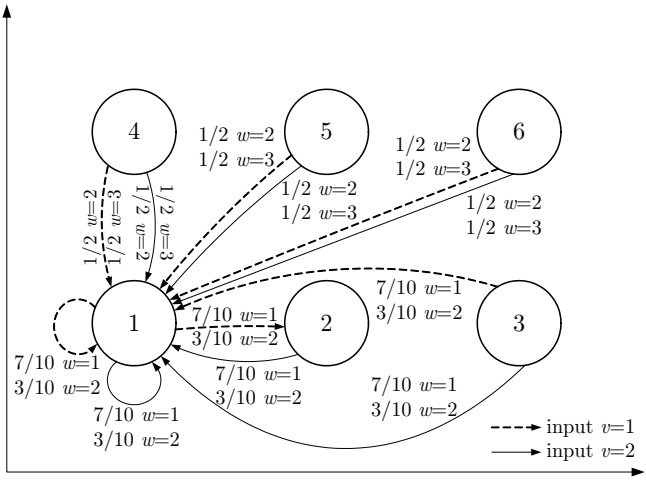


Fig. 8.20. Automaton graph for fault $f = 2$

A diagnosability check for $v = 1$ yields the result that the stochastic automaton is not diagnoseable with respect to the set of faults $\mathcal{N}_f = \{1, 2\}$ within all states $\mathcal{G}_z = \mathcal{N}_z$. For $v = 2$ the automaton is diagnosable.

Three experiments are considered. First, the input is fixed at $v = 2$ and the fault is $f = 1$. An experiment with the initial state $z = 6$ yields the output sequence shown in the left part of Fig. 8.21. A second experiment with the same initial state is made for $v = 2$ and $f = 2$ resulting in the output sequence shown in the middle part of Fig. 8.21, and a third experiment with $v = 1$ and $f = 1$ leads to the right part of Fig. 8.21.

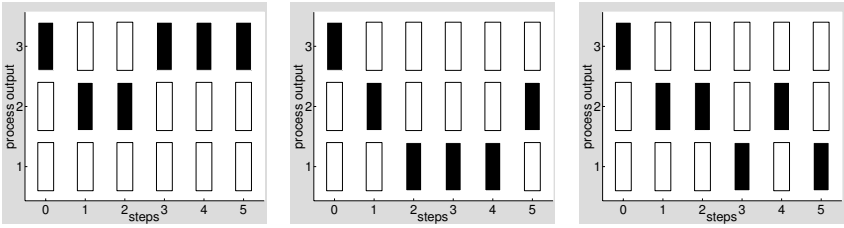


Fig. 8.21. Output sequences for $v = 2$, $f = 1$ (left), $v = 2$, $f = 2$ (middle) and $v = 1$, $f = 1$ (right)

The diagnostic results corresponding to the three experiments are shown in Fig. 8.22. It can be seen that the fault is isolated for $v = 2$, but for $v = 1$ the diagnostic result is merely a simulation of the initially known fault distribution, which results for the given automaton in a sequence of uniform distributions. The result is obtained because the automaton is undiagnosable for $v = 1$ in the whole state set $\mathcal{N}_z$. □

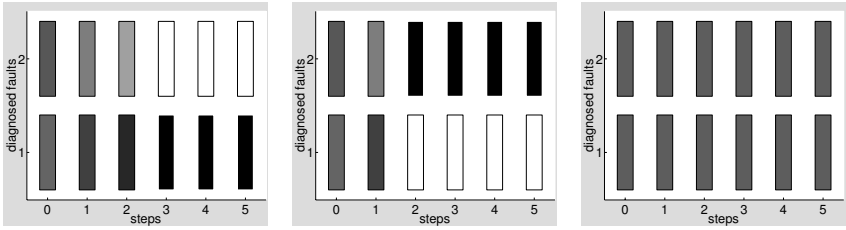


Fig. 8.22. Diagnostic results for the three experiments shown in Fig. 8.21 in the same order

8.5 Remote diagnosis of discrete-event systems

8.5.1 Diagnostic aim

This section concerns the situation in which the diagnostic task has to be accomplished by an on-board and an off-board diagnostic system where the off-board system is connected to the system to be diagnosed via a possibly unreliable information link. This situation is typical for remote diagnosis introduced in Section 1.4.3.

This section follows the idea to decompose the diagnostic task in such a way that the task of fault detection is carried out by the on-board component whereas the fault identification problem is solved off-board (Fig. 1.18). The problem considered now is how to cope with data losses brought about by the communication link. First, it has to be defined which properties the diagnostic system should have in case of lost data.

To fix the required property, it is recalled from Section 1.3.2 that any consistency-based diagnostic system aims at finding the set of fault candidates, which includes all faults f, for which the measured I/O-pair is consistent with the model $M(f)$ that describes the system behaviour subject to the fault f. If the diagnoser has full information

$$\begin{aligned} V_I(0...k) &= (v(0), v(1), ..., v(k_h)) \\ W_I(0...k) &= (w(0), w(1), ..., w(k_h)). \end{aligned}$$

the best possible set $\mathcal{F}^*$ of fault candidates can be obtained:

$$\mathcal{F}^*(k_h) = \{f : \text{The I/O-pair is consistent with the model } \ M(f)\}.$$

Under the usual assumptions that the models are precise enough and the fault set $\mathcal{N}_f$ includes the true fault, the fault f that really occurs in a given situation is known to belong to this set of fault candidates:

$$f \in \mathcal{F}^*(k_h).$$

If data are lost in the data network, instead of the full information (V_I, W_I) only a reduced I/O-pair $(\hat{V}_I, \hat{W}_I)$ is transmitted to the off-board diagnostic system (Fig. 1.18). Then, the set $\hat{\mathcal{F}}$ of fault candidates that the off-board system determines should include the true fault:

$$f \in \hat{\mathcal{F}}(k_h).$$

This claim can only be met, if the off-board diagnostic algorithm is designed so as to satisfy the relation

$$\hat{\mathcal{F}}(k_h) \supseteq \mathcal{F}^*(k_h) \qquad \text{for all } k_h. \tag{8.98}$$

This property is called the *completeness of the diagnostic result* . The aim of this section is to propose a diagnostic method for discrete-event systems described by non-deterministic automata that satisfies this requirement.

8.5.2 On-board fault detection

The on-board component has to detect faults by means of the model $M(f_0)$ of the faultless system. This is done by testing the consistency of the I/O-pair

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

with the model (where the index "D" used in Fig. 1.18 has been removed for the conciseness of notation). The on-board method is developed here for systems that are described by non-deterministic automata introduced in Section 3.6 as 5-tuple

$$\mathcal{N}(\mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, L_n, z_0)$$

with the behavioural relation

$$L_n : \mathcal{N}_z \times \mathcal{N}_w \times \mathcal{N}_z \times \mathcal{N}_v \in \{0, 1\}.$$

The function L_n indicates which state transition $z(k) \rightarrow z(k+1)$ is possible if the system gets the input $v(k)$ and produces the output $w(k)$. If this is possible, the relation

$$L_n(z(k+1), w(k), z(k), v(k)) = 1$$

holds (cf. Eq. (3.14). The diagnostic problem is considered here for a constant fault f, where f_0 denotes the faultless case.

Compared to the diagnosis of stochastic automata described in Section 8.4 the diagnostic problem is simpler because the model just distinguishes between possible and impossible state transitions but does not provide any information about the frequency with which these transitions occur. Therefore, the fault detection task of the on-board component is carried out by simply testing the consistency of the I/O-pair (V, W) with the model $M(f_0)$ of the faultless system.

The I/O-pair is consistent with this model represented by the non-deterministic automaton $\mathcal{N}$ if there exists a state sequence

$$Z(0...k_h) = (z(0), z(1), ..., z(k_h + 1))$$

such that

$$L_n(z(k+1), w(k), z(k), v(k)) = 1$$

hold for all $k = 0, 1, ..., k_h$, where $v(k)$ and $w(k)$ are the k-th element of the sequences $V(0...k_h)$ or $W(0...k_h)$, respectively. This condition is equivalent to the relation

$$\sum_{z(0)} \sum_{z(1)} \cdots \sum_{z(k_h)} \sum_{z(k_h+1)} \prod_{k=0}^{k_h} L_n(z(k+1), w(k), z(k), v(k)) > 0. \tag{8.99}$$

To get a recursive representation of the diagnostic algorithm, define the possiblity

$$\mathrm{Poss}(z \,|\, k_h + 1) \in \{0, 1\} \qquad \text{for all } \; z \in \mathcal{N}_z$$

of the system to be in the state z at time $k_h + 1$ after the I/O-pair of length k_h has been evaluated. This mapping has to satisfy the requirement

$$\mathrm{Poss}(z(k_h + 1) \,|\, k_h + 1) = 1 \Longleftrightarrow$$
$$\sum_{z(0)} \cdots \sum_{z(k_h)} \prod_{k=0}^{k_h} L_n(z(k+1), w(k), z(k), v(k)) > 0.$$

It can be represented in the following recursive way:

$$\mathrm{Poss}(z \,|\, 0) \;=\; 1 \tag{8.100}$$
$$\mathrm{Poss}(z' \,|\, k_h{+}1) \;=\; \sum_{z} L_n(z', w(k_h), z, v(k_h)) \cdot \mathrm{Poss}(z \,|\, k_h). \tag{8.101}$$

A fault is detected if

$$\prod_{z \in \mathcal{N}_z} \mathrm{Poss}(z \,|\, k_h + 1) = 0$$

holds for some k_h, which can be reformulated by introducing

$$\mathrm{Poss}(f_0 \,|\, k_h) \in \{0, 1\}$$

as an indicator of a fault. The relation $\mathrm{Poss}(f_0 \,|\, k_h) = 0$ should show that the faultless case is inconsistent with the I/O-pair and, hence, a fault has been detected. Hence, this indicator is obtained as

$$\mathrm{Poss}(f_0 \,|\, k_h) = \prod_{z \in \mathcal{N}_z} \mathrm{Poss}(z \,|\, k_h + 1). \tag{8.102}$$

Algorithm 8.4 *On-board fault detection algorithm*

Given: Non-deterministic automaton $\mathcal{N}$ describing the faultless system

Initialisation: $\mathrm{Poss}(z \mid 0) = 1 \quad \text{for all } z \in \mathcal{N}_z$
$k_h = 0$

Loop: 1. Measure the current input $v(k_h)$ and output $w(k_h)$.

2. Determine $\mathrm{Poss}(z \mid k_h + 1)$ according to Eq. (8.101) for all $z \in \mathcal{N}_z$

3. Determine $\mathrm{Poss}(f_0 \mid k_h)$ by Eq. (8.102) for all $z \in \mathcal{N}_z$

4. If $\mathrm{Poss}(f_0 \mid k_h) = 0$ holds, stop the algorithm (a fault is detected).

5. $k_h := k_h + 1$
Continue with Step 1.

Result: $\mathrm{Poss}(f_0 \mid k_h)$ for increasing time horizon k_h

8.5.3 Off-board fault identification

The off-board component uses models $M(f)$, $f \in \mathcal{N}_f$ of the faulty system in order to identify the set of fault candidates. These models are non-deterministic automata

$$\mathcal{N}(\mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, \mathcal{N}_f, L_n, z_0)$$

whose behavioural relation

$$L_n : \mathcal{N}_z \times \mathcal{N}_w \times \mathcal{N}_z \times \mathcal{N}_f \times \mathcal{N}_v \ \in \ \{0, 1\}.$$

describes the possible state transition of the system subject to some fault $f \in \mathcal{N}_f$, if

$$L_n(z(k+1), w(k), z(k), f, v(k)) = 1.$$

As the fault is assumed not to change during the time of application of the diagnostic algorithm, the fault f in this relation does not depend on the time instant k.

The off-board component has to evaluate the I/O-pair for all models $M(f)$. The I/O-pair is consistent with the model $M(f)$ and, hence, f is a fault candidate, if the following extension of Eq. (8.99) holds:

$$\sum_{z(0)} \sum_{z(1)} \cdots \sum_{z(k_h)} \sum_{z(k_h+1)} \prod_{k=0}^{k_h} L_n(z(k+1), w(k), z(k), f, v(k)) > 0. \quad (8.103)$$

For the recursive representation of the consistency test, the indicator $\mathrm{Poss}(z \,|\, k_h+1)$ has to be determined with respect to the fault f and is, thus, denoted by $\mathrm{Poss}(z, f \,|\, k_h+1)$. The relation $\mathrm{Poss}(z, f \,|\, k_h+1) = 1$ indicates that the system subject to the fault f can assume at time k_h+1 the state z. Equations (8.100), (8.101) have to be extended to

$$\mathrm{Poss}(z, f \,|\, 0) = 1 \tag{8.104}$$

$$\mathrm{Poss}(z', f \,|\, k_h+1) = \sum_{z} L_n(z', w(k_h), z, f, v(k_h)) \cdot \mathrm{Poss}(z, f \,|\, k_h). \tag{8.105}$$

A fault f is a fault candidate if

$$\mathrm{Poss}(f \,|\, k_h) = \prod_{z \in \mathcal{N}_z} \mathrm{Poss}(z, f \,|\, k_h+1) > 0 \tag{8.106}$$

holds:

$$\mathcal{F}(k_h) = \{f \,:\, \mathrm{Poss}(f \,|\, k_h) > 0\}. \tag{8.107}$$

Algorithm 8.5 *Fault identification algorithm*

Given: Non-deterministic automaton $\mathcal{N}$ describing the faulty system

Initialisation: $\mathrm{Poss}(z, f \,|\, 0) = 1$ for all $z \in \mathcal{N}_z,\ f \in \mathcal{N}_f$
$k_h = 0$

Loop: 1. Measure the current input $v(k_h)$ and output $w(k_h)$.

2. Determine $\mathrm{Poss}(z, f \,|\, k_h+1)$ according to Eq. (8.105) for all $z \in \mathcal{N}_z$ and all $f \in \mathcal{N}_f$

3. Determine $\mathrm{Poss}(f \,|\, k_h)$ by Eq. (8.106) for all $z \in \mathcal{N}_z$ and all $f \in \mathcal{N}_f$

4. Determine the set of fault candidates by Eq. (8.107).

5. $k_h := k_h+1$
Continue with Step 1.

Result: Set of fault candidates $\mathcal{F}(k_h)$ for increasing time horizon k_h

Fault identification after lost data. The main problem with the application of Algorithm 8.5 by the off-board component is the fact that this component is not guaranteed to get the whole I/O-pair. The following describes the necessary extension of this algorithm.

It is assumed that the measurement signals are sent from the on-board system towards the off-board systems as packages of the signal tuples

$$p(k) = \begin{pmatrix} v(k) \\ w(k) \\ k \end{pmatrix},$$

where k is the number of the I/O-pair. By considering the package number k, the off-board system can detect packet losses and it knows which packets were lost during the transmission.

It is now considered how to cope with the situation in which the k-th package is lost, but all other packages are received by the off-board system. Then the input and output sequences

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

have to be processed in such a way that the diagnostic result is complete according to Eq. (8.98), where $\mathcal{F}^*$ denotes the result obtained for the complete I/O-pair by Algorithm 8.5.

Denote the result of the off-board algorithm by $\hat{\text{Poss}}(z, f \mid k_h{+}1)$ and $\hat{\text{Poss}}(f \mid k_h)$ such that the set of fault candidates obtained by the off-board component is given by

$$\hat{\mathcal{F}}(k_h) = \{f : \hat{\text{Poss}}(f \mid k_h) > 0\}. \tag{8.108}$$

If consecutive measurement data are received, the recursion (8.104), (8.105) can be used without changes and only have to be rewritten for the new symbols:

$$\hat{\text{Poss}}(z, f \mid 0) = 1 \tag{8.109}$$

$$\hat{\text{Poss}}(z', f \mid k_h + 1) = \sum_z L_n(z', w(k_h), z, f, v(k_h)) \cdot \hat{\text{Poss}}(z, f \mid k_h) \tag{8.110}$$

If the k-th package is lost, $\hat{\text{Poss}}(z, f \mid k + 2)$ has to be determined directly from the recursion for the $(k-1)$-st measurement as follows:

$$\begin{aligned} \hat{\text{Poss}}(z'', f \mid k + 2) = \sum_{v(k)} \sum_{w(k)} \sum_{z'} \sum_{z} & L_n(z'', w(k{+}1), z', f, v(k{+}1)) \\ & \cdot L_n(z', w(k), z, f, v(k)) \cdot \hat{\text{Poss}}(z, f \mid k) \end{aligned} \tag{8.111}$$

$$\hat{\text{Poss}}(z, f \mid k{+}1) = \hat{\text{Poss}}(z, f \mid k{+}2), \tag{8.112}$$

where the additional summations are carried out for all $v(k) \in \mathcal{N}_v$, $w(k) \in \mathcal{N}_w$ and $z' \in \mathcal{N}_z$. Equation (8.112) has been added to define the function $\hat{\text{Poss}}(z, f \mid k_h)$ also for the time instant $k_h = k + 1$ at which the data loss occurred. Finally, the set of fault candidate is determined by means of the indicator

$$\hat{\text{Poss}}(f \mid k_h) = \prod_{z \in \mathcal{N}_z} \hat{\text{Poss}}(z, f \mid k_h + 1) \tag{8.113}$$

Theorem 8.6 (Completeness of the off-board diagnostic result)
The diagnostic result obtained by Eqs. (8.109) – (8.112) is complete, that is, the requirement (8.98) is satisfied for the set $\hat{\mathcal{F}}(k_h)$ obtained by Eq. (8.108).

Proof. The following shows that for the indicators $\hat{\text{Poss}}(f \mid k_h)$ obtained by Eqs. (8.109) – (8.112) and $\text{Poss}(f \mid k_h)$ obtained by Eqs. (8.104), (8.105) the relation

$$\hat{\text{Poss}}(f \mid k_h) \geq \text{Poss}(f \mid k_h)$$

holds which implies the relation (8.98). As the first $k-1$ packages are received by the off-board component, both indicators are the same until time $k-1$:

$$\hat{\text{Poss}}(f \mid k_h) = \text{Poss}(f \mid k_h) \quad \text{for } k_h = 0, 1, ..., k-1$$

because the recursions (8.109), (8.110) and (8.104), (8.105) coincide. After the data loss at time k, Eq. (8.111) yields $\hat{\text{Poss}}(z, f \mid k+2)$, whereas $\text{Poss}(z, f \mid k+2)$ is obtained by applying Eq. (8.105) twice:

$$\begin{aligned}
& \text{Poss}(z'', f \mid k+2) \\
= \; & \sum_{z'} L_n(z'', w(k+1), z', f, v(k+1)) \cdot \text{Poss}(z', f \mid k+1) \\
= \; & \sum_{z'} L_n(z'', w(k+1), z', f, v(k+1)) \\
& \qquad \cdot \sum_{z} L_n(z', w(k), z, f, v(k)) \cdot \text{Poss}(z, f \mid k).
\end{aligned}$$

The following reformulation yields the required result:

$$\begin{aligned}
& \text{Poss}(z'', f \mid k+2) \\
= \; & \sum_{z'} \sum_{z} L_n(z'', w(k+1), z', f, v(k+1)) \\
& \qquad \cdot L_n(z', w(k), z, f, v(k)) \cdot \text{Poss}(z, f \mid k) \\
\leq \; & \sum_{v(k)} \sum_{w(k)} \sum_{z'} \sum_{z} L_n(z'', w(k+1), z', f, v(k+1)) \\
& \qquad \cdot L_n(z', w(k), z, f, v(k)) \cdot \text{Poss}(z, f \mid k) \\
= \; & \hat{\text{Poss}}(z'', f \mid k+2). \; \square
\end{aligned}$$

With this result, the Algorithm 8.5 can be extended to an off-board diagnostic algorithm that tolerates single package losses:

Algorithm 8.6 *Off-board fault identification algorithm*

Given: Non-deterministic automaton $\mathcal{N}$ describing the faulty system

Initialisation: $\hat{\text{Poss}}(z, f \mid 0) = 1 \quad \text{for all } \; z \in \mathcal{N}_z, \; f \in \mathcal{N}_f$
$k_h = 0$

Loop: 1. Receive the next data package

Next event: 2. If the data concern the I/O-pair $(v(k_h), w(k_h))$ determine $\hat{\text{Poss}}(z, f \mid k_h + 1)$ according to Eq. (8.110) for all $f \in \mathcal{N}_f$

3. Determine $\hat{\text{Poss}}(f \mid k_h)$ by Eq. (8.113)

4. Determine the set of fault candidates by Eq. (8.108).

5. $k_h := k_h + 1$
Continue with Step 1.

Data loss: 2. If a package is lost and the received data concern the I/O-pair $(v(k_h+1), w(k_h+1))$ determine $\hat{\text{Poss}}(z, f \mid k_h+2)$ according to Eq. (8.111) and $\hat{\text{Poss}}(z, f \mid k_h + 1)$ by Eq. (8.112) for all $f \in \mathcal{N}_f$

3. Determine $\hat{\text{Poss}}(f \mid k_h)$ and $\hat{\text{Poss}}(f \mid k_h + 1)$ by Eq. (8.113)

4. Determine the sets $\hat{\mathcal{F}}(k_h)$ and $\hat{\mathcal{F}}(k_h + 1)$ by Eq. (8.108).

5. $k_h := k_h+2$
Continue with Step 1.

Result: Set of fault candidates $\hat{\mathcal{F}}(k_h)$ for increasing time horizon k_h

Extension for multiple package losses. The method described so far for single data losses shows the main idea for dealing with this situation. As no information is received by the off-board component about the input $v(k)$ and the output $w(k)$ and as the completeness of the diagnostic result should be ensured, the diagnostic method excludes faults f only if they have to be excluded for arbitrary I/O-pairs $(v(k), w(k))$. If the summation in Eq. (8.111) over $v(k)$, $w(k)$ and $z' = z(k+1)$ yields a positive result, one could replace the missing data $(v(k), w(k))$ by a pair (v, w) with $v \in \mathcal{N}_v$, $w \in \mathcal{N}_w$ such that the completed I/O-sequence is consistent with the model for the fault f. This "optimistic" replacement of the missing data is necessary in order to ensure the completeness of the diagnostic result.

This idea can obviously extended to multiple data losses, where two or more succeeding packages are not transmitted to the off-board component. Then the summation in Eq. (8.111) has to be extended over all these lost data together with the successor states of these data. Clearly, the larger the number of lost packages is, the more conservative is the diagnostic result. However, the method explained ensures the completeness of the set of fault candidates. That is, no fault f is excluded from this set, for which the received data do not prove its invalidity.

8.6 Sensor and actuator diagnosis

8.6.1 Diagnostic problem

This section presents two observer schemes for isolating sensor or actuator failures. It is based on the diagnostic method developed in Section 8.4, which is now applied to the situation that a sensor or an actuator fails.

In the previous sections scalars have been used for the symbolic input and output. However, control systems usually have multiple input and output signals, associating physically each input to an actuator and each output to a sensor. Therefore, the scalar input and output symbols used before are, in fact, composed of several symbolic values given to the actuators or obtained from the sensors, which are encoded by the scalars v and w. The mappings to be applied to the input or output vectors are formally introduced in the following. The vector notation makes it possible for the diagnostic algorithm to associate with a fault the right sensor or actuator. Hence, the methods that will be developed in this section *isolate* the fault by showing which component (sensor or actuator) fails.

For a multi-input multi-output system, instead of the whole input set

$$\mathcal{N}_v = \{1, 2, \ldots, M_1 \cdot M_2 \cdots M_m\}$$

m sets with the symbolic input values of the m input channels are used

$$\mathcal{N}_{v_i} = \{1, 2, \ldots, M_i\}\,, \qquad i = 1, \ldots, m$$

and a mapping

$$M_{\boldsymbol{v}} : \mathcal{N}_{v,1} \times \cdots \times \mathcal{N}_{v,m} \longrightarrow \mathcal{N}_v$$

is defined such that the input value v used in the preceeding sections is obtained from the symbolic values v_i of the m input channels:

$$v = M_{\boldsymbol{v}}(v_1, \ldots, v_m).$$

Analogously, the sets

$$\begin{aligned} \mathcal{N}_w &= \{1, 2, \ldots, R_1 \cdot R_2 \cdots R_r\} \\ \mathcal{N}_{w_i} &= \{1, 2, \ldots, R_i\}\,, i = 1, \ldots, r \end{aligned}$$

and the mapping

$$\begin{aligned} M_{\boldsymbol{w}} &: \mathcal{N}_{w,1} \times \cdots \times \mathcal{N}_{w,r} \longrightarrow \mathcal{N}_w \\ w &= M_{\boldsymbol{w}}(w_1, \ldots, w_r) \end{aligned}$$

are defined. In the following, the input and output of the system are assumed to be vector-valued and the bijective mappings $M_{\boldsymbol{v}}$ and $M_{\boldsymbol{w}}$ transform these input and output vectors to scalar symbolic values. With these bijective mappings in mind, a vector notation with the input vector $\boldsymbol{v} \in N_{\boldsymbol{v}}$ and the output vector $\boldsymbol{w} \in N_{\boldsymbol{w}}$ with

$$N_{\boldsymbol{v}} = \mathcal{N}_{v,1} \times \cdots \times \mathcal{N}_{v,m}$$

and

$$N_{\boldsymbol{w}} = \mathcal{N}_{w,1} \times \cdots \times \mathcal{N}_{w,r}$$

is used.

The aim of sensor or actuator fault diagnosis is to determine which symbolic input v_i or output w_i causes an inconsistency with the faultless model. The motivation is to identify faulty measurements of the input or output without the need to explicitly model such faults. The assumption simply is that a faulty sensor or faulty actuator behaves differently as a faultless one. Assume that $\tilde{v}_i(k)$ or $\tilde{w}_i(k)$ denote the true input to the system or output from the system. Then for a faultless actuator the relation $v_i(k) = \tilde{v}_i(k)$ and for a faultless sensor the equality $w_i(k) = \tilde{w}_i(k)$ holds true. The occurrence of a fault in the i-component implies that the given equality does not hold for the index i.

The main idea of sensor or actuator diagnosis is explained for corrupted output signals. Assume that output symbols w_i are measured that differ from the symbols $\tilde{w}_i$ that really occur in the system. According to Section 8.3.6, the denominator

$$D(k_h) = \sum_{z(k_h+1), z(k_h)} L(k_h) \cdot \mathrm{Prob}(z(k_h) \mid V(0...k_h-1), W(0...k_h-1)) \quad (8.114)$$

of Eq. (8.64) is nonzero if and only if the I/O pair (V, W) up to time k_h is consistent with the system. Hence, as long as for all output signals w_i, $(i = 1, ..., r)$ the relation $w_i(\kappa) = \tilde{w}_i(\kappa)$, $(\kappa = 0, ..., k_h)$ holds, a nonzero denominator $D(k_h) > 0$ is obtained. If the denominator $D(k_h)$ becomes zero, at least one of the output signals $w_i(\kappa)$ must differ from the correct output $\tilde{w}_i(\kappa)$ for at least one time instant κ. If this inconsistency is removed by repeating the diagnosis for $\kappa = 0...k_h$ while using all but the i-th output, it becomes clear that the i-th output must be corrupted. For this repetition, a model has to be used, which describes the system with respect to all but the i-th output. Resolving the inconsistency in this way isolates the source of an output signal corruption.

8.6.2 Sensor supervision

This section develops two schemes for sensor failure detection. As this scheme is conceptually similar to the generalised observer scheme or the dedicated observer scheme developed for continuous-variable systems [65], the same terminology is

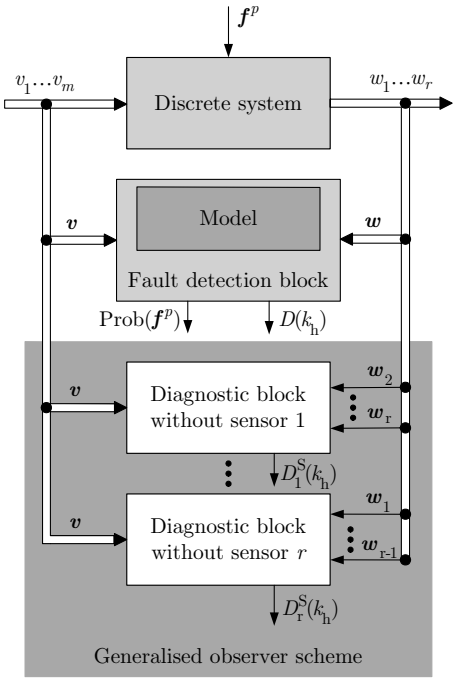


Fig. 8.23. Generalised observer scheme for sensor fault isolation

used here. The sensor or actuator faults which are considered in this section results in a complete failure of these components. Therefore, the term "sensor failure detection" describes the situation better than "sensor fault isolation" and will be used here.

Generalised observer scheme. The idea explained above leads to the *generalised observer scheme* depicted in Fig. 8.23. The upper block is the state observer of the overall system. By means of its output $D(k_h)$ obtained from Eq. (8.114), the consistency of the I/O pair is checked. If $D(k_h)$ becomes zero, the other r observers are invoked to test whether the corresponding output signal is corrupted. Each of the r blocks of the generalised observer scheme obtains all but one output signals and tests the consistency of the input sequence and the sequence of the remaining output signals with a model that describes the system with all but the i-th output. Each block of the generalised observer scheme yields the denominator of the corresponding observer as output. Furthermore, each block yields an observation result based on less information compared to the main observation block.

The lower part of Fig. 8.23 has the same structure as the generalised observer scheme known from sensor failure detection of continuous-variable systems. However, as discrete-event systems are described by stochastic automata rather than dif-

ferential equations, the methods used to apply the generalised observer scheme for continuous and discrete systems differ completely.

The following results formally state the above explanation of the generalised observer scheme. The first result concerns the model to be used by the blocks of the generalised observer scheme (for a proof cf. [159]).

Lemma 8.4 *The stochastic process without the i-th output w_i is represented by the stochastic automaton*

$$\mathcal{S} = (\mathcal{N}_z, N_{\boldsymbol{v}}, \mathcal{N}_{w,1} \times \cdots \mathcal{N}_{w,i-1} \times \mathcal{N}_{w,i+1} \times \cdots \times \mathcal{N}_{w,r}, L_{\bar{w}_i}) \ , \quad (8.115)$$

with the behavioural relation

$$L_{\bar{w}_i}(z', (w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_r) \mid z, \boldsymbol{v}) = \sum_{w_i} L(z', \boldsymbol{w} \mid z, \boldsymbol{v}) \ . \quad (8.116)$$

The lemma says that the stochastic automaton representing the stochastic process without output w_i can be obtained directly from the stochastic automaton representing the whole stochastic process, if the new behavioural relation $L_{\bar{w}_i}$ is determined according to Eq. (8.116). The i^{th} component of the generalised observer scheme determines the denominator (8.114) by means of the automaton (8.115). Its result is denoted by $D_i^S(k_h)$.

Lemma 8.5 *$D(k_h) > 0$ implies $D_i^S(k_h) > 0$ $(i = 1, ..., r)$ for each block of the generalised observer scheme.*

According to this result, the observation blocks of the generalised observer scheme need only to be invoked if $D(k_h)$ becomes zero. Otherwise it is clear that these blocks, which are connected to a fewer number of output signals, indicate the consistency of the I/O pair with their model.

Provided that only one output signal is corrupted, it can be isolated due to the following property of the generalised observer scheme:

Lemma 8.6 *Assume that the i-th output signal contains values $w_i(\kappa)$ different from $\tilde{w}_i(\kappa)$ while the remaining output signals and the input are correct for $\kappa = 0, 1, ..., k_h$. Then $D_i^S(k_h) > 0$ holds.*

Accordingly, the i-th block does not become inconsistent if only the i-th output signal is corrupted. Hence, the generalised observer scheme can be used to isolate the corrupted output signal as follows:

Corollary 8.6 (Identification of corrupted output signals by the Generalised Observer Scheme)
Assume that the i-th output signal contains values $w_i(\kappa)$ different from $\tilde{w}_i(\kappa)$ while the remaining output signals and the input are correct for $\kappa = 0, 1, ..., k_h$. If the observer block yields the result $D(k_h) = 0$ and the generalised observer scheme the results

$$D_j^S(k_h) = 0, \quad j = 1, ..., i-1, i+1, ..., r$$
$$D_i^S(k_h) > 0$$

then the i-th output signal is corrupted.

Example 8.10 *Sensor failure detection of the two-tank system*

The generalised observer scheme is applied to the two-tank example introduced in Section 2.1. The sensors are assumed to deliver discrete sensor readings, which show that the levels of the corresponding tank is in a certain interval. For both sensors, 10 different levels are distinguished. If the pump is assumed to obtain a discrete input, which shuts the pump off or on, the two-tank system is a discrete-event system. Besides the sensor failures, there may be a leakage in Tank 1 and a blockage of the outflow.

As it will be shown in detail in Chapter 9, the two-tank system can be modelled by a stochastic automaton, which has direct reference to the discrete-valued input and the discrete sensor information. A stochastic automaton is obtained from the state-space model of the tank system given in Section 2.1 by means of the abstraction method explained in Section 9.4.2. An automaton with 100 states and 2027 nonzero entries in the behavioural relation L has been obtained, which cannot be shown here. This model takes into consideration only the plant faults (leakage, blockage), but does not refer to the sensor failures. The observer scheme that uses this model has been applied to a laboratory tank system with the following results.

Figure 8.24 shows a sequence of measurements obtained from the two level sensors. The black regions mark the intervals in which the tank levels reside at the corresponding time instant. Only these intervals but no precise measurement values are known. In the experiment, the pump has been commanded to be off except for the time interval between 90 s and 170 s. Before time zero, the diagnostic block gave probability of 1 for the faultless case.

At time 40 s, the level sensor 1 breaks down and gives permanently a qualitative level zero as shown in Fig. 8.24. However, up to time 100 s the fault detection block signals consistency of the measured I/O pair with the model and hence the failure of sensor 1 cannot be detected in this time interval (cf. Fig. 8.25). This is because a low level in Tank 1 for pump off is not in conflict with the system dynamics. A more detailed analysis by applying the diagnosability test explained in Section 8.4.4 shows that the sensor fault is not diagnoseable in the time interval between 40 s and 100 s.

At time 100 s the fault detection block indicates the existence of some fault ($D(k_h) = 0$ for $k_h \geq 100$ s) and the generalised observer scheme is invoked. Intuitively this inconsistency occurs because the level measurement in Tank 1 remained zero when the pump is on. According to Fig. 8.25 two observer blocks remain consistent at 100 s, namely the blocks for a pump fault and for the Sensor 1 fault. Obviously, a pump fault also explains the zero measurement at time 100 s and the result is, therefore, correct. But when at time 160 s the level in Tank 2 reaches the second quantisation interval it is visible from the measurements that Pump 1 is working and, hence, Sensor 1 must be faulty. Exactly this result is obtained,

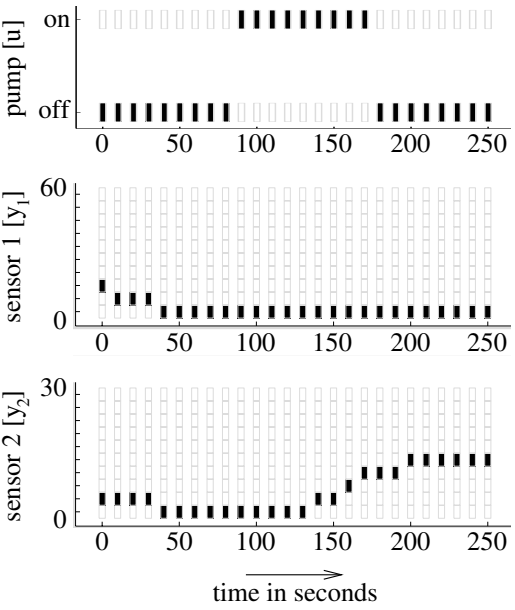


Fig. 8.24. Discrete input sequence (top) and discrete measurement sequences of the two sensors

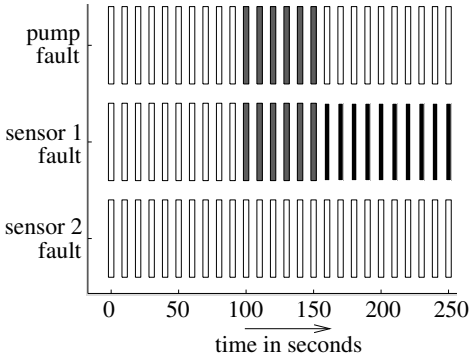


Fig. 8.25. Result of the sensor and actuator supervision system. Grey or black boxes indicate consistency of the I/O pair with the corresponding model at the corresponding time instant.

and only the supervision block not connected to Sensor 1 remains consistent and isolates the faulty sensor. □

Dedicated observer scheme. An alternative way to isolate corrupted I/O signals is provided by the *dedicated observer scheme* shown in Fig. 8.26. In contrast to the generalised observer scheme, each block of the dedicated observer scheme is connected to only one output. The model used for each observer of the dedicated observer scheme is analogous to the model described in Lemma 8.4 if the sum in Eq. (8.116) is determined for all but one output. Lemma 8.5 holds equivalently for the dedicated observer scheme and the corrupted output signal can be isolated from the following result:

Lemma 8.7 *Assume that all inputs are correct. If the observer block yields the result* $D(k_h) = 0$ *and the dedicated observer scheme the result* $D_i^S(k_h) = 0$*, then the* i*-th output signal is corrupted.*

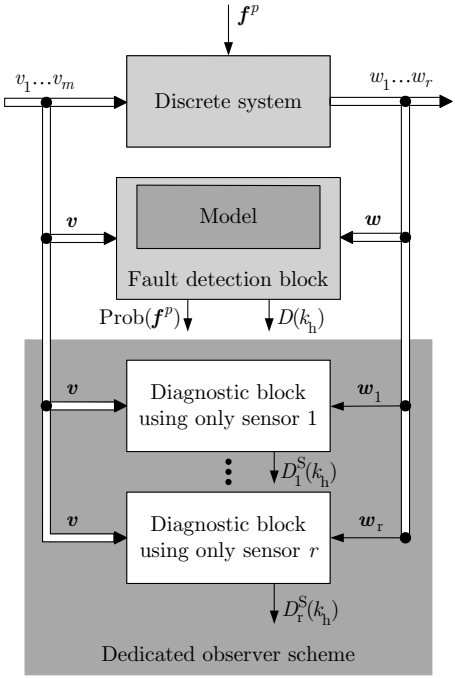


Fig. 8.26. Dedicated observer scheme for sensor fault isolation

The main advantage of the dedicated observer scheme is that multiple corrupted output signals can be detected while the generalised observer scheme, in general,

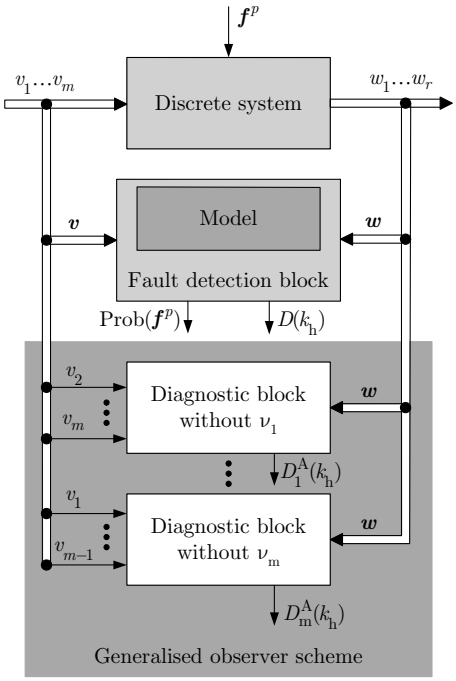


Fig. 8.27. Generalised observer scheme applied to input signal supervision

yields $D_i^S(k_h) = 0$ in this case. On the other hand, the blocks of the generalised observer scheme use more information, and are, therefore, more sensitive to corrupted signals. Furthermore, the generalised observer scheme directly yields the observation result excluding the faulty signal.

8.6.3 Actuator supervision

This section develops a "dual" concept for actuator failure detection. In Fig. 8.27, the generalised observer scheme for supervising the input signals is shown. Its application is as described by Lemmas 8.5 and 8.6. The models to be used in the blocks of the generalised observer scheme are obtained as follows:

Lemma 8.8 *The stochastic process without the i-th input v_i is represented by the stochastic automaton*

$$\mathcal{S} = (\mathcal{N}_z, \mathcal{N}_{v,1} \times \cdots \mathcal{N}_{v,i-1} \times \mathcal{N}_{v,i+1} \times \cdots \times \mathcal{N}_{v,m}, N_{\boldsymbol{w}}, L_{\bar{v_i}}) \ ,$$

with the behavioural relation

$$L_{\bar{v}_i}(z', \boldsymbol{w} \mid z, (v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_m)) = \sum_{v_i} L(z', \boldsymbol{w} \mid z, \boldsymbol{v}) \cdot \mathrm{Prob}(v_i) \ , \tag{8.117}$$

where $\mathrm{Prob}(v_i)$ *denotes the a-priori probability distribution of the* i*-th input signal.*

The generalised observer scheme for actuator failure detection is shown in Fig. 8.27. It is the "dual" structure of Fig. 8.23. The behavioural relations $L_{\bar{v}_i}$ used in the i-th diagnostic block are obtained by Eq. (8.117).

This scheme has the same properties as the generalised observer scheme for sensor failure detection. As long as fault detection block yields a non-vanishing indicator $D(k_h)$, the actuator failure detection blocks yield the same result $(D_i^A(k_h) = 0)$ and, hence, need not to be applied. If the fault detection block indicates the occurence of some fault $(D(k_h) = 0)$, the m diagnostic blocks of the generalised observer scheme are involved to decide, whether the fault is an actuator failure.

In analogy to the dedicated observer scheme for sensor failure detection shown in Fig. 8.26, a dedicated observer scheme for actuator failure detection can be developed.

Remark 8.3 *Explicit sensor or actuator fault models*
The generalised observer scheme and dedicated observer scheme provide an elegant method to isolate sensor and actuator faults without explicitly modelling these faults. The model L of the faultless system implicitly contains the assumption that the sensor signals w_i or actuator signals v_i are at all times identical to those signals actually occurring at the system. Nevertheless, sensor or actuator faults can still be explicitly modelled when using these schemes. Then the behavioural relation $L(z', w \mid z, f = f_i, v)$ for $f = f_i$ describes the system in presence of the sensor fault f_i. In this case, the denominator $D(k_h)$ of the main diagnostic block does not become inconsistent when the sensor fault f_i occurs. □

8.7 Control reconfiguration for stochastic automata

This section concerns fault-tolerant control of discrete-event systems that are described by stochastic automata. It combines the diagnosis of sensor or actuator faults with the reconfiguration of the supervisor that has access to the sensor and actuator signals. It is shown how faulty sensor signals can be replaced by reconstructed sensor output values and how the process supervisor can be reconfigured if faults occur. Due to the state of the art of fault-tolerant control of discrete-event systems, this section includes only a few preliminary investigations.

8.7.1 Automatic substitution of faulty sensors

Using the observer schemes presented in the previous section, it is possible to isolate faulty sensors. In this case, the control system can be reconfigured by replacing the missing sensor information by a computed estimate of the corrupted variable. This will be explained for the generalised observer scheme introduced in the previous section.

Assume that the i-th sensor signal is corrupted and that this sensor failure has been detected as described in Corollary 8.6. Then only the i-th state observer of the generalised observer scheme yields an observation result which is described by

$$\begin{aligned}&\mathrm{Prob}(z(k) \mid \boldsymbol{V}(0...k), \tilde{\boldsymbol{W}}(0...k)) \qquad (8.118)\\ &= \mathrm{Prob}(z(k) \mid \boldsymbol{V}(0...k), W_1(0...k), ..., W_{i-1}(0...k), W_{i+1}(0...k), ..., W_r(0...k)),\end{aligned}$$

while all other observers yield no result because of their vanishing denominators ($D_i^S(k_h) = 0$). Based on the observation result (8.118) the estimate for w_i is obtained from

$$\begin{aligned}&\mathrm{Prob}(w_i(k) \mid \boldsymbol{V}(0...k), \tilde{\boldsymbol{W}}(0...k)) \qquad (8.119)\\ &= \sum_{z'} L_{\bar{w}_i}(z', \tilde{\boldsymbol{w}}(k) \mid z(k), \boldsymbol{v}(k)) \cdot \mathrm{Prob}(z(k) \mid \boldsymbol{V}(0...k), \tilde{\boldsymbol{W}}(0...k)) \ .\end{aligned}$$

A supervisor that needs the i-th sensor reading w_i can use the most probable value:

$$\hat{w}_i(k) = \arg \max_{w_i} \mathrm{Prob}(w_i(k) \mid \boldsymbol{V}(0...k), \boldsymbol{W}(0...k)).$$

This way of substituting the faulty sensor reading by the observed value is analogous to what is done for continuous systems. The difference lies in the fact that the discrete observer used here yields a set of output values w_i with positive probability whereas a continuous observer leads to a unique approximate output value.

8.7.2 Automatic reconfiguration of diagnosis

If a sensor or actuator failure occurs, the diagnostic scheme can be adapted to this situation. This adaptation is similar to the derivation of the blocks of the generalised observer scheme from the state observer of the overall system.

Assume that the i-th output is faulty. Then the diagnostic algorithm can remain in operation after it has been reconfigured so as to ignore w_i. Then the consistency of the measured input and output sequences described by the denominator occuring in Eq. (8.91) for $\sum_{f'} G_f(f' \mid f) = 1$ is restored if L is replaced by $L_{\bar{w}_i}$:

$$\begin{aligned}&D_i^S(k_h) = \\ &\sum_{z(k_h+1), z(k_h), f(k_h)} L_{\bar{w}_i}(k_h) \cdot \mathrm{Prob}(z(k_h), f(k_h) \mid V(0...k_h-1), W(0...k_h-1)) \ .\end{aligned}$$

$L_{\bar{w}_i}(k_h)$ denotes the behavioural relation of the system without the i-th output:

$$L_{\bar{w}_i}(z', (w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_r) \mid z, , f, \boldsymbol{v}) = \sum_{w_i} L(z', \boldsymbol{w} \mid z, f, \boldsymbol{v}) \ .$$

If this denominator does not vanish the measured signals with ignored output w_i are consistent with the model. If it vanishes, a further fault is detected. Analogously, faulty inputs can be dealt with.

This reconfiguration of the diagnostic algorithm will be illustrated for quantised systems in Section 9.6.4.

8.8 Exercises

Exercise 8.1 *Observability of stochastic automata*
Assume that the current state probability distribution is given for time k_h and denoted by $\mathrm{Prob}(z(k_h))$. Prove the following fact: If at time k_h+1 an input $v(k_h+1)$ and an output $w(k_h+1)$ occur for which a decomposition (8.78) is possible for the state set

$$\mathcal{Z}(k_h \mid k_h) = \{z : \mathrm{Prob}(z_p(k_h)=z) > 0\}$$

then in Steps 4 and 5 of the observation algorithm the same results are obtained as by simulating the automaton behaviour according to Eq. (8.27). □

Exercise 8.2 *Diagnosis of fixed faults*
How can the Algorithm 8.3 be simplified if the fault is known not to change over time? □

Exercise 8.3 *Diagnosis of a batch reactor*
The reactor shown in Fig. 8.28 is used within a larger batch process, where it is filled and emptied in order to bring a certain amount of liquid into another reactor. For the behaviour of the reactor only the empty and the full state is distinguished, where the liquid level is above the higher or below the lower border shown in the figure. These states are denoted by z_1 and z_2.

To fill the reactor, the pump is switched on (input v_1), to empty the reactor, the input v_2 opens the valve. A security check ensures that the pump is not switched on if the valve is open.

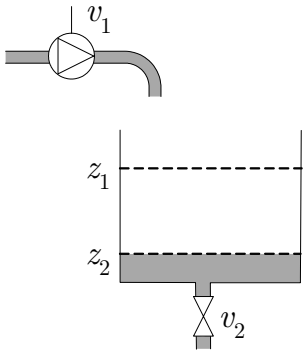


Fig. 8.28. Batch reactor

1. Describe the reactor by a deterministic automaton
2. The fault f_1 breaks the pump. Extend you model in order to describe the reactor for the faultless and the faulty operation mode.
3. The faultless reactor remains faultless with the probability of 99% in all state transitions that are caused by switching the pump or opening and closing the valve. Extend you model to get a stochastic automaton, which reflects this information. □

Exercise 8.4 *Diagnosis of non-deterministic automata*
How can the diagnostic method developed in this chapter be simplified if instead of a stochastic automaton a non-deterministic automaton is used to describe the system under consideration? Do the sets of fault candidates obtained by both methods distinguish? □

8.9 Bibliographical notes

The first results concerning state observation of discrete-event systems occurred in connection with the supervisory control theory developed in [205] where the supervisor has to reconstruct the current state of the system from partially measurable states or events. Reference [129] defined the notion of the observable language and developed results on the existence of the combined supervisory control and the observation problem given. Reference [35] showed that for supervisory control the problem of state observation can be reformulated as an event observation problem.

The classical observability definition has been given in [36] and was used also, for example, in the textbooks in [33] and [223]. According to this definition a stochastic automaton is called *semi-deterministic* or observable if for all states z the successor state $z' = \varphi(z, v, w)$ can be unambiguously determined if the current state z, the current input v and current output w are known. This definition is useful only if it can be *assumed* that one automaton state z at same time k is precisely known. Then, the future sequence of states starting in z can be unambiguously determined. However, as long as this assumption is not satisfied, the notion of observability does not say anything about the solvability of the observation problem. Similar remarks hold true for other observability definitions like the one given in [191] which likewise claim that the automaton state should be unambiguously determined.

The diagnosis of discrete-event systems was the subject of only a few papers in the past. In the references [6], [200] and [222] the diagnostic problem was set up for Petri nets whereas in [128], [157] and [210] non-deterministic or stochastic automata have been considered. Conditions on the automaton under which the fault can be uniquely determined have been derived in [128] and [210]. Reference [200] outlined the connections between discrete-event models and logical descriptions, which opens the way to apply diagnostic methods elaborated in the field of artificial intelligence to discrete-event systems (for survey cf. [86] or [139]). However, most of the diagnostic methods developed in artificial intelligence can only be used for static system descriptions, whereas the methods that have been developed here allow to diagnose dynamical systems far from their equilibrium state.

Based on Algorithm 8.3 for the diagnosis of stochastic automata, a specific sensor and actuator supervision system has been developed in [159].

The issue of complexity reduction of the diagnosis of automata has been considered in [145]. The basic idea was to lump unobservable states of the model together because during the movement in such sets of states the observation or diagnostic algorithm does not gain any additional information about the system. It has been shown that this method of complexity reduction can be applied for non-deterministic automata. However, for stochastic automata the complexity reduction brings about biased diagnostic results.

The results reported in Section 8.7 have been developed in [157]. All proofs of the results given here can be found in this reference.

The extension of the diagnostic method to remote diagnosis, where data loss in the network has to be tolerated, is described in [68] and [69].

Exercises 8.3 are taken from [144].

Chapter 9

Diagnosis and reconfiguration of quantised systems

Quantised systems are continuous-variable systems whose sensor and actuator signals can only be accessed through quantisers that produce symbolic state or event sequences. Hence, quantised systems have a discrete-event behaviour. This chapter shows how quantised systems can be represented by stochastic automata and how state observation, diagnostic and control problems can be solved. First a stochastic automaton is set up so as to represent the discrete-event behaviour of the quantised system completely. Second the given analysis and design problems are solved for the automaton by means of the methods that have been developed in Chapter 8.

9.1 Introduction to quantised systems

9.1.1 Supervision of hybrid systems

The preceding chapters have considered either continuous-variable systems, which have real-valued signals and can be described by differential or difference equations, or discrete-event systems, which have signals with symbolic values and can be described by automata, Petri nets or similar models. This chapter is devoted to an important class of systems, in which both continuous and discrete phenomena have to be taken into account. Such systems are called *hybrid systems*.

The mixture of discrete and continuous signals and discrete and continuous forms of the models used is typical for supervisory control tasks and plays a particular role in diagnosis and fault-tolerant control. Nevertheless, hybrid systems have attracted substantial interest only during the last ten years and only preliminary results are available for their supervision and control.

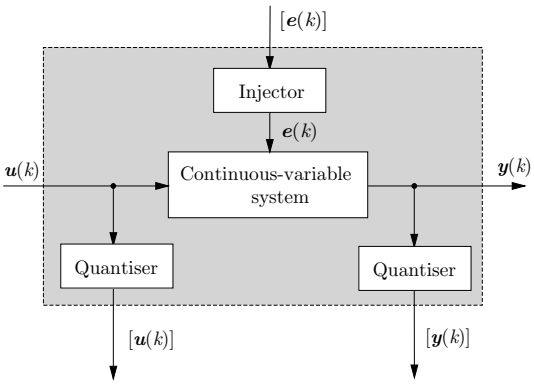


Fig. 9.1. Quantised system

Structure of quantised systems. The main problems in dealing with hybrid systems result from the different ranges of the signals. These problems are investigated in this chapter for quantised systems depicted in Fig. 9.1. The relation between the different signal ranges are represented by quantisers and injectors. The *quantiser* transforms a real-valued signal into a sequence of symbols, where the real-valued signal or signal vector is denoted by a lower-case letter like $\boldsymbol{u}$ or y and the corresponding quantised signals by $[\boldsymbol{u}]$ or $[y]$, respectively. If, in the simplest case, the quantiser decides to which real interval of a given set of intervals the current value $y(t)$ belongs, the value of the quantised signal $[y(t)]$ at the time instant t is the number of the corresponding interval. This interval can be associated with symbolic names like "normal", "high" or "low", which give a semantic signal value. As long as the signal does not leave a given interval, the quantised value remains the same. Hence, a continuous change of $y(t)$ is transformed into a sequence of discrete changes of $[y(t)]$, which shows that the quantiser can be used as an interface between real-valued and symbolic signals.

The *injector* carries out the inverse mapping. Its input is a symbolic signal like $[\boldsymbol{e}]$, which is associated with a real-valued signal $\boldsymbol{e}$. An example is given in Fig. 9.1, where the injector associates to a symbolic fault $[\boldsymbol{e}]$ the real-valued fault input $\boldsymbol{e}$.

The relation between $[\boldsymbol{e}]$ and $\boldsymbol{e}$ can be either deterministic where every symbolic value is associated with a unique real value or non-deterministic where the associated real value is randomly selected from a given set of signal values or may vary within this set as long as the symbolic value does not change. In any case, the injector is the interface from symbolic to real-valued signals.

Reasons for introducing quantisers and injectors. The question why quantisers or injectors occur in the system has many answers:

- **Measurement uncertainties:** Many physical quantities cannot be precisely measured as, for example, the biomass concentration in bioreactors, substance concentrations in the liquid or the gaseous phase, the temperature in cement kilns or

blast furnaces. Then, quantisers are introduced as a representation of systematic measurement errors.

- **Alarms:** The abnormal behaviour of industrial plants is signalled by means of alarm messages, which represent quantised signal values.

- **Discrete actuators:** Many industrial actuators can only be switched among a set of discrete values rather than be varied continuously. For example, gas burners are used in an on/off mode. This fact necessitates the introduction of an injector that transforms discrete values $[u]$ into the associated real input values u.

- **Discrete control:** Many industrial processes are controlled by programmable logic controllers, which react on quantised measurements by prescribing discrete input values. For example, the controller of an elevator does not know the lift position precisely, but has only the information between or at which floor the lift currently is.

- **Switching system dynamics:** The system dynamics switches if the input or state exceeds certain bounds. An example is given by the tank system described in Section 10.1 where the dynamical properties depend on whether the liquid levels are above or below the height of the connecting pipes. According to this quantisation of the levels h_1 and h_2 the equations given on page 506 are valid in one of the four possible configurations of existing or nonexisting flows through the upper valves. Here, signal quantisation occur internally in the system. If brought into the hybrid system structure depicted in Fig. 3.6, the discrete-event part of the tank model switches the continuous model among the four different equations.

These arguments show that quantised systems occur naturally in the engineering practice. However, in addition to the situations described above, injectors or quantisers may be deliberately introduced for the following reasons:

- **Uniformity of the system description:** The mixture of differential equations for the continuous-variable part and automata for the discrete-event part makes the model of hybrid systems very complex. Therefore, it is reasonable to deal with all signals uniformly as discrete-valued signals by introducing additional injectors and quantisers. The considerable simplification of observation and diagnostic problem due to this uniformity of the signals will become obvious in Sections 9.5 and 9.6.

- **Information reduction:** If the control aim concerns a global assessment of the system behaviour, it is reasonable to use models that have direct reference to these assessments. In the diagnostic problem considered in Section 9.6 the faults occurring in the system changes the behaviour qualitatively. Therefore, quantised information about the system behaviour is sufficient to identify the fault.

- **Reference to heuristic models:** The experience of human operators refers to subsets of the signal space rather than to specific signal values. Hence, this knowledge considers continuous-variable systems as systems with quantised signal spaces.

Quantised systems in fault-tolerant control. For systems subject to faults an additional motivation comes from the fact that faults are, in general, quantised phenomena. Obvious examples of faults concern broken wires, a leakage in a pipe, or a valve that is stuck open or closed. However, even if the fault concerns a change of some parameter, the controlled system will show an abnormal behaviour only if the parameter change is large enough and cannot be compensated by the control loops installed. Hence, also in this case a quantisation of the parameter changes into faulty and non-faulty ones is reasonable.

In Fig. 9.1 the fault is, therefore, described by its qualitative value $[\boldsymbol{e}]$, which is transformed into the actual real-valued fault parameter or fault signal by an injector. Diagnosis has only to find the qualitative value $[\boldsymbol{e}]$ rather than the real value $\boldsymbol{e}$. Compared with the diagnosis of discrete-event systems investigated in Chapter 8 the qualitative value $[\boldsymbol{e}(k_h)]$ of the fault corresponds to a symbolic fault $f(k_h)$ that occurs at the given time k_h. Therefore, in the following, the qualitative fault $[\boldsymbol{e}]$ will alternatively be denoted by the fault symbol f.

In summary, in many practical situations the relevant information used in supervisory control is included in the quantised signal. The introduction of the injectors and the quantisers aims at reducing the information and, in this way, at simplifying the control task. It follows the guideline:

> Many process supervision tasks can be solved with reasonable effort only if as much information about the system as possible is ignored.

If a more global information about the system is sufficient to solve a given task, then this global information should be used rather than the more detailed one. In the quantised system approach the resolution of the injector and the quantiser can be used to adapt the "granularity" of the information used by the supervisor to the task to be solved.

9.1.2 The quantised system approach to supervisory control

A quantised system represents a dynamical system with real-valued signals that can only be measured through quantisers (Fig 9.1). Instead of the real-valued input $\boldsymbol{u}(k)$ and output $\boldsymbol{y}(k)$ only the quantised signal values $[\boldsymbol{u}(k)]$ and $[\boldsymbol{y}(k)]$ are available. All tasks that will be considered in this chapter should be solved by using the quantised signals only.

With respect to the general hybrid system shown in Fig. 3.6, the discrete-event subsystem is missing here. This simplification is made in order to emphasise the

main problems that occur in the situation that a continuous-variable system has to be supervised by using symbolic information only. It will be shown that the quantised system can be described by a discrete-event model, which includes the continuous-variable subsystem together with the quantisers and injector. Therefore, the extension to hybrid systems can obviously be made by combining this discrete-event model of the quantised system with the model of the discrete subsystem.

The way how process supervision tasks can be solved for quantised systems will be explained in this chapter by considering two important problems: the observation of the quantised state of the system and the diagnosis of faults. The following presents the problems to be solved together with a brief outline of the way of solution, which will be explained in Sections 9.5 and 9.6. The quantised systems considered here are discrete-time systems where the time k refers to the k-th sampling time.

State observation. State observation concerns the problem of determining the internal state of a dynamical system from the input and output measurements. For continuous-variable systems the main idea is to use a LUENBERGER observer that determines an approximation $\hat{\boldsymbol{x}}$ of the continuous state $\boldsymbol{x}$. However, the application of these results is possible only if the systems input and output are measured quantitatively and if the model has the form of differential or difference equations.

For the quantised system, only symbolic input and output information is available, but a similar state observation problem can be posed. The measurement information yield the sequence of quantised input values

$$[\boldsymbol{U}(0 \,...\, k_h)] = ([\boldsymbol{u}(0)],\, ...,\, [\boldsymbol{u}(k_h)])$$

and the sequence of quantised output values

$$[\boldsymbol{Y}(0 \,...\, k_h)] = ([\boldsymbol{y}(0)],\, ...,\, [\boldsymbol{y}(k_h)]).$$

Due to the more abstract measurement information the task is to reconstruct the qualitative state $[\boldsymbol{x}]$ rather than the real-valued state $\boldsymbol{x}$. The observation problem can be stated as follows:

Problem 9.1 (Observation problem for quantised systems)

Given: *Sequence of quantised input values.*
Sequence of quantised output values.
Model $\mathcal{M}$ of the quantised system.

Find: *Current qualitative state* $[\boldsymbol{x}(k_h)]$.

The algorithm presented in Section 9.5 estimates the probability

$$\mathrm{Prob}([\boldsymbol{x}(k_h)] \mid [\boldsymbol{U}(0 \,...\, k_h)], [\boldsymbol{Y}(0 \,...\, k_h)])$$

that the qualitative state has the value $[\boldsymbol{x}(k_h)]$ under the condition that the given input and output sequences occurred. The current qualitative state belongs to the set

$$\boldsymbol{\mathcal{X}}(k_h \mid k_h) = \{[\boldsymbol{x}(k_h)] \,:\, \mathrm{Prob}([\boldsymbol{x}(k_h)] \mid [\boldsymbol{U}(0 \,...\, k_h)], [\boldsymbol{Y}(0 \,...\, k_h)]) > 0\}\,, \quad (9.1)$$

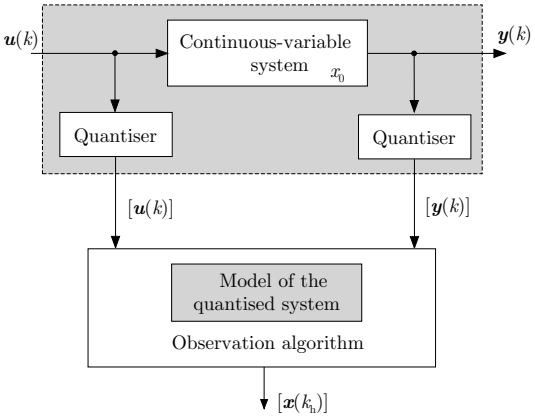


Fig. 9.2. State observation of quantised systems

where the notation $\boldsymbol{\mathcal{X}}(k_h \mid k_h)$ means that the state at time k_h is reconstructed for given qualitative input and output sequences up to time k_h.

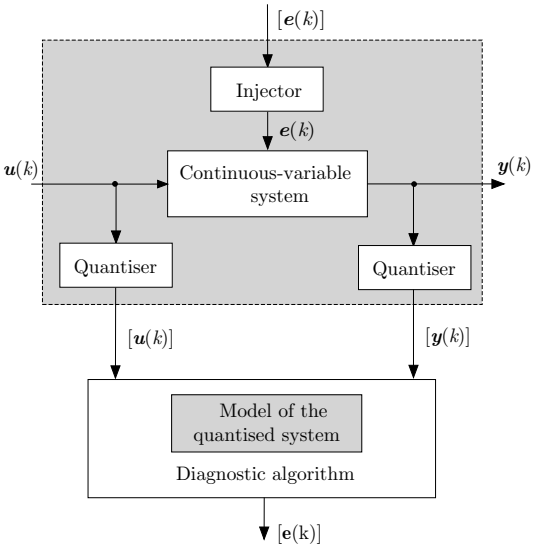


Fig. 9.3. Diagnosis of quantised systems

Process diagnosis. The diagnostic task for quantised systems concerns the problem of finding the symbolic fault value f from the quantised measurement sequences:

Problem 9.2 (Diagnostic problem for quantised systems)
Given: *Sequence of quantised input values.*
Sequence of quantised output values.
Model $\mathcal{M}$ of the quantised system.
Find: *Fault* $f(k_h) = [\boldsymbol{e}(k_h)]$.

The algorithm presented in Section 9.6 estimates the probability

$$\mathrm{Prob}(f(k_h) \,|\, [\boldsymbol{U}(0 \,...\, k_h)], [\boldsymbol{Y}(0 \,...\, k_h)])$$

that a fault $f(k_h) = [\boldsymbol{e}(k_h)]$ has occurred provided that the quantised system with qualitative input sequence $[\boldsymbol{U}(0 \,...\, k_h)]$ has produced the qualitative output sequence $[\boldsymbol{Y}(0 \,...\, k_h)]$. In on-line applications, this task is solved for increasing time horizon $k_h = 1, 2, ...$ and leads to the set

$$\mathcal{F}(k_h \mid k_h) = \{f(k_h) \,:\, \mathrm{Prob}(f(k_h) \mid [\boldsymbol{U}(0 \,...\, k_h)], [\boldsymbol{Y}(0 \,...\, k_h)]) > 0\} \qquad (9.2)$$

of fault candidates.

9.2 Quantised systems

9.2.1 Continuous-variable system

The core of a quantised system is the continuous-variable discrete-time system

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{g}(\boldsymbol{x}(k), \boldsymbol{u}(k)), \qquad \boldsymbol{x}(0) \in \mathcal{X}_0 && (9.3)\\ \boldsymbol{y}(k) &= \boldsymbol{h}(\boldsymbol{x}(k), \boldsymbol{u}(k)) && (9.4) \end{aligned}$$

with input vector $\boldsymbol{u} \in \mathbb{R}^m$, output vector $\boldsymbol{y} \in \mathbb{R}^r$ and the vector of the internal state $\boldsymbol{x} \in \mathbb{R}^n$. $\mathcal{X}_0 \subseteq \mathbb{R}^n$ is the set of initial states that the system can assume. If $\boldsymbol{x}(0)$ is known, this set is a singleton. However, as the state $\boldsymbol{x}$ is not measurable, $\mathcal{X}_0$ is usually a subset of the state space $\mathbb{R}^n$.

It is assumed that for any initial state $\boldsymbol{x}(0) \in \mathcal{X}_0$ and input sequence

$$\boldsymbol{U}(0 \,...\, k_h) = (\boldsymbol{u}(0), \boldsymbol{u}(1), \,..., \boldsymbol{u}(k_h))$$

Eqs. (9.3), (9.4) generate a unique state and a unique output sequence

$$\begin{aligned} \boldsymbol{X}(0 \,...\, k_h) &= (\boldsymbol{x}(0), \boldsymbol{x}(1), \,..., \boldsymbol{x}(k_h))\\ \boldsymbol{Y}(0 \,...\, k_h) &= (\boldsymbol{y}(0), \boldsymbol{y}(1), \,..., \boldsymbol{y}(k_h)), \end{aligned}$$

which are considered over the time interval $[0, k_h]$.

If the system is linear, Eqs. (9.3), (9.4) have the form

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k), \qquad \boldsymbol{x}(0) \in \mathcal{X}_0 && (9.5)\\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) && (9.6) \end{aligned}$$

with matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ of appropriate dimensions. Then, explicit solutions are known:

$$\begin{aligned}\boldsymbol{x}(k) &= \boldsymbol{A}^k\boldsymbol{x}(0) + \sum_{i=0}^{k-1}\boldsymbol{A}^{k-1-i}\boldsymbol{B}\boldsymbol{u}(i)\\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{A}^k\boldsymbol{x}(0) + \sum_{i=0}^{k-1}\boldsymbol{C}\boldsymbol{A}^{k-1-i}\boldsymbol{B}\boldsymbol{u}(i) + \boldsymbol{D}\boldsymbol{u}(k).\end{aligned}$$

Faults occurring in the system are described by an additional input signal $\boldsymbol{e}(k) \in \mathbb{R}^p$, which for the time horizon k_h is given by the sequence

$$\boldsymbol{E}(0\,...\,k_h) = (\boldsymbol{e}(0), \boldsymbol{e}(1),\, ...,\, \boldsymbol{e}(k_h))\,.$$

Accordingly, the state-space model has to be extended to become

$$\boldsymbol{x}(k+1) = \boldsymbol{g}(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{e}(k)), \quad \boldsymbol{x}(0) \in \mathcal{X}_0 \tag{9.7}$$

$$\boldsymbol{y}(k) = \boldsymbol{h}(\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{e}(k))\,. \tag{9.8}$$

For any given input and fault sequences $\boldsymbol{U}$ and $\boldsymbol{E}$ this system is assumed to generate unique state and output sequences $\boldsymbol{X}$ and $\boldsymbol{Y}$.

9.2.2 Quantisation of the signal spaces

The continuous-variable system is considered in quantised signal spaces. The *quantisers* introduce partitions of the signal spaces $\mathbb{R}^m$ and $\mathbb{R}^r$ into a finite number of disjoint sets $\mathcal{Q}_u(\nu)$ ($\nu \in \mathcal{N}_u = \{0, 1, ..., M\}$) and $\mathcal{Q}_y(\omega)$ ($\omega \in \mathcal{N}_y = \{0, 1, ..., R\}$), where $\mathcal{Q}_u(\nu)$ or $\mathcal{Q}_y(\omega)$ denote the set of input values $\boldsymbol{u}$ or output values $\boldsymbol{y}$ with the same quantised values ν or ω. The mapping invoked by the quantiser is symbolised by $[\cdot]$:

$$[\boldsymbol{u}] = \nu \iff \boldsymbol{u} \in \mathcal{Q}_u(\nu) \tag{9.9}$$

$$[\boldsymbol{y}] = \omega \iff \boldsymbol{y} \in \mathcal{Q}_y(\omega). \tag{9.10}$$

The numbers ν or ω are called the quantised values or the *qualitative values* of the input or output, respectively, and $[\boldsymbol{u}]$ or $[\boldsymbol{y}]$ the *qualitative input* or *qualitative output*.

The sets $\mathcal{Q}_u(\nu)$ ($\nu \neq 0$) and $\mathcal{Q}_y(\omega)$ ($\omega \neq 0$) are assumed to be bounded while $\mathcal{Q}_u(0)$ and $\mathcal{Q}_y(0)$ are the unbounded "remaining" subsets of $\mathbb{R}^m$ or $\mathbb{R}^r$, respectively.

For discrete input or output, the quantisation is equivalent to the enumeration of the discrete signal values. For the tank system introduced in Section 2.1 the input $Pos(V_1)$ has two values, which are denoted by 1 and 0: $[Pos(V_1)] \in \{0, 1\}$.

In order to get a concise model of the quantised system, it is reasonable to introduce a quantisation of the state space $\mathbb{R}^n$ into the partitions $\mathcal{Q}_x(\zeta)$ ($\zeta \in \mathcal{N}_x = \{0, 1, ..., N\}$) with

$$[\boldsymbol{x}] = \zeta \iff \boldsymbol{x} \in \mathcal{Q}_x(\zeta).$$

The number ζ of the partition $\mathcal{Q}_x(\zeta)$ to which the current state $\boldsymbol{x}$ belongs is called the *qualitative state* (although it is, more precisely, the qualitative value of the state).

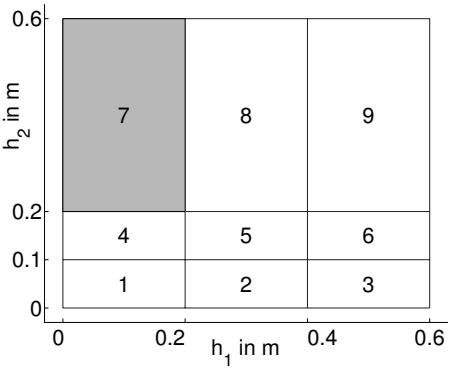


Fig. 9.4. Partition of a two-dimensional state space of the tank system

Figure 9.4 shows an example for the partition of the two-dimensional state space of the tank system. The grey region includes all states $\boldsymbol{x} = (h_1, h_2)'$ with the same qualitative value $[\boldsymbol{x}] = 7$. The two-dimensional state space outside the nine partitions constitutes the unbounded partition $Q_x(0)$.

A consequence of the introduction of the quantiser is the fact that no distinction can be made between different input, state and output values that belong to the same region $\mathcal{Q}_u(\nu)$, $\mathcal{Q}_x(\zeta)$ or $\mathcal{Q}_y(\omega)$. In an application, the regions have to be chosen in such a way that it does not matter for the solution of the process supervision task which real-valued input, state or output of a given region really occurs. The input or output quantisation may be given by practical circumstances, for example, by the sensor locations. The state quantisation can usually be arbitrarily chosen, which gives the possibility to adapt this quantisation to the dynamical phenomena that occur in the system.

Fault injector. The injector shown in Fig. 9.1 transforms the qualitative fault value $[\boldsymbol{e}]$ into a real-valued signal $\boldsymbol{e}$. This transformation can be considered as the inverse operation of quantisation. Every qualitative value $f = [\boldsymbol{e}]$ is associated with a partition $\mathcal{Q}_e$ of the signal space $\mathbb{R}^p$ of the fault signal $\boldsymbol{e}$. As only the qualitative fault value f is assumed to be known, the fault signal $\boldsymbol{e}$ is only known to belong to the partition $\mathcal{Q}_e(f)$:

$$[\boldsymbol{e}] = f \quad \Longleftrightarrow \quad \boldsymbol{e} \in \mathcal{Q}_e(f).$$

In the example considered in the following section the leakage in a tank is described by the flow constant c_2 of a hole which is partitioned into two intervals. The first inverval represents a very small leakage that is considered as negligible (faultless) and the second interval corresponds to the fault.

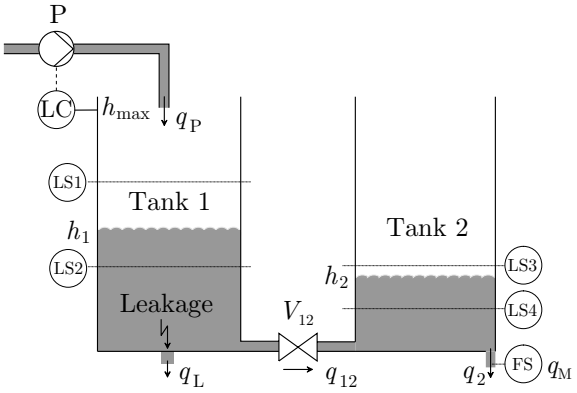


Fig. 9.5. Two-tank system with quantised level measurement

Example 9.1 *Quantised two-tank system*

Consider the tank example introduced in Section 2.1 and assume that the sensors give only quantised information (Fig. 9.5). That is, the sensors merely signal whether the liquid level is above or below their position. The result is a quantised measurement information for the tank levels h_1 and h_2 and the outflow q_M of Tank 2. The quantisation intervals are summarised in Table 9.1.

Table 9.1 Signal quantisation

$[h_1] = 1$	if	$0\,\mathrm{m} \leq h_1 < 0.2\,\mathrm{m}$	$\Leftrightarrow \mathrm{LS2} = 0$
$[h_1] = 2$	if	$0.2\,\mathrm{m} \leq h_1 < 0.4\,\mathrm{m}$	$\Leftrightarrow \mathrm{LS2} = 1 \wedge \mathrm{LS1} = 0$
$[h_1] = 3$	if	$0.4\,\mathrm{m} \leq h_1 < 0.6\,\mathrm{m}$	$\Leftrightarrow \mathrm{LS1} = 1$
$[h_2] = 1$	if	$0\,\mathrm{m} \leq h_2 < 0.1\,\mathrm{m}$	$\Leftrightarrow \mathrm{LS4} = 0$
$[h_2] = 2$	if	$0.1\,\mathrm{m} \leq h_2 < 0.2\,\mathrm{m}$	$\Leftrightarrow \mathrm{LS4} = 1 \wedge \mathrm{LS3} = 0$
$[h_2] = 3$	if	$0.2\,\mathrm{m} \leq h_2 < 0.6\,\mathrm{m}$	$\Leftrightarrow \mathrm{LS3} = 1$
$[q_M] = 1$	if	$0\,\mathrm{l/min} \leq q_M < 3\,\mathrm{l/min}$	
$[q_M] = 2$	if	$3\,\mathrm{l/min} \leq q_M < 6\,\mathrm{l/min}$	
$[q_M] = 3$	if	$6\,\mathrm{l/min} \leq q_M < 10\,\mathrm{l/min}$	

With this quantisation, the input and output of the tank system have the signal values summarised in Table 9.2. The two quantised level measurements $[h_1]$ and $[h_2]$ yield the partition of the output space. Every region $[(h_1, h_2)] \in \{1,\ 2,\ ...,\ 9\}$ corresponds to one combination of quantised values for $[h_1]$ and $[h_2]$. In the following either the quantised level measurement $[(h_1,\ h_2)]$ or the quantised outflow $[q_M]$ is used as quantised output. □

Table 9.2 Quantised input, output and faults of the tank system

Symbol	Value set	Meaning
	Inputs	
$[V_{12}]$	$\{1, 2\}$	Connecting valve closed for $[V_{12}] = 1$, open for $[V_{12}] = 2$
$[u_P]$	$\{1, 2\}$	Pump off for $[u_P] = 1$, nominal velocity for $[u_P] = 2$
	Outputs	
$[h_1, h_2]$	$\{1, 2, ..., 9\}$	Quantised level of the tanks
$[q_M]$	$\{1, 2, 3\}$	Quantised outflow of Tank 2
	Fault	
$[c_L]$	$\{1, 2\}$	No leakage for $[c_L] = 1$, leakage for $[c_L] = 2$

9.2.3 Behaviour of quantised systems

In the succeeding investigations the main ideas of modelling quantised systems and observing the qualitative state of such systems will be explained without reference to possible faults. Therefore, the faultless system (9.3), (9.4) will be considered.

The behaviour of the quantised system is the set of all I/O pairs

$$([\boldsymbol{U}(0 \ldots k_h)], [\boldsymbol{Y}(0 \ldots k_h)])$$

that are consistent with the system dynamics and the signal quantisation. As the qualitative input and output are considered, the behaviour is also referred to as the *qualitative behaviour* of the system (9.3), (9.4):

$$\mathcal{B}_{\text{qual}}(k_h) = \{([\boldsymbol{U}(0 \ldots k_h)], [\boldsymbol{Y}(0 \ldots k_h)]) \ : \ \text{Eqs. (9.3), (9.4) hold}\}. \tag{9.11}$$

As all measurements are qualitative, the initial state of the system is considered on the qualitative level of abstraction as well. This is why all the following investigations concern the set of those qualitative I/O pairs that the quantised system can generate for a *qualitatively* given initial state. If the qualitative initial state $\zeta(0) = [\boldsymbol{x}(0)]$ is precisely known, $\mathcal{X}_0 = \mathcal{Q}_x(\zeta(0))$ holds. If the qualitative initial state is unknown, the set $\mathcal{X}_0$ is the union of several state partitions $\mathcal{Q}_x(\zeta)$ or $\mathcal{X}_0 = \mathbb{R}^n$ holds.

In order to get a better imagination of the qualitative behaviour $\mathcal{B}_{\text{qual}}$ note that for the elements of the I/O sequences the relations

$$[\boldsymbol{u}(k)] \in \mathcal{N}_u \quad = \quad \{0, 1, \ ..., \ M\} \tag{9.12}$$

$$[\boldsymbol{y}(k)] \in \mathcal{N}_y \quad = \quad \{0, 1, \ ..., \ R\} \tag{9.13}$$

and, hence,

$$[\boldsymbol{U}(0 \ldots k_h)] \in \mathcal{N}_u^{k_h+1} \quad = \quad \mathcal{N}_u \times \mathcal{N}_u \times ... \times \mathcal{N}_u \tag{9.14}$$

$$[\boldsymbol{Y}(0 \ldots k_h)] \in \mathcal{N}_y^{k_h+1} \quad = \quad \mathcal{N}_y \times \mathcal{N}_y \times ... \times \mathcal{N}_y \tag{9.15}$$

hold, where the Cartesian products on the right-hand side include the given sets $k_h + 1$ times. Consequently, the qualitative behaviour is a subset of the Cartesian product of $\mathcal{N}_u^{k_h+1}$ and $\mathcal{N}_y^{k_h+1}$:

$$\mathcal{B}_{\text{qual}}(k_h) \subseteq \mathcal{N}_u^{k_h+1} \times \mathcal{N}_y^{k_h+1}. \tag{9.16}$$

The number of elements of this product is $((M+1)\cdot(R+1))^{k_h+1}$. If, for example, only 3 qualitative input and 4 qualitative output signals are considered and the time horizon is $k_h = 4$, this number is $160,000$. The behaviour $\mathcal{B}_{\text{qual}}(k_h)$ selects for a given time horizon k_h those elements out of this large set, which are consistent with the system. How large this number of elements is depends on the system properties. If the system were deterministic (which it is generally not as described below), it would generate a unique output sequence $[\boldsymbol{Y}(0\,...\,k_h)]$ for every given input sequence $[\boldsymbol{U}(0\,...\,k_h)]$ and initial state. In the given example, 243 different input sequences $[\boldsymbol{U}(0...4)]$ exist, which lead to the same number of elements of $\mathcal{B}_{\text{qual}}(4)$ for every initial state.

If later a faulty system is concerned, the behaviour is represented by all triples $([\boldsymbol{U}], [\boldsymbol{E}], [\boldsymbol{Y}])$ that are consistent with the system (9.7), (9.8), the quantisers and the injector. It is given by

$$\begin{aligned}&\mathcal{B}_{\text{qual}}(k_h) = \\ &\{([\boldsymbol{U}(0\,...\,k_h)], [\boldsymbol{E}(0\,...\,k_h)], [\boldsymbol{Y}(0\,...\,k_h)]) \;:\; \text{Eq. (9.7), (9.8) hold}\}.\end{aligned} \tag{9.17}$$

Non-determinism of the qualitative behaviour. An important issue is the fact that it is impossible to predict the qualitative output sequence of a quantised system unambiguously for given qualitative initial state and qualitative input. The set $\mathcal{B}_{\text{qual}}(k_h)$ includes, in general, more than one element $([\boldsymbol{U}], [\boldsymbol{Y}])$ with the same qualitative input sequence $[\boldsymbol{U}]$. The reason for this is given by the fact that the system (9.3), (9.4) may start from any initial state $\boldsymbol{x}(0)$ with the given qualitative value $\zeta(0) = [\boldsymbol{x}(0)]$ and may obtain any input sequence $\boldsymbol{U}$ which is only described by the qualitative sequence $[\boldsymbol{U}]$. For these sets of initial states and input sequences, the resulting output sequences $\boldsymbol{Y}$ yield, in general, different qualitative sequences $[\boldsymbol{Y}]$. This phenomenon is referred to as the *non-determinism of the qualitative behaviour*.

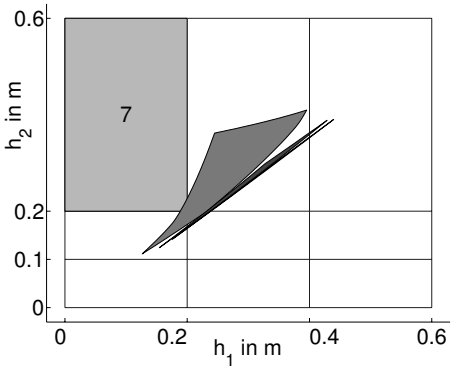


Fig. 9.6. Sets of states reached by the tank system for qualitative initial state $\boldsymbol{x}(0) \in \mathcal{Q}_x(7)$

Example 9.2 *Non-determinism of the qualitative behaviour of the tank system*

The reason why the qualitative behaviour is non-deterministic is illustrated by Fig. 9.6, which shows the set of state trajectories $\boldsymbol{X}$ of the tank system for the initial quantised set of states $[(h_1, h_2)'] = 7$. At time $k = 0$ the tank system may assume any state in the region 7 of the partitioned state space, because the initial state is only qualitatively known. For time $k = 1, 2, 3$, the system may be in any state of the succeeding regions, which were determined for constant input $[V_{12}] = 2$ and $[u_P] = 2$. The important point is that these regions overlap with more than one state partition. Hence, the system generates different qualitative state trajectories

$$[\boldsymbol{X}] = ([\boldsymbol{x}(0)], [\boldsymbol{x}(1)], [\boldsymbol{x}(2)], [\boldsymbol{x}(3)], ...),$$

for example

$$\begin{aligned} [\boldsymbol{X}(0...3)] &= (7, 8, 8, 8) \\ [\boldsymbol{X}(0...3)] &= (7, 4, 4, 4) \\ [\boldsymbol{X}(0...3)] &= (7, 5, 8, 8) \\ [\boldsymbol{X}(0...3)] &= (7, 8, 9, 9) \end{aligned}$$

and, hence, different output trajectories. If, for example, the output is identical to the second level h_2, the quantised output sequences are

$$\begin{aligned} [\boldsymbol{Y}(0...3)] &= (3, 3, 3, 3) \\ [\boldsymbol{Y}(0...3)] &= (3, 2, 2, 2) \\ [\boldsymbol{Y}(0...3)] &= (3, 2, 3, 3), \end{aligned}$$

where the output $y = h_2$ with three qualitative values has been used. Figure 9.7 shows a graphical representation of the set of qualitative state sequences.

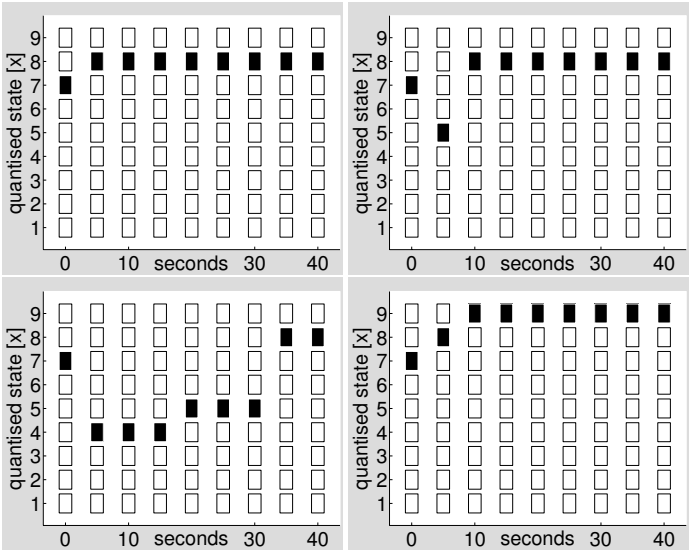
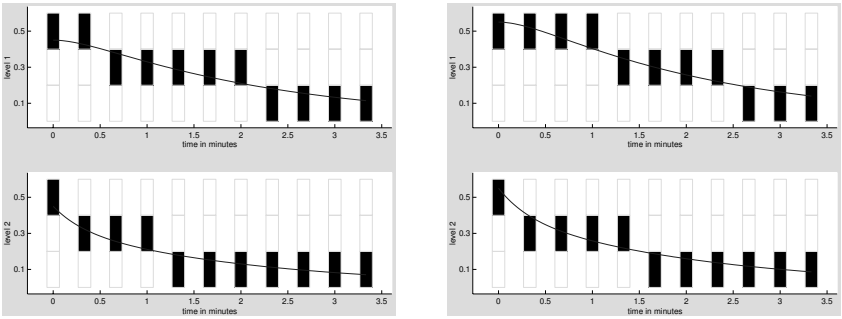


Fig. 9.7. Four quantised trajectories of the tank system for the same quantised initial state $[\boldsymbol{x}_0] = 7$ and constant input $[V_{12}] = 2$, $[u_P] = 2$

This example shows that the non-determinism occurs because a bundle of trajectories has to be considered rather than a unique trajectory. This bundle starts in the common qualitative state $[\boldsymbol{x}_0] = 7$ and has a common qualitative input sequence. The trajectories of this bundle generate different qualitative sequences $[\boldsymbol{Y}]$ and, hence, the qualitative output sequence cannot be predicted unambiguously for given qualitative input sequence and qualitative initial state.



(a) (b)

Fig. 9.8. Two different quantised trajectories of the tank system starting in the same quantised initial state $[\boldsymbol{x}_0]$

Figure 9.8 shows how this non-determinism becomes obvious in experiments with the tank system. In contrast to Fig. 9.7, Fig. 9.8 shows the two quantised tank levels rather than the enumerated state. The two parts of the figure concern two experiments, in which the initial liquid levels are qualitatively the same, but differ quantitatively. The thin lines show the quantitative tank behaviour. Obviously, the resulting qualitative trajectories are different.

This phenomenon has a direct consequence concerning the assertions that an experienced human operator can make about the tank system. If asked about the qualitative tank levels at time $k = 1$ under the assumption that the qualitative initial state is 7, the operator can only predict a set of qualitative levels rather than a unique one. This is, because the operator has merely the qualitative knowledge that Tank 1 has initially a low level h_1 and Tank 2 a high level h_2. The uncertainty of the knowledge about the future tank level is not based on insufficient knowledge about the dynamics of the system under consideration but on the uncertainty of the initial levels in both tanks resulting from the quantised information available. □

The example makes it obvious that the quantised system behaves like a stochastic process. The initial state $\boldsymbol{x}(0)$ can be assumed to be chosen randomly among all initial states with the given qualitative value $\zeta(0) = [\boldsymbol{x}(0)]$. In a more general experiment, also the input sequence $\boldsymbol{U}$ can be chosen randomly among all input sequences with a fixed qualitative value $[\boldsymbol{U}]$. For each of these initial states and input sequences the system generates a unique qualitative output sequence $[\boldsymbol{Y}]$, but since the initial state and the input vary from experiment to experiment so do the qualitative output sequences. No unambiguous prediction of $[\boldsymbol{Y}]$ can be made, but some probabilistic prediction is possible. Such a prediction will be considered in Section 9.4, where a stochastic automaton will be set up that generates a probability distribution for all the qualitative output values generated by the quantised system.

9.2.4 Stochastic properties of quantised systems

In the following, the set $\mathcal{X}_0$ of initial states $\boldsymbol{x}_0$ considered is assumed to be qualitatively given. That is, it is the union of one or more state partitions $Q_x(\zeta)$. The initial state is assumed to be randomly distributed over $\mathcal{X}_0$. Under these circumstances, consider the probability

$$\mathrm{Prob}([\boldsymbol{Y}(0\,...\,k_h)]\,|\,[\boldsymbol{U}(0\,...\,k_h)], \mathcal{X}_0)$$

with which the qualitative output sequence $[\boldsymbol{Y}]$ occurs for a given qualitative input sequence $[\boldsymbol{U}]$ and qualitatively given initial state set $\mathcal{X}_0$. This probability characterises how often the I/O pair $([\boldsymbol{U}], [\boldsymbol{Y}])$ occurs if the probability of the occurrence of $[\boldsymbol{U}]$ and, moreover, the probability of the initial state $\boldsymbol{x}(0) \in \mathcal{X}_0$ are known. Note that the dependency of $\mathrm{Prob}([\boldsymbol{Y}(0\,...\,k_h)]\,|\,[\boldsymbol{U}(0\,...\,k_h)], \mathcal{X}_0)$ upon $\boldsymbol{x}(0)$ is given implicitly by Eqs. (9.3) and (9.4).

To better understand the meaning of this probability, assume that the initial state $\boldsymbol{x}(0)$ is known to belong to one state partion $Q_x(\zeta_0)$ for given ζ_0. Then $\mathrm{Prob}([\boldsymbol{Y}(0\,...\,k_h)]\,|\,[\boldsymbol{U}(0\,...\,k_h)], \mathcal{X}_0)$ says how often $[\boldsymbol{Y}]$ occurs for a given input sequence $[\boldsymbol{U}]$ if many experiments are made with the system starting in the same initial state $\boldsymbol{x}_0 \in \mathcal{X}_0$. If a specific I/O pair $([\boldsymbol{U}], [\boldsymbol{Y}])$ never occurs, because the system cannot follow the qualitative output sequence $[\boldsymbol{Y}]$ for the qualitative input sequence $[\boldsymbol{U}]$ and for some initial state $\boldsymbol{x}(0) \in \mathcal{X}_0$, then

$$\mathrm{Prob}([\boldsymbol{Y}(0\,...\,k_h)]\,|\,[\boldsymbol{U}(0\,...\,k_h)], \mathcal{X}_0) = 0$$

holds. Other I/O pairs may occur with different frequencies, which lead to positive probability values.

If this probability should be determined, the "experiments" considered just now have to be investigated in more detail. Since $[\boldsymbol{U}]$ is the input to the quantised system whereas $[\boldsymbol{Y}]$ describes the effect of this input together with the initial state, the input $\boldsymbol{U}$ and the initial state $\boldsymbol{x}(0)$ have to be varied from experiment to experiment within the given sets of qualitatively equivalent input sequences and initial states. As the frequency, with which a certain output sequence $\boldsymbol{Y}$ occurs, also depends on how often a certain input sequence $\boldsymbol{U}$ and initial state $\boldsymbol{x}(0)$ is chosen, the probability distribution of these variables have to be fixed. This can be done, in principle, in an arbitrary way. To understand the following investigations, it can be assumed that *uniform* probability distributions are used. Then every value $\boldsymbol{x}(0)$ that belongs to the set $\mathcal{X}_0$ occurs with the same probability. The same holds for the input sequences. However, the following investigations are valid for arbitrary probability distributions.

The experiments to be made have to bring the system in the chosen initial state $\boldsymbol{x}(0)$ and to determine the qualitative system trajectory $[\boldsymbol{Y}(0\,...\,k_h)]$ for the chosen input sequence $\boldsymbol{U}(0\,...\,k_h)$. After all experiments have been made, the relative frequencies of the occurrence of the different qualitative output sequences $[\boldsymbol{Y}]$ give the (approximate) value of the probability $\mathrm{Prob}([\boldsymbol{Y}(0\,...\,k_h)]\,|\,[\boldsymbol{U}(0\,...\,k_h)], \mathcal{X}_0)$ to be found.

With this probability, the qualitative behaviour of the quantised system defined in Eq. (9.11) can be expressed in the form

$$\begin{aligned}&\mathcal{B}_{\text{qual}}(k_h) \\ &= \{[\boldsymbol{U}(0\,...\,k_h)], [\boldsymbol{Y}(0\,...\,k_h)] \,:\, \text{Prob}([\boldsymbol{Y}(0\,...\,k_h)] \,|\, [\boldsymbol{U}(0\,...\,k_h)], \mathcal{X}_0) > 0\}.\end{aligned} \tag{9.18}$$

As the number of elements of $\mathcal{B}_{\text{qual}}(k_h)$ is very large (cf. Section 9.2.3), it is not possible to graphically illustrate the probability of the I/O pairs. However, it is often more interesting to know with which probability a specific qualitative output value $[\boldsymbol{y}(k_h)]$ occurs. This probability can be obtained as boundary probability

$$\begin{aligned}&\text{Prob}([\boldsymbol{y}(k_h)] \,|\, [\boldsymbol{U}(0\,...\,k_h)], \mathcal{X}_0) \\ &= \sum_{[\boldsymbol{Y}(0\,...\,k_h-1)]} \text{Prob}([\boldsymbol{Y}(0\,...\,k_h)] \,|\, [\boldsymbol{U}(0\,...\,k_h)], \mathcal{X}_0) \\ &= \sum_{[\boldsymbol{y}(0)],\,...,\,[\boldsymbol{y}(k_h-1)]} \text{Prob}([\boldsymbol{y}(0)], ..., [\boldsymbol{y}(k_h)] \,|\, [\boldsymbol{U}(0\,...\,k_h)], \mathcal{X}_0),\end{aligned} \tag{9.19}$$

where the summation is made over all elements $[\boldsymbol{y}(k)]$ of the output sequence with the exception of the last element $[\boldsymbol{y}(k_h)]$. This probability is depicted in Fig. 9.9 for the tank system with $\boldsymbol{y} = \boldsymbol{x} = (h_1, h_2)'$. This figure summarises the information given by the four state sequences depicted in Fig. 9.7 and associates these sequences with the probability of their occurrence. The darker the rectangles are, the higher is the probability. White rectangles show that the corresponding qualitative output value cannot occur at the corresponding time instant.

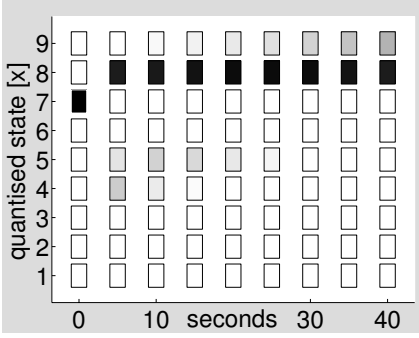


Fig. 9.9. Statistical properties of the tank system

Markov property. The probability considered so far concerns sequences of qualitative input and output values of arbitrary time horizon k_h:

$$\text{Prob}([\boldsymbol{y}(0)], [\boldsymbol{y}(1)], \,...,\, [\boldsymbol{y}(k_h)] \mid [\boldsymbol{u}(0)], [\boldsymbol{u}(1)], \,...,\, [\boldsymbol{u}(k_h)], \mathcal{X}_0).$$

As these sequences may be arbitrarily long, it is impossible to store all the probability distributions,

$$\text{Prob}([\boldsymbol{y}(0)] \mid [\boldsymbol{u}(0)], \mathcal{X}_0)$$

$$\begin{gathered}\mathrm{Prob}([\boldsymbol{y}(0)],[\boldsymbol{y}(1)] \mid [\boldsymbol{u}(0)],[\boldsymbol{u}(1)],\mathcal{X}_0)\\ \mathrm{Prob}([\boldsymbol{y}(0)],[\boldsymbol{y}(1)],[\boldsymbol{y}(2)] \mid [\boldsymbol{u}(0)],[\boldsymbol{u}(1)],[\boldsymbol{u}(2)],\mathcal{X}_0)\\ \vdots\\ \mathrm{Prob}([\boldsymbol{y}(0)],[\boldsymbol{y}(1)],\ ...,\ [\boldsymbol{y}(k_h)] \mid [\boldsymbol{u}(0)],[\boldsymbol{u}(1)],\ ...,\ [\boldsymbol{u}(k_h)],\mathcal{X}_0)\end{gathered}$$

for all arguments. Therefore, it is interesting to known whether it is possible to determine these probability distributions recursively, where the probability distribution obtained for the time horizon k_h is determined from the probability distribution for the time horizon $k_h - 1$. Similarly, the probability considered in Eq. (9.19) should be determined from the probability of the output $[\boldsymbol{y}(k-1)]$.

Such recursive representations are possible, if the system possesses the Markov property. Then a transition probability p_{tr} exists with which the probability for time k_h can be obtained from the probability of time $k-1$.

In general, dynamical systems possess the Markov property with respect to the state $\boldsymbol{x}$ although this property does not hold with respect to the output $\boldsymbol{y}$. This becomes obvious from Eq. (9.3),

$$\boldsymbol{x}(k+1) = \boldsymbol{g}(\boldsymbol{x}(k),\boldsymbol{u}(k)), \qquad \boldsymbol{x}(0) \in \mathcal{X}_0$$

where the state transition is represented by the function $\boldsymbol{g}$. This equation says that the state at time $k+1$ can be unambiguously determined from state $\boldsymbol{x}(k)$ without the knowledge of the states $\boldsymbol{x}(k-1)$, $\boldsymbol{x}(k-2)$ etc. occurring in the further history of the system. In a probabilistic setting, the probability of the state transition is one for the pair of states $(\boldsymbol{x}(k+1),\boldsymbol{x}(k))$ that occur in this equation together with the input $\boldsymbol{u}(k)$ and it is zero for all other pairs.

For the quantised system, the state $\boldsymbol{x}$ is replaced by the qualitative state $[\boldsymbol{x}]$. The Markov property would make it possible to represent the probability $\mathrm{Prob}([\boldsymbol{x}(k+1)])$ in dependence upon $\mathrm{Prob}([\boldsymbol{x}(k)])$. Then the relation

$$\mathrm{Prob}([\boldsymbol{x}(k+1)]) = \tilde{\boldsymbol{g}}(\mathrm{Prob}([\boldsymbol{x}(k)]),\mathrm{Prob}([\boldsymbol{u}(k)])) \tag{9.20}$$

would hold for some function $\tilde{\boldsymbol{g}}$. However, in general, a quantised system does *not* possess the Markov property and, hence, such a recursive representation does not exist. This has severe consequences for the solution of the modelling task which will be investigated in the next section:

> As the quantised system does not possess the Markov property with respect to the quantised input, state and output signals, every model that possesses the Markov property, can only be an approximate representation of the quantised system.

Example 9.3 *Violation of the Markov property by the tank system*

In order to explain why generally quantised systems do not possess the Markov property, Fig. 9.10 shows the movement of the initial set of states $\mathcal{X}_0$ of the tank system in the quantised state space for the constant input $[V_{12}] = 2$ and $[u_P] = 1$. Since the qualitative initial state is assumed to be $[\boldsymbol{x}(0)] = 3$,

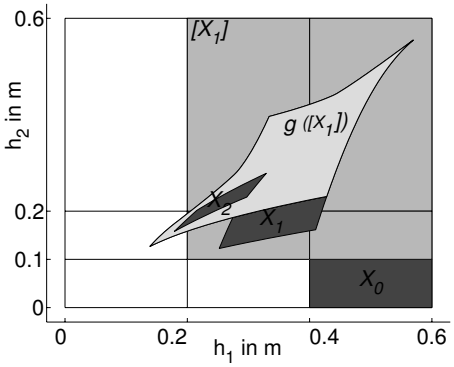


Fig. 9.10. Discussion of the Markov property of the quantised system

$$\mathcal{X}_0 = \mathcal{Q}_x(3),$$

the initial state $\boldsymbol{x}(0)$ lies in the partition denoted by X_0. The sets X_1 and X_2 are the sets of successor states $\boldsymbol{x}(1)$ and $\boldsymbol{x}(2)$ that can be obtained by the model given in Section 2.1. Obviously, $[\boldsymbol{x}(1)] \in [X_1] = \{5, 6, 8, 9\}$ and $[\boldsymbol{x}(2)] \in [X_2] = \{4, 5, 8\}$ hold. These results are obtained from a quantisation of the trajectory bundle of the continuous-variable system (9.3) that starts in the partition $\mathcal{Q}_x(3)$.

Now consider what a qualitative model of the form (9.20) can do. As the input is constant, the system can be considered as autonomous with the simpler model

$$\mathrm{Prob}([\boldsymbol{x}(k+1)]) = \tilde{\boldsymbol{g}}(\mathrm{Prob}([\boldsymbol{x}(k)])) \; .$$

For $k = 0$ and $\mathrm{Prob}([\boldsymbol{x}(0)]=3) = 1$ the function $\tilde{\boldsymbol{g}}$ yields

$$\begin{aligned} \mathrm{Prob}([\boldsymbol{x}(1)]=8) &= 0.11, \qquad & \mathrm{Prob}([\boldsymbol{x}(1)]=9) &= 0.06 \\ \mathrm{Prob}([\boldsymbol{x}(1)]=5) &= 0.78, \qquad & \mathrm{Prob}([\boldsymbol{x}(1)]=6) &= 0.05 \end{aligned}$$

which is related to the relative overlap of the set X_1 with the partitions $\mathcal{Q}_x(5)$, $\mathcal{Q}_x(6)$, $\mathcal{Q}_x(8)$ and $\mathcal{Q}_x(9)$ (for a detailed analysis cf. Section 9.4). For $k = 1$ the function $\tilde{\boldsymbol{g}}$ has to determine $\mathrm{Prob}([\boldsymbol{x}(k+1)])$ by using this result only. As this function does not know that the system has started at $k = 0$ in the set $\mathcal{Q}_x(3)$, it assumes that the state can lie anywhere in the sets $\mathcal{Q}_x(5)$, $\mathcal{Q}_x(6)$, $\mathcal{Q}_x(8)$ and $\mathcal{Q}_x(9)$ whose union

$$[X_1] = \mathcal{Q}_x(5) \cup \mathcal{Q}_x(6) \cup \mathcal{Q}_x(8) \cup \mathcal{Q}_x(9)$$

is depicted in medium grey in Fig. 9.10. Therefore, the function $\tilde{\boldsymbol{g}}$ concerns the mapping of this set, which results in the light grey set in Fig. 9.10 which conservatively overapproximates the true set $X_2 = \boldsymbol{g}(\boldsymbol{g}(X_0))$ denoted by $\boldsymbol{g}([X_1])$. The light grey set intersects with the regions $4, 5, 8$ and 9 of the partitioned state space. Consequently, $\tilde{\boldsymbol{g}}$ yields a positive probability $\mathrm{Prob}([\boldsymbol{x}(2)] = 9) > 0$ although the set X_2 does not intersect with $\mathcal{Q}_x(9)$ and, hence, the correct result is $\mathrm{Prob}([\boldsymbol{x}(2)]=9) = 0$.

The reason for the "error" in the calculation with the model 9.20 is the missing Markov property. The example demonstrates that with the additional knowledge about $[\boldsymbol{x}(0)]$ the determination of $[\boldsymbol{x}(2)]$ is better compared to the determination based on the information about $[\boldsymbol{x}(1)]$ only. In contrast to this, the Markov property requires that this additional knowledge has no effect. □

9.3 A behavioural view on the process supervision problems for quantised systems

As the Fig. 9.2 and 9.3 show, the observation and the diagnostic methods have access to the I/O pair $[\boldsymbol{U}(0\,...\,k_h)]$ and $[\boldsymbol{Y}(0\,...\,k_h)]$ of the quantised system. The observation problem is solved by looking for qualitative states $[\boldsymbol{x}(k_h)]$ for which the given I/O pair may occur. In general, the solution will not be unique but represents a set $\mathcal{Z}(k_h \mid k_h)$ of such qualitative states. The diagnostic problem is solved by searching for all faults $f(k_h)$ for which the given I/O pair may occur. It yields the set $\mathcal{F}(k_h)$ of fault candidates.

Note that in both problems, the distinction between input and output is of no importance. Both sequences included in the I/O pair together provide the information about the current movement of the quantised system used when solving the observation or diagnostic task. It has only to be known which I/O pairs may occur and which I/O pairs may not occur. It is just the information provided by the behaviour $\mathcal{B}_{\text{qual}}$ of the quantised system. Any model of the quantised system used when solving the observation or diagnostic task should provide this information.

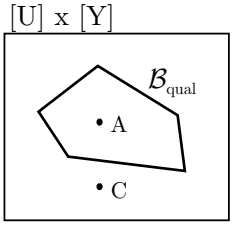


Fig. 9.11. Behaviour $\mathcal{B}_{\text{qual}}$

The behaviour $\mathcal{B}_{\text{qual}}(k_h)$ for given time horizont k_h has a nice graphical interpretation, because it is a subset of the Cartesian product of the sets of sequences $[\boldsymbol{U}(0\,...\,k_h)]$ and $[\boldsymbol{Y}(0\,...\,k_h)]$ (cf. Eq. (9.16) and Fig. 9.11). Qualitative I/O pairs $([\boldsymbol{U}(0\,...\,k_h)], [\boldsymbol{Y}(0\,...\,k_h)])$ for which the relation

$$([\boldsymbol{U}(0\,...\,k_h)], [\boldsymbol{Y}(0\,...\,k_h)]) \in \mathcal{B}_{\text{qual}}(k_h)$$

is valid are called *consistent* with the quantised system. For example, the I/O pair symbolised by the point A is consistent with the quantised system whereas the I/O pair C is inconsistent.

State observation of quantised systems. From the behavioural viewpoint, the observation problem consists in determining those qualitative state sequences

$$[\boldsymbol{X}(0\,...\,k_h)] = ([\boldsymbol{x}(0)], [\boldsymbol{x}(1)],\ ...,\ [\boldsymbol{x}(k_h)])$$

that may occur for the measured input sequence $[\boldsymbol{U}(0\,...\,k_h)]$ and may produce the measured output sequence $[\boldsymbol{Y}(0\,...\,k_h)]$. Then the I/O pair is called consistent with

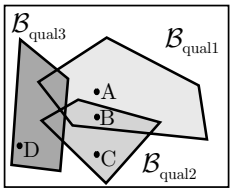


Fig. 9.12. Consistency-based observation and diagnosis

the behaviour that occurs for the initial state $\zeta_0 = [\boldsymbol{x}(0)]$ and ζ_0 is a possible qualitative initial state and $[\boldsymbol{x}(k_h)]$ a possible qualitative state at time k_h.

In more detail, assume that, as in Fig. 9.12, three different behaviours are given, which represent the quantised system for three different qualitative initial states $\mathcal{X}_0 = \mathcal{Q}_x(\zeta_{01})$, $\mathcal{X}_0 = \mathcal{Q}_x(\zeta_{02})$, and $\mathcal{X}_0 = \mathcal{Q}_x(\zeta_{03})$. If the measured I/O pair corresponds to point A, it is consistent with $\mathcal{B}_{\text{qual1}}$, which implies that the qualitative initial state is found to be ζ_{01}. If C or D are measured, the qualitative initial state ζ_{02} or ζ_{03} are uniquely determined. The point B illustrates that the observation problem may not have a unique solution. The I/O pair represented by B is consistent with two behaviours and, hence, the quantised system may have one of the two qualitative initial states ζ_{01} and ζ_{02}.

Fault diagnosis of quantised systems. The same way of solution is used for the diagnostic problem with the only difference that the quantised system is now subject to the fault sequence $\boldsymbol{E}$. Then, the behaviours $\mathcal{B}_{\text{qual1}}$, $\mathcal{B}_{\text{qual2}}$ and $\mathcal{B}_{\text{qual3}}$ used in Fig. 9.12 show the sets of triples

$$([\boldsymbol{U}(0 \ldots k_h)], [\boldsymbol{E}(0 \ldots k_h)], [\boldsymbol{Y}(0 \ldots k_h)])$$

that are consistent with the quantised system for three different qualitative initial states. In the graphical representation, the behaviour is now a subset of the set of such triples.

The measured I/O pair $([\boldsymbol{U}], [\boldsymbol{Y}])$ includes the first and the third element of the triple $([\boldsymbol{U}], [\boldsymbol{E}], [\boldsymbol{Y}])$ describing the behaviour of the faulty system. To solve the diagnostic problem it has to be checked whether the measured pair is consistent with some triple in the sense that the measurements are identical with the first and the third element of the triple. Then a possible fault sequence is given by the second element $[\boldsymbol{E}]$ of this triple for the initial state $\zeta(0)$ to which the behaviour belongs. That is, the diagnostic task is solved by testing the consistency of the measured I/O pair with the behaviour of the quantised system. This illustrates *consistency-based diagnosis* of quantised systems.

Way of solution for both problems. The development of a solution to the observation and the diagnostic problems consists of two major steps. First, a suitable representation of the quantised system has to be found. Section 9.4 will show that a stochastic automaton is such a suitable representation and that the automaton can

be determined from the description of the quantised system by means of Eqs. (9.3), (9.4) together with the quantisers. Second, methods that use this model and the observed input and output sequences must be elaborated in order to solve the state observation or the diagnostic problems. As the model used is a stochastic automaton, the methods developed in Chapter 8 can be applied for these purposes. As the results obtained for the stochastic automaton should be used for the quantised system, the automaton has to satisfy some completeness requirements, which will be developed in the sequel.

Requirement on the model. The behavioural view on the supervision problems to be solved shows that the model of the quantised system has to provide the behaviour $\mathcal{B}_{\text{qual}}(k_h)$ of the quantised system for the relevant time horizon k_h. Therefore, any model that describes the qualitative behaviour $\mathcal{B}_{\text{qual}}$ can be used. The model of the hybrid system given by Eqs. (2.1) – (2.3) together with the description of the quantisers given in Table 9.1 provides such a representation of the qualitative behaviour $\mathcal{B}_{\text{qual}}$.

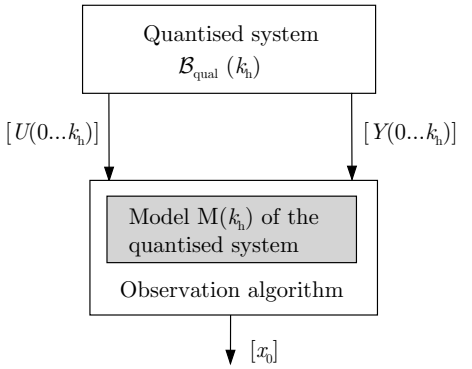


Fig. 9.13. A behavioural view on the observation problem

However, this hybrid representation of the quantised system leads to very complex observation and diagnostic algorithms because it combines differential equations with inequalities or logical formulae that describe the quantisers. Therefore, a more compact model will be introduced in Section 9.4 which has the form of a stochastic automaton whose state, input and output correspond directly with the quantised state, quantised input and quantised output of the given system. Instead of the behaviour $\mathcal{B}_{\text{qual}}$, the supervision problems will be solved by means of the behaviour $\mathcal{M}$ of the stochastic automaton, which likewise depends on the qualitative initial state and the fault occurring in the quantised system (cf. Fig. 9.13). This model directly refers to the qualitative versions of the input, state and output and is called *qualitative model*.

As both the observation and the diagnostic methods are based on a consistency check for a given I/O pair and the model, the model has to represent *all* I/O pairs

that may occur for the given quantised system, i.e. it has to be *complete* according to the following definition.

Definition 9.1 (Completeness)
A model with the behaviour $\mathcal{M}$ that satisfies the relation

$$\mathcal{M}(k_h) \supseteq \mathcal{B}_{\text{qual}}(k_h) \tag{9.21}$$

for all k_h is called complete*.*

Complete models include all I/O pairs for a given time horizon k_h that are consistent with the quantised system.

From Eq. 9.21 it follows that there may exist pairs

$$\begin{aligned}(V(0\,...\,k_h), W(0\,...\,k_h)) \quad &\in \quad \mathcal{M}(k_h)\\ (V(0\,...\,k_h), W(0\,...\,k_h)) \quad &\notin \quad \mathcal{B}_{\text{qual}}(k_h)\end{aligned}$$

which are consistent with the model but not with the quantised system. These pairs are called *spurious*. Their existence is a typical phenomenon encountered in qualitative modelling. The reason for the existence of spurious solutions is given by the fact that the qualitative model should be less complex than the precise model. Hence, it has to ignore some information about the properties of the quantised system. In particular, the qualitative model has the Markov property to provide a recursive representation of the behaviour $\mathcal{M}$ whereas the quantised system does not possess the Markov property.

The importance of the completeness of the model used for state observation and fault diagnosis is given by the following corollary:

> A model is suitable for solving the state observation problem or the diagnostic problem if and only if it is complete.

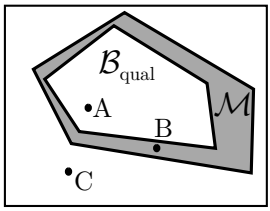


Fig. 9.14. Diagnosis with complete models

For a complete model with behaviour $\mathcal{M}$ an I/O pair like C in Fig. 9.14, which is not consistent with $\mathcal{M}$, is also not consistent with $\mathcal{B}_{\text{qual}}$:

$$([\boldsymbol{U}], [\boldsymbol{Y}]) \notin \mathcal{M}(k_h) \;\Rightarrow\; ([\boldsymbol{U}], [\boldsymbol{Y}]) \notin \mathcal{B}_{\text{qual}}(k_h). \tag{9.22}$$

This has a direct consequence for the observation problem. Assume that $\mathcal{B}_{\text{qual}}$ is the behaviour for $\mathcal{X}_0 = \mathcal{Q}_x(\zeta_0)$ and that the measured I/O pair corresponds to the point

C. Then the inconsistency of the I/O pair C with $\mathcal{M}$ implies the inconsistency of C with $\mathcal{B}_{\text{qual}}$ and it is known that the initial state cannot have been in $\mathcal{X}_0$: $\boldsymbol{x}_0 \notin \mathcal{X}_0$. On the other hand, an I/O pair like A, which is consistent with $\mathcal{B}_{\text{qual}}$ is also consistent with $\mathcal{M}$ and again the result obtained for $\mathcal{M}$ corresponds to the result that would be obtained for $\mathcal{B}_{\text{qual}}$.

Different results are obtained for I/O pairs that lie within $\mathcal{M}$ but not in $\mathcal{B}_{\text{qual}}$ like the pair B (spurious I/O pairs). Obviously, B is consistent with $\mathcal{M}$ but inconsistent with the quantised system. As a consequence, the sets of qualitative initial states or fault candidates that are obtained by a supervision algorithm that uses the model $\mathcal{M}$ are supersets of the solution sets that would be obtained if $\mathcal{B}_{\text{qual}}$ were used as representation of the quantised system. The fact that a superset of the solution is obtained rather than the precise solution, is the "price" for using the simple model $\mathcal{M}$ rather than the more complex representation of the quantised system by the behaviour $\mathcal{B}_{\text{qual}}$. The more spurious solutions exist, the more qualitative initial states or fault candidates occur in the solution set that should not occur there. Hence, the precision of the solution decreases. It is, therefore, the aim of modelling to find a model that satisfies the completeness requirement (9.21) with the smallest possible set of spurious solutions.

9.4 Discrete-event models of quantised systems

9.4.1 Modelling problem

From the point of view of the diagnostic algorithm that only has access to the qualitative I/O sequences, the input and output of the quantised system switch from one discrete value to another when time proceeds. Hence, the quantised system behaves like a discrete-event system. In more detail, Section 9.2.4 has shown that the quantised system is non-deterministic and can be considered as a stochastic process. The model used for diagnostic purposes has to describe this stochastic process.

> As the qualitative behaviour is non-deterministic, the model has to be non-deterministic.

This section shows how a model can be obtained that satisfies the completeness requirement (9.21).

To state the modelling problem more formally, denote the model input by $\nu(k)$, the model output by $\omega(k)$ and the behaviour of the model by $\mathcal{M}$. The solution to the modelling problem will be explained for the faultless system (9.3), (9.4) and later extended to faulty systems. For given initial state $\zeta(0)$ and given input sequence

$$\mathcal{V}(0 \ldots k_h) = (\nu(0), \nu(1), \nu(2), \ldots, \nu(k_h))$$

the model generates the output sequence

$$\Omega(0 \ldots k_h) = (\omega(0), \omega(1), \omega(2), \ldots, \omega(k_h)).$$

The model behaviour $\mathcal{M}(k_h) = \{(\mathcal{V}(0 \ldots k_h), \Omega(0 \ldots k_h))\}$ includes all I/O pairs with time horizon k_h that the model may produce. Since the model should be a representation of the quantised system, its input and output can assume the same values as the qualitative input and qualitative output signals of the quantised system: $\nu(k) \in \mathcal{N}_u$, $\omega(k) \in \mathcal{N}_y$.

Moreover, the model state ζ is interpreted as the qualitative state $[\boldsymbol{x}]$ of the system and, hence, $\zeta \in \mathcal{N}_x$ holds. This choice of the model state is very important because it makes it possible to associate with each model state a qualitative system state. In particular, for the initial state the relation

$$\zeta(0) = [\boldsymbol{x}(0)] \tag{9.23}$$

is valid.

As the considerations of Section 9.3 have shown, the completeness of the model is a fundamental requirement. The following sections show how stochastic automata can be found that provide a complete description of the quantised system.

9.4.2 Representation of autonomous quantised systems by stochastic automata

This and the next sections investigate how a quantised system can be described by a stochastic automaton (8.12)

$$\mathcal{S} = (\mathcal{N}_x, \mathcal{N}_u, \mathcal{N}_y, L, \mathrm{Prob}(z(0))) \tag{9.24}$$

whose state, input or output sets are identical to the sets of qualitative states, qualitative input values or qualitative output values, respectively. The modelling problem is to find the behavioural relation L such that the automaton $\mathcal{S}$ is a complete model of the quantised system. Then, the automaton is also called an *abstraction* of the system (9.3), (9.4).

Qualitative modelling of autonomous quantised systems. First, an autonomous quantised system ($\boldsymbol{u} = 0$) is considered whose qualitative state can be measured ($[\boldsymbol{y}(k)] = [\boldsymbol{x}(k)]$). The continuous-variable part of this system is given by

$$\boldsymbol{x}(k+1) = \boldsymbol{g}(\boldsymbol{x}(k)), \qquad \boldsymbol{x}(0) \in \mathcal{X}_0 \tag{9.25}$$

$$\boldsymbol{y}(k) = \boldsymbol{x}(k). \tag{9.26}$$

The quantised system should be described by the stochastic automaton

$$\mathcal{S}_a = (\mathcal{N}_x, G, \mathrm{Prob}(z(0)))$$

where G denotes the state transition probability (cf. Eq. (8.18) for $\omega = 0$).

As the set of automaton states coincides with the set $\mathcal{N}_x$ of qualitative states of the system 9.25, in the graphical representation each partition of the state space is associated with an automaton state. This is illustrated by Fig. 9.15 for the quantised tank system with 9 state partitions.

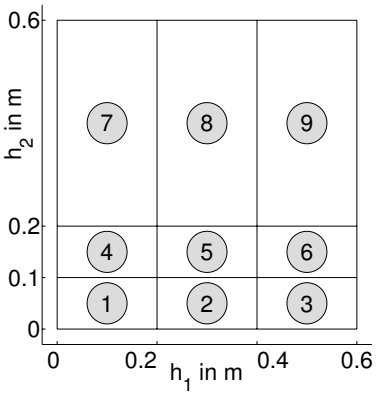


Fig. 9.15. Definition of the automata states for the quantised tank system

Completeness of the qualitative model. In order to illustrate the meaning of the completeness of the obtained model, use the model for a prediction of the qualitative behaviour of the quantised system for the time horizon $[0, k_h]$ by determining the state trajectory of the automaton. Assume that the qualitative initial state $[\boldsymbol{x}(0)]$ of the quantised system has been measured. Then the initial probability distribution of the automaton is given by

$$\mathrm{Prob}(z(0) = z) \begin{cases} > 0 & \text{if} \quad z = [\boldsymbol{x}_0] \ \text{ for some } \ \boldsymbol{x}_0 \in \mathcal{X}_0 \\ = 0 & \text{else.} \end{cases}$$

The future state trajectory can be determined by Eq. (8.24), which for the autonomous automaton reads as

$$\begin{aligned} \mathrm{Prob}(Z(0 \ldots k_h)) = G(z(k_h) \,|\, z(k_h - 1)) \cdot \ G(z(k_h - 1) \,|\, z(k_h - 2)) \\ \cdot \, G(z(1) \,|\, z(0)) \cdot \mathrm{Prob}(z(0)). \end{aligned} \tag{9.27}$$

The behaviour of the automaton includes all state trajectories that occur with non-vanishing probability:

$$\mathcal{M}(k_h) \quad = \quad \{Z(0 \ldots k_h) \ : \ \mathrm{Prob}(Z(0 \ldots k_h)) > 0\} \tag{9.28}$$

$$\begin{aligned} = \quad & \{(z(0), z(1), ..., z(k_h)) \ : \\ & \mathrm{Prob}(z(0)) > 0 \\ & G(z(k+1) \mid z(k)) > 0 \text{ for } k = 0, 1, ...k_h - 1\} \,. \end{aligned} \tag{9.29}$$

Due to the one-to-one relation between the qualitative states of the quantised system and the automaton states, the state trajectories of the automaton can be interpreted directly as qualitative state trajectories of the quantised system:

$$Z(0 \ldots k_h) = [\boldsymbol{X}(0 \ldots k_h)] \,. \tag{9.30}$$

The completeness requirement of the model claims that if the qualitative state trajectory

$$[\boldsymbol{X}(0\,...\,k_h)] = ([\boldsymbol{x}(0)]=z(0),\ [\boldsymbol{x}(1)]=z(1),\ ...,[\boldsymbol{x}(k_h)]=z(k_h))$$

can be generated by the quantised system, the trajectory

$$Z(0\,...\,k_h) = (z(0), z(1),\ ...,\ z(k_h))$$

has to belong to $\mathcal{M}(k_h)$.

Unfortunately, Eq. (9.30) is not valid for all $Z(0\,...\,k_h)$. The completeness relation (9.21) implies that the quantised system may not generate all qualitative state trajectory that belong to the set $\mathcal{M}(k_h)$, but there may be some state trajectories Z that the quantised system cannot follow. These are the spurious state sequences of the model.

Abstraction of the stochastic automaton. The initial state probability distribution $\mathrm{Prob}(z(0))$ and the state transition relation G of the automaton have to be chosen so that the model is complete. The question how to find such an automaton for a given quantised system is answered in the following lemma:

Lemma 9.1 (Complete model of the autonomous quantised system)
A stochastic automaton $\mathcal{S}_a = (\mathcal{N}_z, G, \mathrm{Prob}(z(0)))$ is a complete model of the autonomous quantised system if and only if the following conditions are satisfied:

$$G(z' \mid z) \quad > \quad 0 \iff \mathrm{Prob}([\boldsymbol{x}(1)]=z' \mid [\boldsymbol{x}(0)]=z) > 0 \tag{9.31}$$

$$\mathrm{Prob}(z(0)=z) \quad > \quad 0 \ \text{ for all } \ z=[\boldsymbol{x}_0],\ \ \boldsymbol{x}_0 \in \boldsymbol{\mathcal{X}}_0. \tag{9.32}$$

A stochastic automaton that satisfies these requirements is called an *abstraction* of the autonomous quantised system.

Equation (9.32) ensures that the automaton states, which correspond to possible qualitative initial states, have a non-vanishing probability. If $[\boldsymbol{x}(0)]$ is known, $\mathrm{Prob}(z(0))$ is chosen according to

$$\mathrm{Prob}(z(0)) = \begin{cases} 1 & \text{for } \ z(0)=[\boldsymbol{x}(0)] \\ 0 & \text{otherwise.} \end{cases} \tag{9.33}$$

If $[\boldsymbol{x}(0)]$ is not known, the condition (9.32) can be satisfied by associating a positive probability with all possible qualitative initial states. That is, if a set $\mathcal{Z}(0) \subseteq \mathcal{N}_x$ is available which is known to include the true value of $[\boldsymbol{x}(0)]$, the relation

$$\mathrm{Prob}(z(0)=z) > 0 \text{ for all } z \in \mathcal{Z}(0)$$

has to be satisfied.

In order to determine the function $G(z' \mid z)$, the value of

$$\mathrm{Prob}([\boldsymbol{x}(1)]=z' \mid [\boldsymbol{x}(0)]=z)$$

is determined for all possible combinations of z and z' by means of the state-space model (9.25) of the continuous-variable system and the definitions of the quantiser. To do this, assume that the initial state $\boldsymbol{x}(0)$ is uniformly distributed over the sets

$\mathcal{Q}_x(z)$ and determine the probability that $[\boldsymbol{x}(1)] = z'$ holds for given z and z'. For linear systems $\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k)$ with rectangular partitions this can be done by mapping the corner points of the set $\mathcal{Q}_x(z)$ by applying the matrix $\boldsymbol{A}$ and by determining the intersection of the resulting set with $\mathcal{Q}_x(z')$. For nonlinear systems the conditional probability (9.31) can be approximately determined by mapping a grid of initial states and "counting" those points $\boldsymbol{x}(1)$ that fall into the set $\mathcal{Q}_x(z')$.

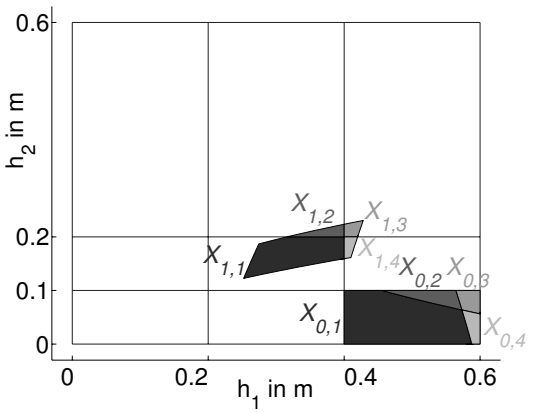


Fig. 9.16. Abstraction of a stochastic automaton describing the autonomous tank system

Example 9.4 *Abstraction of the autonomous tank system*

Consider the tank system for the same configuration as in Fig. 9.10. Figure 9.16 shows the map of the set $X_0 = \mathcal{Q}_x(3)$ of initial states towards the set $X_1 = \boldsymbol{g}(X_0)$ which includes all successor states $\boldsymbol{x}(1)$. In order to determine the transition probability, the set $\mathcal{X}_1$ has to be decomposed into the sets $X_{1,1}$, $X_{1,2}$, $X_{1,3}$ and $X_{1,4}$ where the relations

$$X_{1,2} = X_1 \cap \mathcal{Q}_x(8), \qquad X_{1,3} = X_1 \cap \mathcal{Q}_x(9)$$
$$X_{1,1} = X_1 \cap \mathcal{Q}_x(5), \qquad X_{1,4} = X_1 \cap \mathcal{Q}_x(6)$$

hold. Then it is determined which initial states $\boldsymbol{x}(0)$ lead to a state $\boldsymbol{x}(1) \in X_{1,1}$. The set of such states is denoted by $X_{0,1}$. The same is done for $\boldsymbol{x}(0)$ that yield successor states in $X_{1,2}$, $X_{1,3}$ or $X_{1,4}$, which are summarised into the sets $X_{0,2}$, $X_{0,3}$ or $X_{0,4}$, respectively. That is, the trajectories are followed in reverse direction leading to the decomposition of the set $\mathcal{Q}_x(3)$ into the disjoint sets $X_{0,1}$ to $X_{0,4}$. Then the transition probability can be determined as the quotient of the areas of these sets, which are given by the measure $\lambda(\cdot)$ of these sets:

$$\begin{aligned}
\mathrm{Prob}([\boldsymbol{x}(1)]{=}5 \mid [\boldsymbol{x}(0)]{=}3) &= \frac{\lambda(X_{0,1})}{\lambda(X_0)} \\
\mathrm{Prob}([\boldsymbol{x}(1)]{=}6 \mid [\boldsymbol{x}(0)]{=}3) &= \frac{\lambda(X_{0,4})}{\lambda(X_0)} \\
\mathrm{Prob}([\boldsymbol{x}(1)]{=}8 \mid [\boldsymbol{x}(0)]{=}3) &= \frac{\lambda(X_{0,2})}{\lambda(X_0)}
\end{aligned}$$

$$\mathrm{Prob}([\boldsymbol{x}(1)]=9 \mid [\boldsymbol{x}(0)]=3) \quad = \quad \frac{\lambda(X_{0,3})}{\lambda(X_0)}$$

More precisely, these areas are the Lebesgue measures of the sets.

These steps have to be performed for all partitions $\mathcal{Q}_x(z)$, $z \in \mathcal{N}_x$. Note that this investigation includes only the behaviour of the continuous-variable system for one time step (i.e. from $k = 0$ towards $k = 1$), although the automaton can be used later to generate state trajectories $\mathcal{Z}(0 \ldots k_h)$ of arbitrary length k_h. □

Reasonable choice of the state transition probability. Equation (9.31) claims that whenever the quantised system can perform a state transition from the qualitative state z towards the qualitative state z' then the automaton has to be able to perform the same state transition. As the automaton should not be used only to generate a complete behaviour but also to predict the probability with which the I/O pairs occur, the transition probability is chosen as follows:

$$G(z' \mid z) = \mathrm{Prob}([\boldsymbol{x}(1)]=z' \mid [\boldsymbol{x}(0)]=z). \tag{9.34}$$

Then, Eq. (9.27) gives an estimate of the probability with which a given state trajectory Z is generated if the quantised systems is at $k = 0$ in a given qualitative initial state $[\boldsymbol{x}(0)] = z_0$. This probability is only an *estimate* and not the true probability, because the relation (9.21) is, in general, not satisfied with the equality sign and the spurious trajectories are predicted with non-vanishing probability. Hence, some non-spurious trajectories must be associated with imprecise probability measures. [1]

Often, instead of the probability of the whole state trajectory Z the probability of a certain qualitative state $[\boldsymbol{x}(k)]$ is to be predicted. Equation (8.32), which simplifies for the autonomous case to the recursive relation

$$\mathrm{Prob}(z(k+1)) = \sum_{z(k) \in \mathcal{N}_z} G(z(k+1) \mid z(k)) \cdot \mathrm{Prob}(z(k)),$$

can be used to determine $\mathrm{Prob}(z(k+1))$ for the given time horizon $k = 0, \ldots, k_h - 1$. The result gives an estimate of the probability with which the qualitative state $[\boldsymbol{x}(k)] = z(k)$ is assumed on any state trajectory that the quantised system may generate for the given qualitative initial state. The completeness property of the qualitative model implies that only qualitative states of the set

$$\mathcal{Z}(k) = \{[\boldsymbol{x}(k)] \,:\, \mathrm{Prob}(z(k)=[\boldsymbol{x}(k)]) > 0\}$$

can be assumed or, vice versa, that the quantised system is known not to assume any qualitative state $[\boldsymbol{x}(k)]$ for which the relation

$$\mathrm{Prob}(z=[\boldsymbol{x}(k)]) = 0$$

holds.

[1] Even if, under very restrictive conditions, the relation (9.21) were satisfied with the equality sign, the stochastic automaton would yield merely an estimate rather than the true probability distributions.

Example 9.5 *Prediction of the qualitative state of the tank system*

Figure 9.17 shows the qualitative state trajectory of the tank system that has been generated by means of the qualitative model. The grey rectangles show with which probability the given qualitative state is assumed at the corresponding time instants. White rectangles say that it is impossible that the quantised tank system assumes the corresponding qualitative state for the time instant k considered. The completeness of the model implies that the qualitative states assumed by the tank system during the trajectories depicted in Fig. 9.7 are associated with a non-vanishing probability (grey box) in Fig. 9.17. The probabilities obtained by means of the qualitative model differ from the true values depicted in Fig. 9.9 because the stochastic automaton is only an approximate model of the quantised system. □

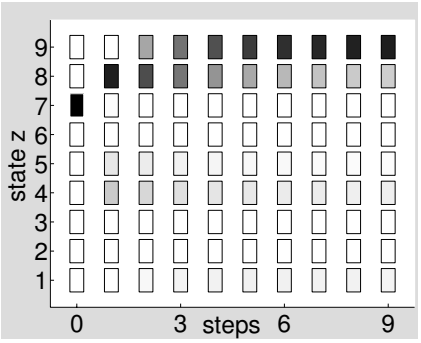


Fig. 9.17. Qualitative state trajectory of the tank system predicted by means of the qualitative model

Abstraction algorithms. The determination of the right-hand side of Eq. (9.34) is a numerical problem of its own. The main difficulty lies in the determination of the set $\mathcal{X}_1$ of successor states $\boldsymbol{x}(1)$ for a given set $\mathcal{X}_0 = \mathcal{Q}_x(z)$ of initial state $\boldsymbol{x}(0)$. Several methods have been elaborated.

The simplest method uses a grid of $\mathcal{N}$ points distributed uniformly over the set $\mathcal{Q}_x(z)$ and determines the set of successor states $\boldsymbol{x}(1)$. Then the number $\mathcal{N}'$ of states with the same qualitative value $[\boldsymbol{x}(1)] = z'$ is obtained and the probability approximated by the relative frequency as follows:

$$\mathrm{Prob}\left([\boldsymbol{x}(1)] = z' \mid [\boldsymbol{x}(0)] = z\right) \approx \frac{\mathcal{N}'}{\mathcal{N}}.$$

The larger the number $\mathcal{N}$ the better is this approximation. However, even for a low dynamical order $\nu = \dim(\boldsymbol{x})$ a very large number of grid points have to be used to get a reasonable approximation.

The main problem of applying this method is the fact that the completeness of the model cannot be ensured. The reason for this is given by the fact that even for a large number of grid points not all state transitions $z \mapsto z'$ are found. Hence, the algorithm yields

$$\mathrm{Prob}\left([\boldsymbol{x}(1)] = z' \mid [\boldsymbol{x}(0)] = z\right) = 0$$

even if the state transition $z \mapsto z'$ is possible for the quantised system.

A complete model can be obtained by an abstraction method that uses a Lipschitz condition for the function $\boldsymbol{g}$. Accordingly, a constant L is determined such that

$$||\boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{x}')|| \leq L\,||\boldsymbol{x} - \boldsymbol{x}'||$$

holds for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{Q}_x(z)$. The main idea of this abstraction method is to map a subset $\mathcal{Q} \subset \mathcal{Q}_x(z)$ which results in the set

$$\mathcal{Q}' = \{\boldsymbol{x}(1) \mid \boldsymbol{x}(1) = \boldsymbol{g}(\boldsymbol{x}(0)), \boldsymbol{x}(0) \in \mathcal{Q}\}$$

of successor states. Instead of determining $\mathcal{Q}'$, a superset $\bar{\mathcal{Q}}'$ can be found by mapping the center point $\boldsymbol{c}$ of $\mathcal{Q}$ into the point $\boldsymbol{c}' = \boldsymbol{g}(\boldsymbol{c})$ and by using the Lipschitz constant L for determining an upper bound of the distance that any point $\boldsymbol{x}(1)$ may have from the center $\boldsymbol{c}'$. If $\mathcal{Q}_x(z)$ is partitioned into sufficiently small sets $\mathcal{Q}$, the union of the sets $\bar{\mathcal{Q}}'$ found in this way is a reasonable approximation of $\mathcal{X}_1$. The advantage of this method with respect to the point-mapping method described above results from the fact that a superset of $\mathcal{X}_1$ is found and, hence, no qualitative state transition is missed. The model obtained in this way is complete.

If the function $\boldsymbol{g}$ has specific properties, simpler algorithms can be used. For example, if $\boldsymbol{g}$ is linear

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x},$$

and the quantiser includes a rectangular partitioning of the state space, it is sufficient to map the corner points of $\mathcal{Q}_x(z)$ by multiplying them by the matrix $\boldsymbol{A}$. The points obtained are the corner points of $\mathcal{X}_1$ (cf. Fig. 9.16).

9.4.3 Extensions to quantised systems with input and output

The abstraction method explained in the preceding section for autonomous systems should be extended now for systems with input and output. A stochastic automaton

$$\mathcal{S} = (\mathcal{N}_x, \mathcal{N}_u, \mathcal{N}_y, L, \mathrm{Prob}(z(0)))$$

is used whose state, input and output sets are identical to the sets of qualitative states, qualitative input and qualitative output values of the quantised system. The behavioural relation L and the initial state probability have to be chosen according to the following theorem.

Theorem 9.1 (Complete model of the quantised system)
A stochastic automaton is a complete model of the quantised system if and only if the following conditions are satisfied:

$$L(z', w \mid z, v) > 0$$
$$\Longleftrightarrow \mathrm{Prob}([\boldsymbol{x}(1)] = z', [\boldsymbol{y}(0)] = w \mid [\boldsymbol{x}(0)] = z, [\boldsymbol{u}(0)] = v) > 0 \quad (9.35)$$
$$\mathrm{Prob}(z(0) = z) > 0 \ \text{ for all } \ z = [\boldsymbol{x}_0] \ \ \boldsymbol{x}_0 \in \mathcal{X}_0 \quad (9.36)$$

Like in the autonomous case (cf. Eq. (9.34)), L is best chosen according to

$$L(z', w \mid z, v) = \mathrm{Prob}([\boldsymbol{x}(1)] = z', [\boldsymbol{y}(0)] = w \mid [\boldsymbol{x}(0)] = z, [\boldsymbol{u}(0)] = v). \quad (9.37)$$

Note that the right-hand side of Eq. (9.35) can be determined by means of the state-space model (9.3), (9.4) of the continuous-variable system and the definitions of the quantisers. The initial state $\boldsymbol{x}(0)$ and the input $\boldsymbol{u}(0)$ are distributed over the sets $\mathcal{Q}_x(z)$ or $\mathcal{Q}_u(v)$ and the task is to determine the probability that $[\boldsymbol{x}(1)] = z'$ and $[\boldsymbol{y}(0)] = w$ holds for given z, v, z' and w. An approximation of L can be obtained by mapping a grid of initial states for selected input values and "counting" those points $\boldsymbol{x}(1)$ or $\boldsymbol{y}(0)$ that fall into the sets $\mathcal{Q}_x(z')$ or $\mathcal{Q}_y(w)$, respectively.

This abstraction step can be explained by again using Fig. 9.16. As now different output values belong to different values of L, the decomposition of the sets X_0 and X_1 into the sets $X_{0,1} \ldots X_{0,4}$ or $X_{1,1} \ldots X_{1,4}$, respectively, has to be refined. All these sets have to be further decomposed according to the output $[\boldsymbol{y}]$ that the system produces when it moves from the set $X_{0,i}$ towards the set $X_{1,i}$. The resulting partitions are "numbered" by the output w, which leads to the notations $X_{0,i}(w)$ or $X_{1,i}(w)$ and the relations

$$\begin{aligned} X_{0,i} &= \cup_{w \in \mathcal{N}_w} X_{0,i}(w) \\ X_{1,i} &= \cup_{w \in \mathcal{N}_w} X_{1,i}(w) \end{aligned}$$

holds. Furthermore, the state transitions have to be considered for all the possible input values v. Therefore, the "departure" sets $X_{0,i}$ and the "arrival" sets $X_{1,i}$ depend on the input v, which is symbolised by the additional argument: $X_{0,i}(v, w)$ and $X_{1,i}(v, w)$. Then the behavioural relation L can be determined for the state transition starting in the qualitative state 3 and ending in the qualitative states $5, 6, 8$ or 9 as follows:

$$\begin{aligned} L(w, 5 \,|\, 3, v) &= \mathrm{Prob}([\boldsymbol{y}(0)] = w, [\boldsymbol{x}(1)] = 5 \mid [\boldsymbol{x}(0)] = 3, [\boldsymbol{u}(0)] = v) \\ &= \frac{\lambda(X_{0,1}(v, w))}{\lambda(\mathcal{Q}_x(3))} \end{aligned}$$

$$\begin{aligned} L(w, 6 \,|\, 3, v) &= \mathrm{Prob}([\boldsymbol{y}(0)] = w, [\boldsymbol{x}(1)] = 6 \mid [\boldsymbol{x}(0)] = 3, [\boldsymbol{u}(0)] = v) \\ &= \frac{\lambda(X_{0,4}(v, w))}{\lambda(\mathcal{Q}_x(3))} \\ &\vdots \end{aligned}$$

9.4.4 Representation of faulty quantised systems

The abstraction method developed so far can be used to find a complete model for the faulty quantised system. The description of the stochastic automaton has to be extended so as to refer to the fault as an additional input (cf. Fig. 9.1):

$$\mathcal{S} = (\mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, \mathcal{N}_f, L, \mathrm{Prob}(z(0))). \tag{9.38}$$

The behavioural relation $L(z', w \mid z, v, f)$ has to satisfy the condition (9.35) for given fault f:

$$\begin{aligned} &L(z', w \mid z, v, f) \\ &\quad = \mathrm{Prob}([\boldsymbol{x}(1)] = z', [\boldsymbol{y}(0)] = w \mid [\boldsymbol{x}(0)] = z, [\boldsymbol{u}(0)] = v, [\boldsymbol{e}] = f). \end{aligned} \tag{9.39}$$

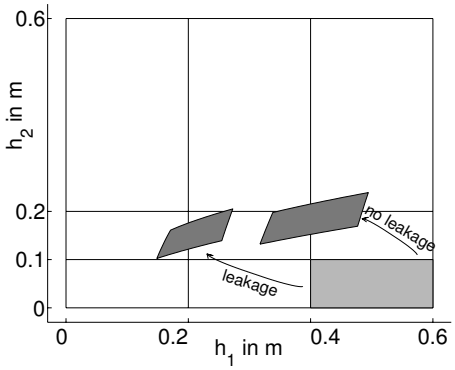


Fig. 9.18. Set of states of the tank system reached with and without fault

Example 9.6 *Model of the faulty tank system*

For the tank system, a stochastic automaton has been obtained by means of the abstraction method explained in this section. Figure 9.18 shows the set of states reached by the faultless and by the faulty system for $k = 1$ when starting in $\boldsymbol{x}(0) \in \mathcal{Q}_x(3)$ for the input signals $[V_{12}] = 2$ and $[u_P] = 2$. In the faultless case the region overlaps with the regions $5, 6, 8$ and 9 of the partitioned state space.

In this example, the quantised outflow $[q_M]$ of Tank 2 is considered as output. The automaton graph for the faultless case and for the above input is shown in Fig. 9.19. The vertices

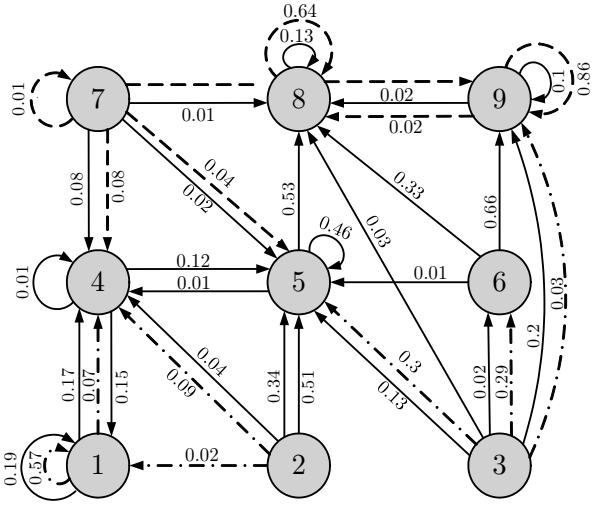


Fig. 9.19. Automaton graph of faultless tank system ($[c_L] = 1$) for the input $[V_{12}] = 2$ and $[u_P] = 2$

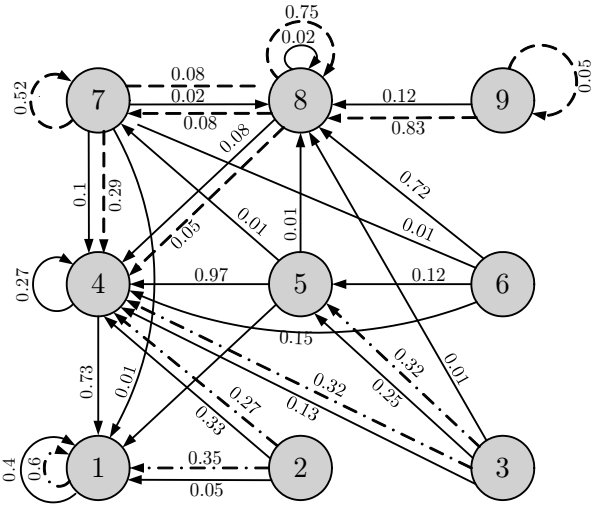


Fig. 9.20. Automaton graph of faulty tank system ($[c_L] = 2$) for the input $[V_{12}] = 2$ and $[u_P] = 2$

correspond to the quantised states shown in Fig. 9.15. The colour and thickness of the edges denote the different quantised output values, from thin black for $[q_M]=1$ to thick light grey for $[q_M]=3$. The edge labels refer to the probability of the transition.

For a leakage $[c_L]=2$ the qualitative behaviour of the tank system changes considerably, which can be seen from the difference between the automaton graphs for the faultless and the faulty systems in Figs. 9.19 and 9.20. □

The fault $f = [\boldsymbol{e}(k)]$ is used here as an additional (unknown) input to the quantised system. In general, some information about the frequency of the change of this signal is known and should be used during the diagnosis. Like for the stochastic automaton (cf. Fig. 8.9 on page 388) a fault model is used to represent this information (Fig. 9.21).

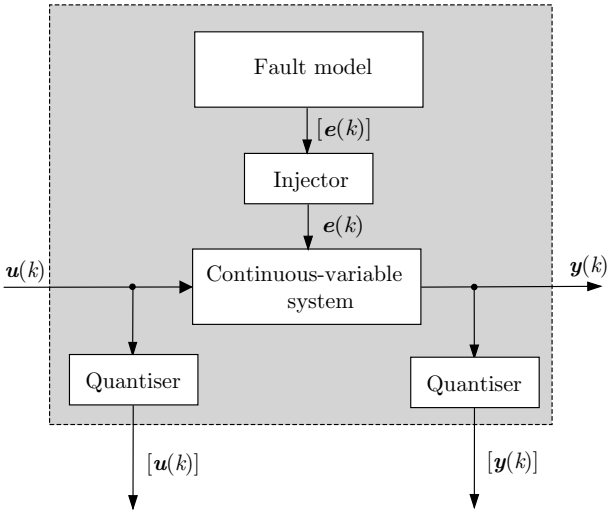


Fig. 9.21. Quantised system with fault model

The fault is assumed to be the output of the stochastic automaton

$$\mathcal{S}_f = (\mathcal{N}_f, G_f, \mathrm{Prob}(f(0))) \tag{9.40}$$

which is called the *fault model*. The transition relation G_f describes the fault-state transition probability

$$\begin{aligned} &G_f : \mathcal{N}_f \times \mathcal{N}_f \;\longrightarrow\; [0,1] \\ &G_f(f' \mid f) = \mathrm{Prob}([\boldsymbol{e}(1)]=f' \mid [\boldsymbol{e}(0)]=f)\;, \end{aligned} \tag{9.41}$$

which is the conditional probability that the fault changes from $[\boldsymbol{e}(0)]=f$ towards $[\boldsymbol{e}(1)]=f'$ within one time step. The a-priori probability distribution over the initial fault set is given by $\mathrm{Prob}(f(0))$.

The combination of the stochastic automaton, which represents the quantised system, with the fault model is depicted in Fig. 9.22. It is a stochastic automaton

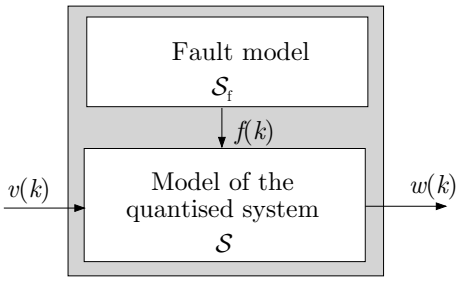


Fig. 9.22. Model of the quantised system combined with the fault model

$$\tilde{\mathcal{S}} = (\mathcal{N}_{\tilde{z}}, \mathcal{N}_v, \mathcal{N}_w, \tilde{L}, \mathrm{Prob}(\tilde{z}(0))) \tag{9.42}$$

whose state set is given by the Cartesian product

$$\mathcal{N}_{\tilde{z}} = \mathcal{N}_z \times \mathcal{N}_f \tag{9.43}$$

and whose behavioural relation $\tilde{L}$ can be obtained from L and G_f according to the relation

$$\tilde{L}(z', f', w \mid z, f, v) = L(z', w \mid z, f, v) \cdot G_f(f' \mid f) \tag{9.44}$$

with $z, z' \in \mathcal{N}_z$, $v \in \mathcal{N}_v$, $w \in \mathcal{N}_w$ and $f, f' \in \mathcal{N}_f$ (cf. Eqs. (8.42) – (8.44)). This model will be used later for solving diagnostic tasks.

9.5 State observation of quantised systems

9.5.1 Observation method

This section deals with the state observation problem for quantised systems given in Section 9.1.2. The task is to find the current qualitative state $[\boldsymbol{x}(k_h)]$ for the measured sequences of input values

$$[\boldsymbol{U}(0 \ldots k_h)] = ([\boldsymbol{u}(0)], \ldots, [\boldsymbol{u}(k_h)])$$

and output values

$$[\boldsymbol{Y}(0 \ldots k_h)] = ([\boldsymbol{y}(0)], \ldots, [\boldsymbol{y}(k_h)]).$$

The model $\mathcal{S}$ used for solving this problem is the stochastic automaton

$$\mathcal{S} = (\mathcal{N}_x, \mathcal{N}_u, \mathcal{N}_y, L, \mathrm{Prob}(z(0)))$$

introduced in Eq. (9.24). Therefore, the observation problem can be solved by directly applying the observation method developed in Section 8.3 where

$$\begin{array}{lll} V(0 \ldots k_h) & \text{is replaced by} & [\boldsymbol{U}(0 \ldots k_h)] \\ W(0 \ldots k_h) & & [\boldsymbol{Y}(0 \ldots k_h)]\,. \end{array} \tag{9.45}$$

The automaton state $z(k)$ describes the qualitative value $[\boldsymbol{x}(k)]$ of the system state $\boldsymbol{x}$. Equations (8.64) – (8.66) yield

$$\mathrm{Prob}([\boldsymbol{x}(k_h)] \mid k_h) = \frac{\sum\limits_{[\boldsymbol{x}(k_h+1)]} L(k_h) \cdot \mathrm{Prob}([\boldsymbol{x}(k_h)] \mid k_h-1)}{\sum\limits_{[\boldsymbol{x}(k_h)],\,[\boldsymbol{x}(k_h+1)]} L(k_h) \cdot \mathrm{Prob}([\boldsymbol{x}(k_h)] \mid k_h-1)} \qquad (k_h = 0, 1, ...) \tag{9.46}$$

with

$$\mathrm{Prob}([\boldsymbol{x}(k_h)] \mid k_h-1) = \frac{\sum\limits_{[\boldsymbol{x}(k_h-1)]} L(k_h-1) \cdot \mathrm{Prob}([\boldsymbol{x}(k_h!-!1)] \mid k_h-2)}{\sum\limits_{[\boldsymbol{x}(k_h)],\,[\boldsymbol{x}(k_h-1)]} L(k_h!-!1) \cdot \mathrm{Prob}([\boldsymbol{x}(k_h-1)] \mid k_h-2)} \qquad (k_h = 1, 2, ...) \tag{9.47}$$

$$\mathrm{Prob}([\boldsymbol{x}(0)] \mid -1) := \mathrm{Prob}([\boldsymbol{x}(0)]), \tag{9.48}$$

where the abbreviation

$$L(k_h) = L([\boldsymbol{x}(k_h+1)], [\boldsymbol{y}(k_h)] \mid [\boldsymbol{x}(k_h)], [\boldsymbol{u}(k_h)])$$

has been used. The set

$$\mathcal{Z}(k_h \mid k_h) = \{[\boldsymbol{x}(k_h)] \,:\, \mathrm{Prob}([\boldsymbol{x}(k_h)] \mid k_h) > 0\} \tag{9.49}$$

describes all current qualitative states $[\boldsymbol{x}(k_h)]$ that the observation method provides.

9.5.2 Discussion of the result

The recursion relations (9.46) – (9.48) have been obtained by using the relations (8.64) – (8.66) developed for the stochastic automaton, where the abbreviation (8.50)

$$\mathrm{Prob}(z(k_h) \,|\, k_h) := \mathrm{Prob}(z(k_h) \mid W(0 \,...\, k_h), V(0 \,...\, k_h))$$

had been used. Therefore, it is reasonable to expect that $\mathrm{Prob}([\boldsymbol{x}(k_h)] \mid k_h)$ which occurs in Eqs. (9.46) – (9.48) is likewise an abbreviation of the probability

$$\mathrm{Prob}([\boldsymbol{x}(k_h)] \,|\, [\boldsymbol{U}(0 \,...\, k_h)], [\boldsymbol{Y}(0 \,...\, k_h)])$$

with which the qualitative state $[\boldsymbol{x}]$ can be assumed by the quantised system provided that the quantised system has generated the I/O pair $([\boldsymbol{U}(0 \,...\, k_h)], [\boldsymbol{Y}(0 \,...\, k_h)])$. However, this conjecture is not true, because the stochastic automaton used to solve the observation problem is only a complete but not a precise model of the quantised system. Therefore, the result obtained by the observation method can only be an approximation of the probability to be found:

$$\mathrm{Prob}([\boldsymbol{x}(k_h)] \mid k_h) \approx \mathrm{Prob}([\boldsymbol{x}(k_h)] \,|\, [\boldsymbol{U}(0 \,...\, k_h)], [\boldsymbol{Y}(0 \,...\, k_h)])\,. \tag{9.50}$$

As a consequence, the set $\mathcal{Z}(k_h \mid k_h)$ obtained from Eq. (9.49) need not coincide with the solution set of the observation problem defined in Eq. (9.1):

$$\mathcal{X}(k_h \mid k_h) = \{[\boldsymbol{x}(k_h)] \,:\, \mathrm{Prob}([\boldsymbol{x}(k_h)] \mid [\boldsymbol{U}(0 \,...\, k_h)], [\boldsymbol{Y}(0 \,...\, k_h)]) > 0\}.$$

However, due to the completeness of the model the set $\mathcal{Z}$ is know to be a superset of $\mathcal{X}$:

$$\mathcal{Z}(k_h \mid k_h) \supseteq \mathcal{X}(k_h \mid k_h)\,. \tag{9.51}$$

Theorem 9.2 (Observation of quantised systems)
If the stochastic automaton $\mathcal{S}$ is a complete model of the quantised system, the qualitative state $[\boldsymbol{x}(k_h)]$ of the quantised system belongs to the set $\mathcal{Z}(k_h \mid k_h)$ described by Eq. (9.49), where $\mathrm{Prob}([\boldsymbol{x}(k_h)] \mid k_h)$ is given by Eqs. (9.46) – (9.48).

For the application, this theorem yields the following conclusions:

Corollary 9.1 (Observation result for the quantised system)

- *The quantised system is known to be in some qualitative state*

  $$[\boldsymbol{x}(k_h)] \in \mathcal{Z}(k_h \mid k_h)$$

  and not to be in any state $[\boldsymbol{x}(k_h)] \notin \mathcal{Z}(k_h \mid k_h)$.
- *$\mathrm{Prob}([\boldsymbol{x}(k_h)] \mid k_h)$ is an estimate of the probability with which the quantised system assumes the state $[\boldsymbol{x}(k_h)]$.*

A-priori knowledge about the initial state. An a-priori probability distribution $\mathrm{Prob}([\boldsymbol{x}(0)])$ has to be known to initialise the observation method. As for the stochastic automaton, it is most important to ensure that

$$\mathrm{Prob}([\boldsymbol{x}(0)]) > 0$$

holds for the true qualitative initial state $[\boldsymbol{x}(0)]$, which is unknown. If nothing is known about $[\boldsymbol{x}(0)]$, a good choice of the a-priori probability distribution is the uniform distribution over the set $\mathcal{N}_x$ of all qualitative states

$$\mathrm{Prob}([\boldsymbol{x}(0)]) = 1/(N+1) \text{ for all } [\boldsymbol{x}(0)] \in \mathcal{N}_x\,,$$

where $N+1$ is the number of qualitative states defined by the state quantiser.

Consistent I/O pairs. It has been discussed in detail in Section 8.3.3 that state observation problems can be solved for stochastic automata only if the measured

I/O pair is consistent with the automaton. The same holds true here for the quantised system. However, this does not pose any problems, because the model is set up to be a complete abstraction of the quantised system. Therefore, any I/O pair that the quantised system may generate is consistent with the stochastic automaton used in the observation method. Consequently, the denominators occurring in Eqs. (9.46) and (9.47) are positive in each recursion step. If they become zero, either the a-priori knowledge about the initial state was wrong, the measurements has been perturbed by some disturbances and, hence, became inconsistent with the model or, for some reason, the qualitative model is not complete. If the qualitative model has been obtained by the abstraction method given in Section 9.4 the latter situation can only occur if the continuous-variable state-space model (9.3), (9.4) is wrong.

9.5.3 Observation algorithm

In the observation algorithm the following abbreviations are used:

$$h([\boldsymbol{x}(k_h)]) = \sum_{[\boldsymbol{x}(k_h+1)]} L(k_h) \cdot \mathrm{Prob}([\boldsymbol{x}(k_h)] \mid [\boldsymbol{U}(0 \ldots k_h - 1)], [\boldsymbol{Y}(0 \ldots k_h - 1)])$$

$$p_k([\boldsymbol{x}(k_h)]) = \mathrm{Prob}([\boldsymbol{x}(k_h)] \mid [\boldsymbol{U}(0 \ldots k_h)], [\boldsymbol{Y}(0 \ldots k_h)])$$

$$p_r([\boldsymbol{x}(k_h+1)]) = \mathrm{Prob}([\boldsymbol{x}(k_h+1)] \mid [\boldsymbol{U}(0 \ldots k_h)], [\boldsymbol{Y}(0 \ldots k_h)])$$

Algorithm 9.1 *Observation algorithm for quantised systems*

Given: Complete model $\mathcal{S}$ of the quantised system.
A-priori initial state probability $\mathrm{Prob}([\boldsymbol{x}(0)])$.

Initialise: $p_r([\boldsymbol{x}]) = \mathrm{Prob}([\boldsymbol{x}(0)])$
$k_h = 0$.

Do:

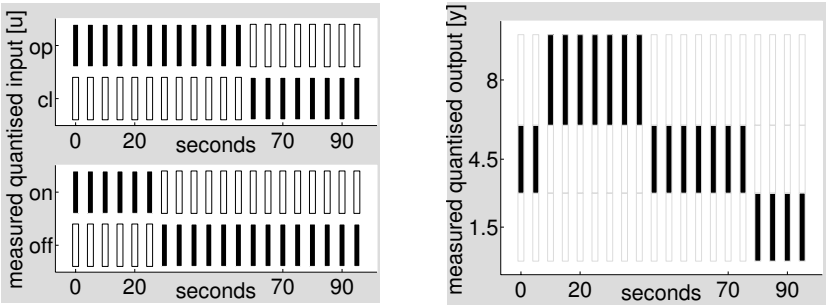
1. Measure $[\boldsymbol{u}(k_h)],\ [\boldsymbol{y}(k_h)]$.
2. For all $[\boldsymbol{x}] \in \mathcal{N}_x$ determine
$h([\boldsymbol{x}]) = \sum_{[\bar{\boldsymbol{x}}]} L([\bar{\boldsymbol{x}}], [\boldsymbol{y}(k_h)] \mid [\boldsymbol{x}], [\boldsymbol{u}(k_h)]) \cdot p_r([\boldsymbol{x}])$.
3. If $\sum_{[\boldsymbol{x}]} h([\boldsymbol{x}]) = 0$ holds, stop the algorithm (inconsistent I/O pair or wrong initial state distribution).
4. For all $[\boldsymbol{x}] \in \mathcal{N}_x$ determine $p_k([\boldsymbol{x}]) = \dfrac{h([\boldsymbol{x}])}{\sum\limits_{[\boldsymbol{x}]} h([\boldsymbol{x}])}$.
5. For all $[\boldsymbol{x}] \in \mathcal{N}_x$ determine
$$p_r([\boldsymbol{x}]) = \frac{\sum\limits_{[\bar{\boldsymbol{x}}]} L([\boldsymbol{x}], [\boldsymbol{y}(k_h)] \mid [\bar{\boldsymbol{x}}],\ [\boldsymbol{u}(k_h)])\, p_r([\bar{\boldsymbol{x}}])}{\sum\limits_{[\boldsymbol{x}]} h([\boldsymbol{x}])}.$$
6. Determine $\mathcal{Z}(k_h \mid k_h) = \{[\boldsymbol{x}] \,:\, p_k([\boldsymbol{x}]) \neq 0\}$.
7. $k_h := k_h + 1$
Continue with Step 1.

Result: $p_k([\boldsymbol{x}(k_h)]) \approx \mathrm{Prob}([\boldsymbol{x}(k_h)] \mid [\boldsymbol{U}(0 \ldots k_h)], [\boldsymbol{Y}(0 \ldots k_h)])$ and $\mathcal{Z}(k_h \mid k_h)$ for increasing time horizon k_h .

Example 9.7 *State observation of a tank system*

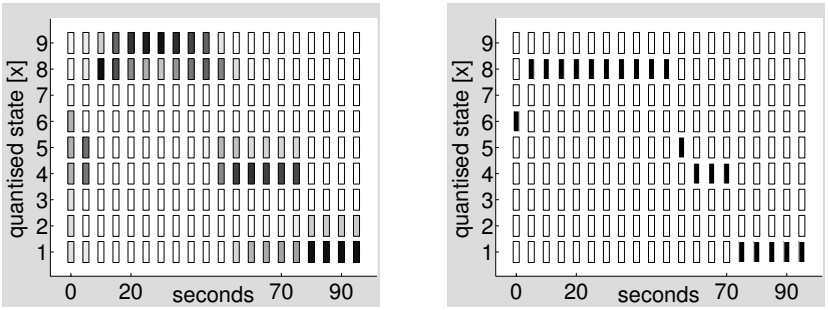
The observation algorithm is now applied to the two-tank system described in Section 2.1 in the quantised configuration given in Section 9.4. On the left-hand side of Fig. 9.23 the sequence $[\boldsymbol{U}(0 \ldots k_h)]$ of measured quantised input signals is shown. The upper sequence corresponds to the valve position $[V_{12}]$ (open or closed) and the lower to the quantised pump input $[u_P]$. The right-hand side of this figure shows the measured output sequence $[\boldsymbol{Y}(0 \ldots k_h)]$, where the output corresponds to the quantised outflow $[q_M]$ of Tank 2. The task is to determine from these sequences the quantised states $[\boldsymbol{x}(k_h)]$ at each time instant $k_h = 0, 1, \ldots$ for unknown initial state set $\mathcal{X}_0$.

On the left-hand side of Fig. 9.24 the observation result obtained by the above algorithm is shown. The grey boxes depict the probability distributions $\mathrm{Prob}([\boldsymbol{x}(k_h)] \mid k_h)$ for



(a) (b)

Fig. 9.23. Quantised input and output sequences used to solve the state observation problem



(a) (b)

Fig. 9.24. Observation result (left) and state sequence of the considered experiment (right)

$k_h = 0, 1, ...$ for the 9 different quantised states. For comparison, the state sequence of the experiment for which the output sequence was obtained is shown on the right-hand side of the figure. It can be seen, that the observation result is nonzero in the states of the "real" sequence at all times. Note that the state sequence obtained in this experiment is not the only possible one yielding the measurements of Fig. 9.23. However, the observation result is guaranteed to cover all possible sequences. □

9.6 Diagnosis of quantised systems

9.6.1 Diagnostic method

This section deals with the diagnosis of quantised system. To solve the problem given in Section 9.1.2, the fault $[\boldsymbol{e}(k_h)]$ has to be determined for the measured sequences of qualitative input values

$$[\boldsymbol{U}(0 \ldots k_h)] = ([\boldsymbol{u}(0)],\ ...,\ [\boldsymbol{u}(k_h)])$$

and qualitative output values

$$[\boldsymbol{Y}(0\,...\,k_h)] = ([\boldsymbol{y}(0)],\ ...,\ [\boldsymbol{y}(k_h)]).$$

The model of the faulty quantised system used here is given by the automaton

$$\tilde{\mathcal{S}} = (\mathcal{N}_{\tilde{z}}, \mathcal{N}_v, \mathcal{N}_w, \tilde{L}, \mathrm{Prob}(\tilde{z}(0)))$$

defined in Section 9.4.4. Therefore, the diagnostic problem can be solved by directly applying the diagnostic method developed for stochastic automata in Section 8.4 after

$V(0\,...\,k_h)$	is replaced by	$[\boldsymbol{U}(0\,...\,k_h)]$
$W(0\,...\,k_h)$		$[\boldsymbol{Y}(0\,...\,k_h)]\,.$

With these substitutions, Eqs. (8.91) – (8.93) yield the following result:

$$\mathrm{Prob}([\boldsymbol{e}(k_h)] \mid k_h) = \frac{\sum\limits_{\substack{[\boldsymbol{e}(k_h+1)]\\ [\boldsymbol{x}(k_h+1)],[\boldsymbol{x}(k_h)]}} L(k_h)\cdot G_f(k_h)\cdot \mathrm{Prob}([\boldsymbol{e}(k_h)],[\boldsymbol{x}(k_h)] \mid k_h-1)}{\sum\limits_{\substack{[\boldsymbol{e}(k_h)],[\boldsymbol{e}(k_h+1)]\\ [\boldsymbol{x}(k_h)],[\boldsymbol{x}(k_h+1)]}} L(k_h)\cdot G_f(k_h)\cdot \mathrm{Prob}([\boldsymbol{e}(k_h)],[\boldsymbol{x}(k_h)] \mid k_h-1)} \qquad (k_h = 0,1,...) \tag{9.52}$$

with

$$\mathrm{Prob}([\boldsymbol{e}(k_h)],[\boldsymbol{x}(k_h)] \mid k_h-1) = \frac{\sum\limits_{[\boldsymbol{e}(k_h-1)],[\boldsymbol{x}(k_h-1)]} L(k_h-1)\cdot G_f(k_h-1)\cdot \mathrm{Prob}([\boldsymbol{e}(k_h-1)],[\boldsymbol{x}(k_h-1)] \mid k_h-2)}{\sum\limits_{\substack{[\boldsymbol{e}(k_h)],[\boldsymbol{e}(k_h-1)]\\ [\boldsymbol{x}(k_h)],[\boldsymbol{x}(k_h-1)]}} L(k_h-1)\cdot G_f(k_h-1)\cdot \mathrm{Prob}([\boldsymbol{e}(k_h-1)],[\boldsymbol{x}(k_h-1)] \mid k_h-2)} \qquad (k_h = 1,2,...) \tag{9.53}$$

$$\mathrm{Prob}([\boldsymbol{x}(0)],[\boldsymbol{e}(0)] \mid -1) := \mathrm{Prob}([\boldsymbol{e}(0)])\cdot \mathrm{Prob}([\boldsymbol{x}(0)]) \tag{9.54}$$

and the abbreviation

$$L(k_h) := L([\boldsymbol{x}(k_h+1)],[\boldsymbol{y}(k_h)] \mid [\boldsymbol{x}(k_h)],[\boldsymbol{u}(k_h)],[\boldsymbol{e}(k_h)])\,. \tag{9.55}$$

9.6.2 Discussion of the result

Like in the observation algorithm, $\mathrm{Prob}([\boldsymbol{e}(k_h)] \mid k_h)$ determined by these equations is not identical to the probability distribution of the current fault $[\boldsymbol{e}(k_h)]$, but it is an approximation:

$$\mathrm{Prob}([\boldsymbol{e}(k_h)] \mid k_h) \approx \mathrm{Prob}([\boldsymbol{e}(k_h)] \mid [\boldsymbol{U}(0\,...\,k_h)],[\boldsymbol{Y}(0\,...\,k_h)]).$$

The set of faults for which $\mathrm{Prob}([\boldsymbol{e}(k_h)] \mid k_h) > 0$ holds is a superset of the set of fault candidates that would be obtained if the hybrid model of the quantised system

were used to solve the diagnostic problem that may really occur in the quantised system.

Theorem 9.3 (Diagnosis of quantised systems)
If the stochastic automaton $\mathcal{S}$ is a complete model of the quantised system, the fault that occurs in the quantised systems belongs to the set

$$\mathcal{F}(k_h) = \{[\boldsymbol{e}(k_h)] \,:\, \mathrm{Prob}([\boldsymbol{e}(k_h)] \,|\, k_h) > 0\}, \tag{9.56}$$

where $\mathrm{Prob}([\boldsymbol{e}(k_h)] \,|\, k_h)$ *is described by Eqs. (9.52) – (9.54).*

Note that the algorithm does not include a simulation of the system behaviour. The main idea is to determine the probability with which the system has made the state transitions that are necessary to produce the currently measured output for the currently measured input.

Corollary 9.2 (Diagnostic results for the quantised system)

- *The quantised system is known to be subjected to some fault $f \in \mathcal{F}(k_h)$.*
- **Fault detection:** *If $f_0 \notin \mathcal{F}(k_h)$ holds, the quantised system is known to be subjected to some fault (where f_0 denotes the faultless system).*
- **Fault identification:** *If $\mathcal{F}(k_h) = \{f_i\}$ is a singleton, the system is known to be subjected to fault f_i provided that the occurring fault belongs to the set $\mathcal{F}$.*

As long as $\mathcal{F}(k_h)$ has more than one element, the probability $\mathrm{Prob}(f \,|\, k_h)$ describes an approximation of the probability that the fault f occurs.

Diagnosability of quantised systems. The results on diagnosability of stochastic automata apply directly to the quantised system if the automaton represents a *complete* model. The quantised system is diagnosable whenever the automaton is diagnosable. That is, if the fault is diagnosable from the less precise model it can also be diagnosed from the precise representation given by the continuous-variable model together with the quantisers and the injector.

9.6.3 Diagnostic algorithm

These results lead to the following diagnostic algorithm in which $[\boldsymbol{U}]$ and $[\boldsymbol{Y}]$ denote the sequences $[\boldsymbol{U}(0\,...\,k_h)]$ and $[\boldsymbol{Y}(0\,...\,k_h)]$ of length k_h+1. The following abbreviations are used:

$$h([\boldsymbol{e}(k_h)],[\boldsymbol{x}(k_h)]) = \sum_{[\boldsymbol{e}(k_h+1)]\ [\boldsymbol{x}(k_h+1)]} L(k_h)\cdot G_f(k_h)\cdot \mathrm{Prob}([\boldsymbol{x}(k_h)]\mid\ [U(0\,...\,k_h-1)],[Y(0\,...\,k_h-1)])$$

$$p_k([\boldsymbol{e}(k_h)],[\boldsymbol{x}(k_h)]) = \mathrm{Prob}([\boldsymbol{e}(k_h)],[\boldsymbol{x}(k_h)]\mid\ [U(0\,...\,k_h)],[Y(0\,...\,k_h)])$$

$$p_r([\boldsymbol{e}(k_h+1)],[\boldsymbol{x}(k_h+1)]) = \mathrm{Prob}([\boldsymbol{e}(k_h+1)],[\boldsymbol{x}(k_h+1)]\mid\ [U(0\,...\,k_h)],[Y(0\,...\,k_h)])$$

Algorithm 9.2 *Diagnosis of quantised systems*

Given: Complete model $\mathcal{S}$ of the quantised system.
Fault model $\mathcal{S}_f$.
Initial state probability distribution $\mathrm{Prob}([\boldsymbol{x}(0)])$.
Initial fault probability distribution $\mathrm{Prob}([\boldsymbol{e}(0)])$.

Initialisation: $p_r([\boldsymbol{e}],[\boldsymbol{x}]) = \mathrm{Prob}([\boldsymbol{e}(0)] = f)\cdot\cdot\mathrm{Prob}([\boldsymbol{x}(0)] = z)$ for all $f \in \mathcal{N}_f$ and $z \in \mathcal{N}_z$
$k_h = 0$.

1. Measure the current input $[\boldsymbol{u}(k_h)] = v$ and output $[\boldsymbol{y}(k_h)] = w$.
2. For all $[\boldsymbol{e}] \in \mathcal{N}_f$ and $[\boldsymbol{x}] \in \mathcal{N}_x$ determine
$h([\boldsymbol{e}],[\boldsymbol{x}]) = \sum_{[\bar{\boldsymbol{e}}],[\bar{\boldsymbol{x}}]} L([\bar{\boldsymbol{x}}],[\boldsymbol{y}(k_h)]\,|\,[\boldsymbol{x}],[\boldsymbol{e}],[\boldsymbol{u}(k_h)])\cdot \cdot G_f([\bar{\boldsymbol{e}}]\,|\,[\boldsymbol{e}])\cdot p_r([\boldsymbol{e}],[\boldsymbol{x}])$.
3. If $\sum_{[\boldsymbol{e}],[\boldsymbol{x}]} h([\boldsymbol{e}],[\boldsymbol{x}]) = 0$ holds, stop the algorithm (inconsistent I/O pair or wrong initial distributions).
4. For all $[\boldsymbol{e}] \in \mathcal{N}_f$ and $[\boldsymbol{x}] \in \mathcal{N}_x$ determine
$$p_k([\boldsymbol{e}],[\boldsymbol{x}]) = \frac{h([\boldsymbol{e}],[\boldsymbol{x}])}{\sum_{[\boldsymbol{e}],[\boldsymbol{x}]} h([\boldsymbol{e}],[\boldsymbol{x}])}.$$

5. For all $[\boldsymbol{e}] \in \mathcal{N}_f$ and $[\boldsymbol{x}] \in \mathcal{N}_x$ determine

$$p_r([\boldsymbol{e}],[\boldsymbol{x}]) = \frac{\sum\limits_{[\bar{\boldsymbol{e}}],[\bar{\boldsymbol{x}}]} L([\boldsymbol{x}], w \mid [\bar{\boldsymbol{x}}], [\boldsymbol{u}(k_h)], [\bar{\boldsymbol{e}}]) \cdot G_f([\boldsymbol{e}] \mid [\bar{\boldsymbol{e}}]) \cdot p_r([\bar{\boldsymbol{e}}],[\bar{\boldsymbol{x}}])}{\sum\limits_{[\boldsymbol{e}],\,[\boldsymbol{x}]} h([\boldsymbol{e}],[\boldsymbol{x}])}.$$

6. Determine $\mathrm{Prob}([\boldsymbol{e}(k_h)] \mid k_h) = \sum_{[\boldsymbol{x}]} p_k([\boldsymbol{e}],[\boldsymbol{x}])$.
7. Determine $\mathcal{F}(k_h)$ according to Eq. (9.56).
8. $k_h := k_h + 1$
   Continue with Step 1.

Result: $\mathrm{Prob}([\boldsymbol{e}(k_h)] \mid k_h)$ and $\mathcal{F}(k_h)$ for increasing time horizon k_h.

Example 9.8 *Diagnosis of a quantised tank system*

To demonstrate this diagnostic algorithm, the tank system of Section 2.1 is considered as a quantised system. On the left-hand side of Fig. 9.25 the sequence $[\boldsymbol{U}(0 \ldots k_h)]$ of measured quantised input values is shown. The upper sequence corresponds to the valve position $[V_{12}]$ and the lower to the quantised pump input $[u_P]$. The right-hand side of this figure shows the measured output sequence $[\boldsymbol{Y}(0 \ldots k_h)]$, which describes to the quantised outflow $[q_M]$ of Tank 2. The task is to determine from these sequences the unknown quantised fault $[\boldsymbol{e}(k)] = [c_L]$ at each time instant $k_h = 0, 1, \ldots$, for unknown initial state set $\mathcal{X}_0$. The fault is assumed to be constant during the experiment, so that the fault model (8.95) can be used.

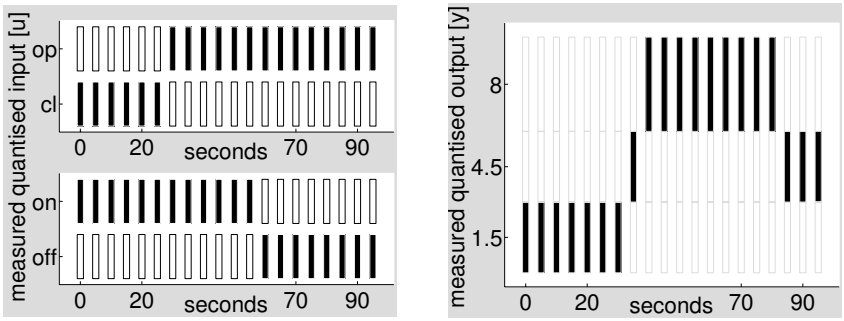
On the left-hand side of Fig. 9.26 the diagnostic result is depicted. It is shown how the probability distribution $\mathrm{Prob}([\boldsymbol{e}(k_h)] \mid k_h)$ changes for $k_h = 0, 1, \ldots$. Initially both faults have the same probability. The probabilities change with increasing time horizon k_h. After 8 steps, the measured sequences are only consistent with the model for the faultless case $[e] = 1$, showing that no leakage occurred in this experiment. The case $[e] = 2$ corresponding to a leakage in Tank 1 is excluded. □

9.6.4 Reconfiguration in case of sensor or actuator failures

If sensor or actuator failures are not included in the fault set $\mathcal{F}$, the diagnostic Algorithm 9.2 stops in Step 3 as soon as a faulty sensor or actuator causes that the denominator $\sum_{[\boldsymbol{e}],[\boldsymbol{x}]} h([\boldsymbol{e}],[\boldsymbol{x}])$ to become zero. In this case, the faulty sensor or actuator can be isolated by using the observer schemes introduced in Section 8.7. Furthermore, the diagnostic algorithm can be automatically reconfigured as described in Section 8.7.2. This is illustrated by the following example.

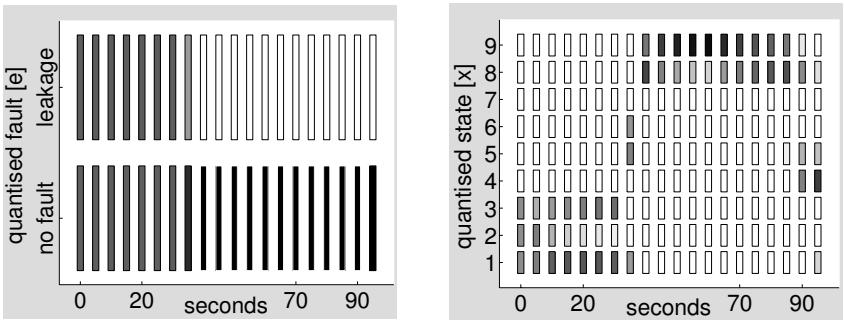
Example 9.9 *Automatic reconfiguration of diagnosis*

The automatic reconfiguration of diagnosis is shown using the example of the two-tank system of Section 2.1. Actuators are the valve V_{12} which can be either closed or opened and the pump P which can be switched on with $u_P = u_{P,nom}$ or switched off with $u_P = 0$. As plant fault, a



(a) (b)

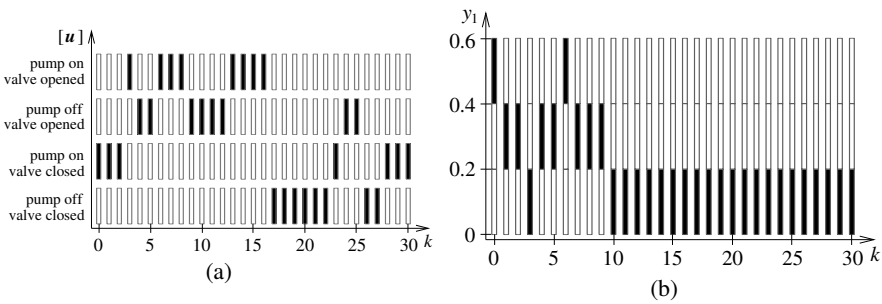
Fig. 9.25. Quantised input and output sequences available for fault diagnosis



(a) (b)

Fig. 9.26. Diagnostic result (left) and observed states (right)

leakage in the left tank is considered. The levels in both tanks are measured by means of the discrete level sensors at the positions leading to the state-space partition shown in Fig. 9.4. For simplification, the discrete level sensors at Tank 1 are treated as quantised level sensor 1 and those at Tank 2 as quantised level sensor 2. The sampling time is $T_s = 10\,\mathrm{s}$.



(a) (b)

Fig. 9.27. Quantised input sequence (left) and faulty interval measurements of level sensor 1 (right).

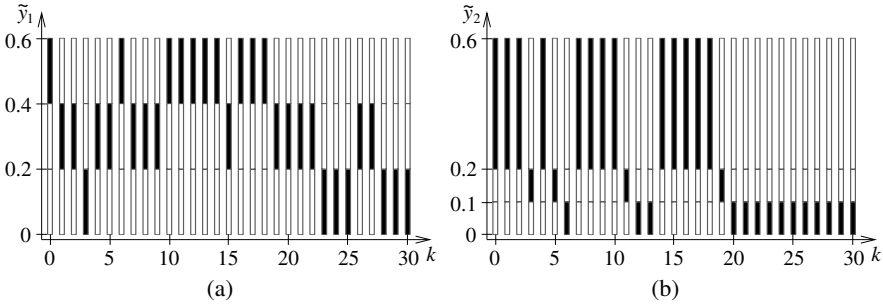


Fig. 9.28. Sequence of the true quantised liquid levels in the left and right tank, respectively.

In an experiment, the quantised input sequence shown on the left-hand side of Fig. 9.27 has been applied to the tank system. This yields the sequences of quantised tank levels shown in Fig. 9.28. Instead of the correct output shown on the left-hand side of the second figure, sensor 1 yields the sequence shown on the right-hand side of the first figure. It can be seen that from time $k = 10$ onwards, the sensor yields the measurement value $[0, 0.2)$ independently the actual level in the left tank.

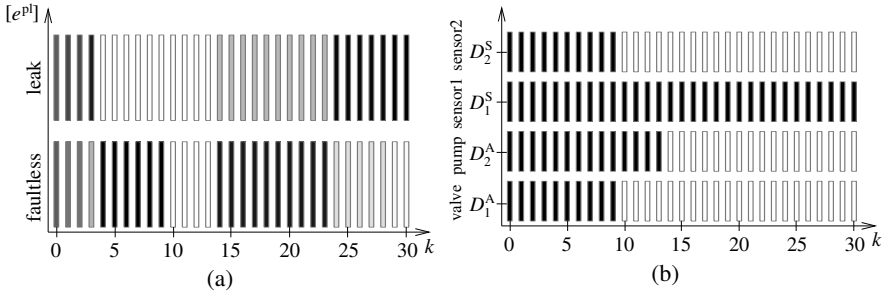


Fig. 9.29. Diagnostic result for the plant fault (left) and denominators of the blocks of the generalised observer scheme (right).

To the measured sequences, the generalised observer scheme shown in Fig. 8.23 is applied. Furthermore, two blocks for actuator supervision as shown in Fig. 8.27 are used. That is, besides the diagnostic result, the four signals $D_1^S(k)$, $D_2^S(k)$, $D_1^A(k)$ and $D_2^A(k)$ are determined, which correspond to the denominators of the diagnostic blocks without Sensor 1, Sensor 2, Actuator 1 (valve) or Actuator 2 (pump), respectively. The diagnostic result is shown on the left-hand side of Fig. 9.29. The values of the four denominators are shown on the right-hand side of the figure, where a black bar indicates a nonzero denominator.

From the left-hand side of Fig. 9.29, it can be seen that until $k = 9$ the diagnostic algorithm works correctly and that the faultless case is isolated for unknown initial plant fault after a few steps. At $k = 10$, when Sensor 1 breaks down, the denominator $D(s)$ of the main diagnostic block becomes zero. The diagnostic block yields no further diagnostic results which is indicated by the white bars from $k = 10$ onwards. The right-hand side of the figure shows that at $k = 10$, two other denominators, namely $D_2^S(k)$ and $D_1^A(k)$, also become zero indicating that neither a valve fault nor a failure of in Sensor 2 could have caused the inconsistency with the model of the main diagnostic block. After a few more steps at $k = 14$, the measured se-

quences also become inconsistent with the model that has no regard of the pump input. This indicates that Sensor 1 must be faulty because the only block which is still consistent with the measurement sequences is the block of the generalised observer scheme that does not use the information of Sensor 1.

Having identified the faulty component, the diagnostic system is reconfigured so that only Sensor 2 is used for diagnosis. Such a diagnostic block is already included in the generalised observer scheme. The probabilities of the plant faults determined by this block are shown on the left-hand side of Fig. 9.29 from time $k=14$ onwards. Note that the new diagnostic block implicitly performs a state observation of the level in Tank 1.

At time $k=18$, a leakage occurs in the left tank and is present until the end of the experiment. It can be seen that after some time, this plant fault is identified. However, the reconfigured diagnostic system has a lower performance because it obtains less information due to the loss of Sensor 1. If Sensor 1 was still operating, the leakage could be detected already at time $k=19$ when the level in the left tank decreases for a closed connecting valve, cf. Fig. 9.27 (left) and Fig. 9.28 (left). The reconfigured scheme is slower but allows that the diagnosis can be continued though the original diagnostic block could no longer be used from time $k=10$ onwards. □

9.6.5 Extensions and application examples

Diagnosis of transient faults. The diagnostic problem dealt with so far concerned the task to find the current fault $f(k_h)$ by using the measured I/O pair. The solution to this problem is not appropriate if the fault is apparent in the measurements only after some time delay. Transient faults, which vary quickly in time, represent such faults, where the moment of the fault occurrence lies several time instants before the effects of the fault become measurable.

To understand the difference with respect to the diagnostic problem tackled until now, remember that the fault is described by the sequence

$$F(0\,...\,k_h) = (f(0), f(1), f(2),\ ...,\ f(k_h))$$

and that until now only the last element of this sequence has to be found. If the fault becomes "visible" from the measurements only after some time delay, at time k_h the task to be solved is to find all possible sequences $F(0\,...\,k_h)$ and to decide whether these sequences include faults $f(k_h - k) \neq f_0$ that occurred k time instants ago, where f_0 again denotes the faultless operation mode of the system.

As a consequence, the problem has to be posed in such a way that all possible fault sequences $F(0\,...\,k_h)$ have to be found for which the measured I/O pair $([\boldsymbol{U}], [\boldsymbol{Y}])$ belongs to the system behaviour $\mathcal{B}_{\text{qual}}(k_h)$ of the quantised system. Then it is known that the sequence $F(0\,...\,k_h)$ can have happened and, consequently, the faults occurring in this sequence may have occurred at the corresponding time instants.

The diagnostic method explained in this chapter can be extended to solve this problem. The main idea remains the same but the summation occurring in the formulas have to be made over all possible fault sequences $F(0\,...\,k_h)$.

Diagnosis of discrete-event quantised systems. This chapter concerned discrete-time quantised systems, whose continuous-variable core can be described by the

Eqs. (9.3), (9.4) with given sampling time. However, another viewpoint can be adopted concerning the temporal quantisation. If a continuous-time system is considered, the quantiser may not only determine the current qualitative state, but it may also identify the time instants at which these qualitative values change. Then the continuous-time continuous-variable system "moves" whenever its output trajectory $\boldsymbol{y}(t)$ crosses a border between two adjacent output space partitions $\mathcal{Q}_y(i)$ and $\mathcal{Q}_y(j)$. A change of the qualitative value $[\boldsymbol{y}]$ is called an event.

The main ideas described in this chapter can be used to diagnose discrete-event quantised systems. Then, the stochastic automaton describes the sequence of discrete events generated by the quantised system. The diagnostic algorithm can be directly applied if the interface between the quantised system and the algorithm is modified such that the I/O sequences are composed of events rather than of qualitative input and output values obtained by sampling.

Although the main ideas are the same, the results may differ considerably. The reason for this is given by the fact that stochastic automata of discrete-event quantised systems have no information about the temporal distance of the events, but this temporal distance may be decisive for the diagnosability.

Therefore, timed discrete-event models as, for example, semi-Markov processes have to be used. Such models can be set up in a similar way as described in Section 9.4 if the probability of the occurrence of events are evaluated also with respect to their occurrence times. The diagnostic algorithm presented for stochastic automata can be extended to this kind of models.

Application examples. The methods described in this chapter have been applied to different laboratory experiments and practical problems.

- **Diagnosis of a batch process:** The example used to illustrate the methods throughout this chapter presents a hybrid system with switching dynamics.

- **Diagnosis and fault-tolerant control of a neutralisation process:** Many important signals that occur in the process industry are not precisely measurable or even immeasurable. A neutralisation process has been used to demonstrate the applicability of the observation and diagnostic methods. The fault-tolerant control of a part of this process is described in Section 10.2.

- **Diagnosis of an $\mathbf{H}_2$ compressor:** For large industrial compressor systems, no quantitative model (9.3), (9.4) is available. By approximating the behavioural relation by the relative frequency with which the system under consideration has changed its qualitative state during its operation over several years, a stochastic automaton as qualitative model of the compressor system has been obtained and used for on-line diagnosis.

- **Diagnosis of the power stage of a diesel engine:** Automotive systems have quick dynamics and, hence, rather strong real-time constraints for diagnosis. The quantised system approach has the advantage of reducing the information available

to that which is necessary to solve the diagnostic task. Dealt with as a discrete-event quantised system, the power stage of a diesel engine has successfully been equipped with a diagnostic module.

Example 9.10 *Diagnosis of the air path of a diesel engine*

Growing demands on automobiles in terms of reliability, economy, and safety necessiate severe improvements of automated on-board diagnosis. To guarantee low emission levels the diagnosis of the injection, compression, and combustion plays a key role. This example concerns the qualitative diagnosis of the air path of a diesel engine with a single turbo charger (Fig. 9.30).

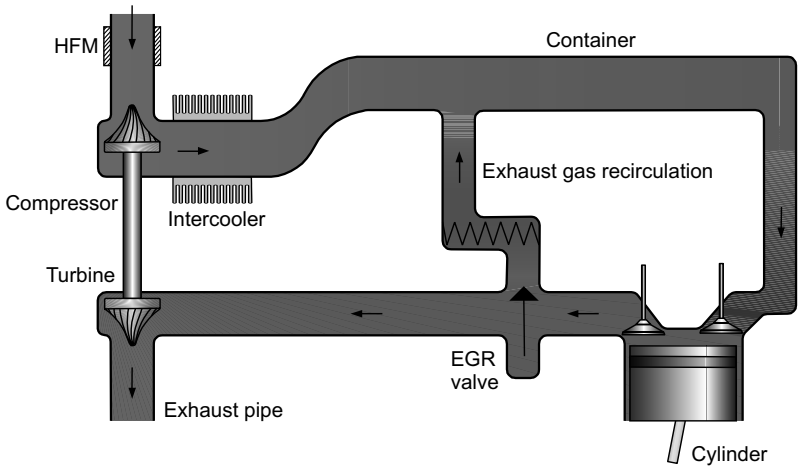


Fig. 9.30. Air path of a diesel engine

The air path consists of the following components (Fig. 9.30):

- Hot-wire air flow meter (HFM): Measurement of the incoming air flow.
- Compressor: Compression of the incoming air flow.
- Intercooler: Cooling of the compressed air to increase the density.
- Container: Blending of the compressed air with parts of the waste gas.
- Cylinder: Combustion of the fuel and airmix from the container.
- Exhaust gas recirculation valve (EGR valve): Control of the percentage of the recirculated exhaust gas.
- Turbine: Driving the compressor; equipped with a variable turbine geometry (VTG).
- Exhaust pipe: Channelling of the waste gas to the outside.

The air path is subject to the input signals n_E (engine speed), m_F (fuel flow per stroke), A_{EGR} (effective area of the EGR valve) and A_{VTG} (effective area of the turbine). The following output signals can be measured: q_1 (incoming air flow), p_2 (pressure in the container) and T_2 (temperature in the container). The continuous-variable state-space model (9.3), (9.4) has seven state variables, four input and three output signals (Fig. 9.31).

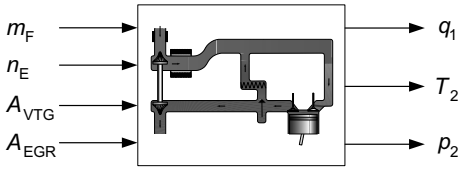


Fig. 9.31. Block diagram of the air path

The special structure of the air path results in a strong coupling among the different system states. The recirculated exhaust gas affects the airmix that streams into the engine by changing its temperature, pressure, and oxygen content. In turn, the airmix influences the exhaust gas that is leaving the engine. A second feedback is realised through the turbo charger. It couples the input air flow with the exhaust gas flow.

Faults that occur in the components of this system influence the emission, which is subject to legal restrictions. Therefore, the sensor and actuator faults listed in the table below have to be considered for diagnosis. The diagnostic task is to detect faults in the air system and to identify the faults f_1 throught f_9 by using measurement sequences of the sensors and input sequences to the actuators mentioned above.

Fault	Fault description
f_0	Faultless operation
f_1	HFM offset +0,02 kg/s
f_2	HFM breakdown 0,0001 kg/s
f_3	HFM drift -20
f_4	EGR blocked close
f_5	EGR blocked open
f_6	T_2-sensor drift +30
f_7	T_2-sensor drift -30
f_8	p_2-sensor drift +30
f_9	p_2-sensor drift -30

The airpath is considered as a quantised system. For all measured signals, a parition of the signal space is introduced. For example, for the air flow q_1 six regions starting from "very low" up to "maximum" are distinguished after the discretisation (Fig. 9.32). The dark or light rectangles indicate in which interval the signal q_1 lies at the respective time instant. As can be seen in the figure, the trajectories of the signal q_1 for the two fault cases f_0 and f_3 can be distinguished easily in spite of the rough discretisation.

For the evaluation of the diagnostic method, the input signals from test series with an experimental car have been used. In the tests the EGR valve was closed, which was adopted as a modelling assumption.

The qualitative model has been determined by the abstraction algorithm explained in Section 9.4.

Results. The results obtained by diagnosing the different faults are briefly presented in the following. It has turned out that the faults f_2, f_6 – f_9 are very easy to be diagnosed and therefore the attention is focused on the remaining faults. The diagnostic result describes the probability for the occurrence of each fault, which is marked by the intensity of the bars in the following diagrams.

Figure 9.33 shows the diagnostic result for the faultless case f_0. The faultless operation is unambigously recognised. After starting the diagnosis at time 0 the faultless case is the most

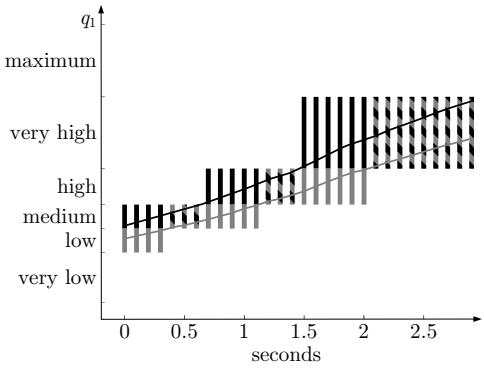


Fig. 9.32. Air flow q_1 during faultless operation f_0 (black) and during fault f_3 (grey)

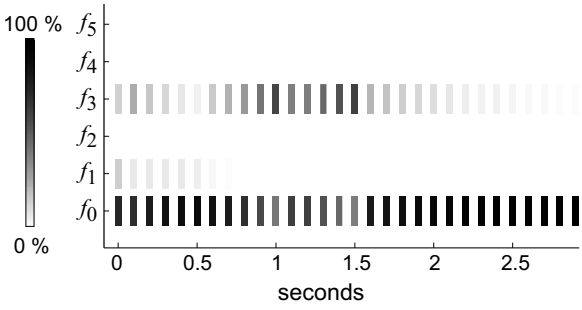


Fig. 9.33. Identification of the faultless case

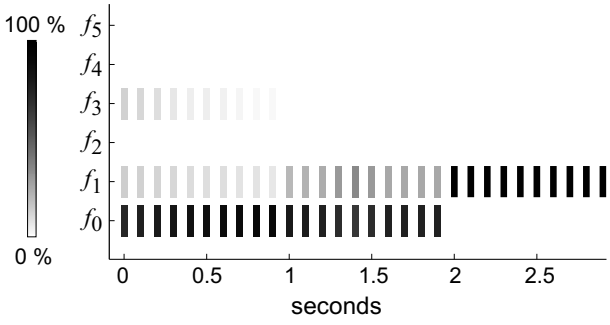


Fig. 9.34. Identification of the fault f_1

likely one but the faults f_1 and f_3 cannot be excluded. The higher initial probability of the faultless case is given to the diagnostic system because it is assumed that the system normally works without any faults. After a short time of diagnosing, the fault f_1 is excluded. During the time span from one to two seconds the signals from the air path coincide as well with the faultless model as with the model for the fault f_3. After two seconds the probability for the fault f_3 decreases continuously and the diagnostic system shows the correct result: no fault is present in this experiment.

The figure illustrates the fact that the diagnostic result is obtained by accumulating the information included in the measurement sequences. After about 2.5 seconds, the diagnostic algorithm has sufficient information for determining the faultless case unambiguously.

If the air path is affected by the fault f_1, the faults f_0, f_1 and f_3 are stated possible when the diagnosis starts (Fig. 9.34). The higher probability of f_0 is due to the higher initial probability explained above. During the diagnosis the probability of f_3 decreases continuously and the probability of f_1 increases accordingly. At 1.9 seconds a measurement has occurred that is impossible in the faultless case. Therefore, the faultless case is excluded and the diagnostic system returns the correct result.

Evaluation of the results. The presented qualitative diagnostic method allows diagnosing faults that, in general, cannot be diagnosed by signal-based methods. The qualitative approach means that the behaviour of the air path is modelled as coarse as possible so that many details can be ignored but the faulty behaviour can still be distinguished from the nominal behaviour. Thereby the unavoidable uncertainties of the model are taken into account and the air path can be modelled in a simple manner. □

These examples show that difficulties in the measurement of important output signals are not the only reason for using the quantised system approach to the diagnosis of continuous-variable systems. If the system has a hybrid dynamics, the method to use a discrete-event abstraction as the model for diagnosis has the advantages to deal with a unique discrete-event system and to use recursive algorithms that can be applied in real time.

9.7 Fault-tolerant control of quantised systems

9.7.1 Reconfiguration problem

As the actions to be made in fault-tolerant control refer to severe changes of the control input and possibly include switches to new sensors or actuators, the quantised system approach is particularly reasonable for these steps. The discrete-event model of the plant has direct reference to the discrete decision variables to be found. This fact will become obvious in this section. In a more elaborate form, this method is illustrated by its application to the neutralisation process described in Section 10.2.

The control problem considered here concerns the following situation. Assume that a fault has appeared in the plant and a diagnostic algorithm has found this fault. As the diagnosis takes some time, the system has moved from its nominal operation point to another point, because the nominal controller is applied to the faulty system. This situation occurs, in particular, if a sensor or an actuator fails, because then the control law is restricted to that part of the plant-controller interface that is still

working. The control to be found requires the reconfiguration of the control loop, because the controller can have access only to the faultless sensors and faultless actuators.

In this situation a fault-tolerant controller has first to move the system state back into the nominal operation point and then to stabilise the system in this point. Therefore, the control algorithm consists of two steps:

1. A discrete controller has to be found that moves the operation point back into the nominal one.
2. A new continuous controller has to be found that stabilises the faulty system in the operation point.

The first controller has, usually, discrete input and output signals, because the operation point has to be moved through a large part of the state space, which does not necessitate precise numerical measurements. The stabilising controller is a feedback controller of the faulty system, which can be designed by well known controller design methods after the available actuators and sensors have been identified and the reconfigurability investigated (cf. Chapters 4 and 5). The following concentrates on the first part of the fault-tolerant control algorithm.

Main idea. As the movement of the operation point requires broad changes of the input, the quantised systems approach is reasonable for the solution of this task even if the plant has continuous inputs and outputs. Hence, quantisers are introduced deliberately for continuous-variable signals. This step facilitates a uniform discrete-event view on the system, which may have both discrete and continuous inputs and outputs.

A discrete-event model is abstracted by the methods described in Section 9.4. For the explanation of the main idea of the control design for this model, a nondeterministic automaton is sufficient, because the extension to stochastic automata is straightforward. The automaton can be graphically interpreted by the automaton graph. The directed edges of the graph show which state transitions among the quantised states the system can perform, where the edge label $([\boldsymbol{u}], f)$ denotes the quantised control input and the fault under which this transition occurs. By using this representation, the control problem can be formulated as a graph search problem. A path from the vertex representing the current qualitative state has to be found towards one of those vertices that represent the claimed operation point. This path may include only those edges that are valid for the current fault, which is assumed to be given by the diagnostic algorithm. The labels belonging to the edges in the path signify which control actions have to be used. The sequence of this control actions represents the solution to the control problem.

This method will be explained now in more detail. It is applied in Section 10.1.3 to reconfigure the controller of a three-tank system.

9.7.2 Graph-theoretic formulation of the control problem

Since a diagnostic algorithm can only find the fault after the system under consideration has sufficiently changed its behaviour and, thus, left its nominal state, the qualitative initial state $[\boldsymbol{x}(0)]$ is different from the required state and assumed in the following to be arbitrary, but known. The problem is to find a controller

$$[\boldsymbol{u}(k)] = k_q([\boldsymbol{x}(k)], f) \tag{9.57}$$

such that the operation point is moved from the current point $[\boldsymbol{x}(0)]$ into one operation point included in the set $\mathcal{Z}_{\mathrm{Aim}}$

$$\mathcal{Z}_{\mathrm{Aim}}(f) = \{[\boldsymbol{x}] \,|\, \text{Control specifications are satisfied}\} \subset \mathcal{N}_z. \tag{9.58}$$

These points are selected so as to satisfy the given specifications on the closed-loop system under the faulty conditions. Note that the controller (9.57) uses the quantised information about the current state $\boldsymbol{x}$ and represents a qualitative state feedback. More restrictive control laws like a qualitative output feedback

$$[\boldsymbol{u}(k)] = \tilde{k}_q([\boldsymbol{y}(k)], f)$$

can be found in a similar way.

A graph-theoretic interpretation of this problem can be obtained as follows. At time $k = 0$ the quantised system is in the qualitative initial state $z_0 = [\boldsymbol{x}(0)]$, which is represented by the vertex of the automaton graph with the same name. The controller (9.57) has to be chosen such that the system arrives at the set $\mathcal{Z}_{\mathrm{Aim}}(f)$ and remains there. That is, the quantised system together with the controller has to be *qualitatively stable*.

Graph-theoretic characterisation of qualitative stability. The qualitative stability can be tested by the automaton graph G_c of the closed-loop system. Assume that the controller (9.57) is already known. Then the non-deterministic automaton

$$N_c = (\mathcal{N}_z, \mathcal{N}_f, L_c, z_0)$$

of the closed-loop system has the state transition relation

$$L_c(z, f) = L_n(z, k_q(z, f), f)$$

where L_n is the state transition relation of the non-deterministic automaton of the plant subject to fault f. The graph G_c of this automaton can be obtained from the automaton graph of the plant by deleting all edges (z_i, z_j) whose labels $([\boldsymbol{u}], f)$ do not satisfy the control law (9.57) with $[\boldsymbol{x}(k)] = z_i$.

For the stability analysis the set of vertices $\mathcal{Z}_{\mathrm{Aim}}$ is replaced in the graph G_c by a new vertex z_{Aim}. All edges (z, z') starting in some vertex $z \notin \mathcal{Z}_{\mathrm{Aim}}$ and going to some $z' \in \mathcal{Z}_{\mathrm{Aim}}$ are replaced by an edge (z, z_{Aim}). In this way a new graph denoted by $G_c'(\mathcal{N}_z', \mathcal{E}_c')$ is obtained. The reconfiguration problem is solved if the conditions given in the following theorem are satisfied:

Theorem 9.4 (Qualitative stability)
A qualitative controller (9.57) solves the control problem, if the graph $G'_c(\mathcal{N}'_z, \mathcal{E}'_c)$ satisfies the following three conditions:

1. *The graph G'_c has no strongly connected vertices.*
2. *There are no self-circles around any vertex $z \neq z_{\mathrm{Aim}}$.*
3. *The vertex z_{Aim} is the only end vertex of G'_c. That is, z_{Aim} is the only vertex that has no outgoing edge.*

The conditions appearing in the theorem have been proved to be necessary and sufficient for the qualitative stability of the quantised closed-loop system provided that the qualitative model is complete. Hence, the reconfiguration problem can be formulated as follows:

> **Reconfiguration problem:** *Find a controller (9.57) such that the reduced automaton graph $G'_c(\mathcal{N}'_z, \mathcal{E}'_c)$ of the closed-loop system satisfies the three conditions given in Theorem 9.4.*

9.7.3 A reconfiguration method

First, some notions from graph theory have to be recalled. It is assumed that in the automaton graph there is at most one edge between a predecessor state and a successor state. This assumption can be satisfied by lumping parallel edged together, where the labels are combined accordingly.

A subgraph $T(\mathcal{N}_z, \mathcal{E}_t)$ of $G(\mathcal{N}_z, \mathcal{E})$ with $\mathcal{E}_t \subseteq \mathcal{E}$ is called a *spanning tree* if any pair of its vertices z_i and z_j are connected and if T has no cycle. Two vertices z_i and z_j are called *strongly connected* if there exists a path from z_i towards z_j and a path from z_j towards z_i. A subgraph $G_i(\mathcal{N}_{zi}, \mathcal{E}_i)$ of G is called a strongly connected component if all vertices of $\mathcal{N}_{zi}$ are strongly connected. The graph that results after replacing every strongly connected component by a new vertex is called the *condensed graph*.

Algorithm 9.3 *Solution of the reconfiguration problem by means of a qualitative model*

Given: Automaton graph $G(\mathcal{N}_z, \mathcal{E})$,
Goal set $\mathcal{Z}_{\mathrm{Aim}}$.

1. Determine the graph $G'(\mathcal{N}'_z, \mathcal{E}')$ by replacing the set of vertices $\mathcal{Z}_{\mathrm{Aim}}$ by the single vertex z_{Aim}.
2. Determine the condensed graph $\hat{G}(\hat{\mathcal{N}}_z, \hat{\mathcal{E}})$ of $G'(\mathcal{N}'_z, \mathcal{E}')$ by first determining all strongly connected components G_i of G' and second replacing G_i by hyper vertices.
3. For all strongly connected components G_i determine a spanning tree T_i.
4. Using the condensed graph and the spanning trees, construct a new graph $G_{new}(\mathcal{N}'_z, \mathcal{E}_{new})$ with $G_{new} = \hat{G} \cup T_1 \cup T_2 \cup \cdots \cup T_s$ where s is the number of strongly connected components.
5. Determine the control law $k_q(z, f)$ as follows: For every vertex $z \in \mathcal{N}'_z$ and every fault f, select the qualitative output $[\boldsymbol{u}]$ associated with the edge (z, z') in the graph G_{new}.

Result: Qualitative controller (9.57).

The strongly connected components G_i can be constructed using a depth-first-search algorithm whose complexity is linear with respect to the number of vertices. For each vertex of the condensed graph there exists a spanning tree T_i. Such trees can be found by Tremaux and Tarjan's algorithm whose complexity is linear. Each vertex $z \neq z_i$ of graph G' will be transferred to the vertex z_i of G'_{z_i} using spanning tree T_i. Then, for each vertex z_i there exists a path to the desired vertex $\mathcal{Z}_{\mathrm{Aim}}$ in the condensed graph $\hat{G}$.

The result is the mapping k_q of the controller (9.57), which can be represented by a table and which can be easily implemented.

9.8 Exercises

Exercise 9.1 *Selection of reasonable state-space partitions*

Consider an autonomous first-order discrete-time system. Under what conditions on the state partitions does the system map one state set precisely into another state set with the consequence that the quantised system can be described by a *deterministic* automaton?

Can this determinism be retained if the system has an input by appropriately partitioning the input space? □

Exercise 9.2 *Determinstic discrete-event behaviour of an undamped oscillator*
Consider an undamped oscillator described by the state-space model

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & \omega \\ -\omega & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \qquad \begin{pmatrix} x_1(0) \\ x_2(0) \end{pmatrix} = \begin{pmatrix} x_{10} \\ x_{20} \end{pmatrix}.$$

Show that the sampling time can be chosen such that the oscillator has a deterministic discrete-event behaviour in a state space where both state variables x_1 and x_2 are independently partitioned.

Does this result hold if the oszillator is damped (where the damping factor δ replaces the zeros in the system matrix)? □

Exercise 9.3 *Quantised tank system*
Consider a two-tank system in a state space, whose partitioning distinguishes merely between a full and an empty tank. At time $t = 0$ both tanks are filled. An outlet valve of the right tank is open. Set up a simple second-order state-space model and use the abstraction algorithm to get a non-deterministic or a stochastic automaton describing the quantised tank system.

Extend the model, if the left tank has a pump, which can be switched on and off and which produces an inflow into the left tank that is twice as large as the outflow from the right tank. How do you have to extend the automaton to cope with this extension of the quantised system? □

Exercise 9.4 *Spring-mass system*
Consider a damped spring-mass system. A fault reduces the mass be the factor two. Draw the movement of the system in the faultless and the faulty case and decide whether this fault can be diagnosed by only measuring quantised values of the mass position. □

9.9 Bibliographical notes

Results on the diagnosis of quantised systems have been obtained in two fields. The problem of abstracting discrete-event representations for quantised systems has been investigated as a step for the analysis of hybrid systems or for the verification of discrete controllers. The publications [138], [141] and [153] have shown that quantised systems have, in general, a non-deterministic qualitative behaviour and do not possess the Markov property. Hence, they cannot be represented precisely by any model that possesses the Markov property like stochastic automata.

Methods for abstracting discrete-event representations of discrete-time or discrete-event quantised systems have been proposed in [118], [138], [141], [153], [117], [201], [204], [246]all of which aim at finding complete models in the sense defined in Section 9.4. [204] showed that by using different definitions of the model state, a hierarchy of discrete abstractions can be obtained, which generate different numbers of spurious solutions.

In the field of computer-aided modelling the abstraction problem has been considered in [136] and [262] with the aim to find deterministic discrete-event representations, which, according to [138] is possible only for a very small class of quantised systems. There are only preliminary results concerning the question how to partition the signal spaces in order to

obtain abstractions with a small number of spurious solutions, cf. [153], [147], [115]. A connection of the stochastic automaton as discrete-event description of quantised systems and the Frobenius-Perron operator is given in [158].

On the other hand, the diagnosis of quantised systems is based on methods for diagnosing discrete-event systems described by automata, which have been elaborated in [128], [157], [211] and [210]. The complete solution to the diagnostic problem for stochastic automata given in [157] is the basis for the results reviewed in Section 8.4. First results for quantised systems have been described in [125] and [156]. In [63] it is shown that discrete-event representations of quantised systems can be used for diagnosis if and only if they are complete. This reference also gives an example to demonstrate that different models can be used for the same quantised system all of which are complete but differ concerning the number of spurious solutions and, hence, yield diagnostic results of different precision.

The solution to the state observation problem for the quantised system is based on the solution to the observation problem for stochastic automata. This problem has been dealt with only by a few authors, for example in [157] and [191].

The extensions to transient faults is described in [216].

The diagnosis of discrete-event quantised systems has be developed in [63] and [143], where the latter reference presents a method that takes the temporal event distances into account. The corresponding abstraction methods, which transforms the continuous-variable description of the system into a discrete-event model, are developed in [60] and [142].

Theorem 9.4 has been proved in [140]. The problem how to partition the state space in order to avoid the non-determinsm on the quantised system level and to get a deterministic automaton as the precise qualitative model of the quantised system is still open [147].

Application examples of fault diagnosis of quantised systems are presented in [156] (neutralisation process), [124] (H_2 compressor), [61], [62] (diesel injection system) and [64], [174] (air path of a diesel motor), which is summarised here as Example 9.10.

The reconfiguration method described in Section 9.7 has been first published in Chapter 12 of [3].

Chapter 10

Application examples

This chapter presents five applications that illustrate how the methods developed in the preceeding chapters can be applied under real practical conditions and how they can be combined to get a complete solution of fault-tolerant control problems. A three-tank system, a chemical process, a ship propulsion system, a steam generator and a steering-by-wire system for a warehouse truck are considered, each of which have been investigated in detail including experimental tests.

10.1 Fault-tolerant control of a three-tank system

10.1.1 Control problem

Consider the three coupled tanks depicted in Fig. 10.1. These tanks are connected by pipes which can be controlled by different valves. Water can be filled into the left and right tanks using two identical pumps. Measurements available from the process are the continuous water levels h_i of each tank and, additionally, from tank T_2 discrete signals from two capacitive proximity switches signalling whether the water level in the tank is above or below the position of the sensor.

In the nominal case (Fig. 10.2), only the left tank T_1 and the middle tank T_2 are used. The right tank T_3 and pump P_2 act as redundant hardware. The purpose of the system is to provide a continuous water flow $q_2(t) = q_N$ to a consumer. Therefore, the water level in the middle supply-tank T_2 has to be maintained within the interval $h_{2L} < h_2 < h_{2H}$, i.e. between the two discrete level sensors of tank T_2.

Water flows between the tanks can be controlled by several valves (V_{12L}, V_{12H}, V_{23L}, V_{23H}). All valves can only be completely opened or completely closed (on/off valves). The connection pipes between the tanks are placed at the bottom of the tanks (pipes with valves V_{12L}, V_{23L}) and at a height of h_H (pipes with valves V_{12H},

V_{23H}). One of the considered faults is a leakage in tank T_1 (see below). If such a leakage occurs, there is an additional outflow q_L of tank T_1 (cf. Fig. 10.1).

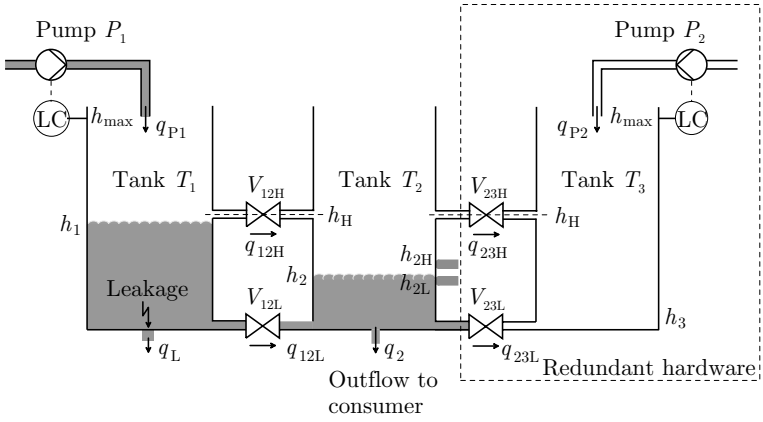


Fig. 10.1. Three-tank system

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

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

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

$$\dot{V} = A \cdot \dot{h} = \sum q_{\text{in}} - \sum q_{\text{out}} \, , \tag{10.1}$$

where $\sum q_{\text{in}}$ is the sum over all water inflows and $\sum q_{\text{out}}$ the sum over all water outflows of the tank. In (10.1), A is the cross-section area and h the water level in the cylindric tank. For the three tanks Eq. (10.1) yields:

$$\dot{h}_1 = \frac{1}{A}(q_{P1} - q_{12L} - q_{12H} - q_L) \tag{10.2}$$

$$\dot{h}_2 = \frac{1}{A}(q_{12L} + q_{12H} - q_{23L} - q_{23H} - q_2) \tag{10.3}$$

$$\dot{h}_3 = \frac{1}{A}(q_{P2} + q_{23L} + q_{23H}) \, . \tag{10.4}$$

The flows in Eqs. (10.2) - (10.4) depend on the levels h_1, h_2 and h_3 as well on the position of the valves and the commands u_{P1}, u_{P2} given to the pumps. For example, the existence of the flow q_{12H} depends on the water levels h_1 and h_2 and

the position of the valve V_{12H}. The flow is only nonzero if the valve is open and at least one liquid level exceeds the height h_H of the upper connecting pipe.

More precisely, the following expressions are obtained for the flows, with the parameters given in Table 10.1:

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

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

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

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

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

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

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

$$q_L = \begin{cases} c_L\sqrt{h_1} & \text{if } h_1>0 \text{ and leakage in tank 1} \\ 0 & \text{otherwise .} \end{cases}$$

Nominal configuration. In the nominal case, valves V_{12L}, V_{23H}, V_{23L} are closed and not in use. Valve V_{12H} is used to control the water level in tank T_2, pump P_1 to control the level in tank T_1. To control the water levels in the reservoir-tank T_1 and the supply-tank T_2, a conventional PI-controller and an discrete (on-off) controller are used (Fig. 10.2):

Table 10.1 Parameters and variables of the three-tank system and the controllers

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

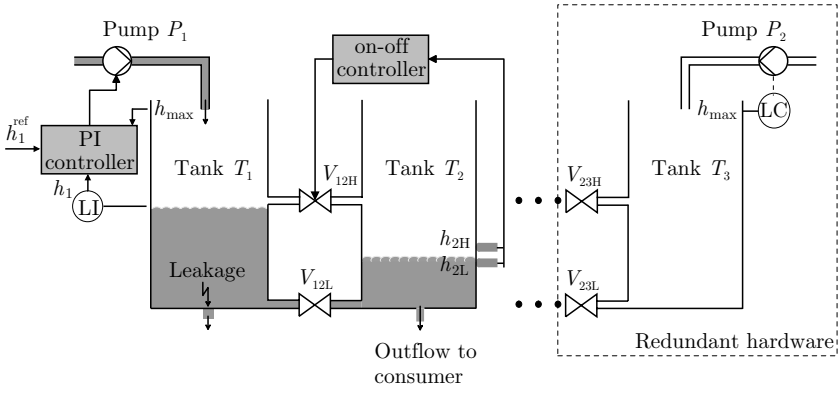


Fig. 10.2. Nominal configuration of the three-tank system

$$\begin{aligned} u_{P1}(t) &= k(h_1(t), h_1^{\mathrm{ref}}) \\ &= K_P \cdot (h_1^{\mathrm{ref}} - h_1(t)) + K_I \cdot \int_0^{'} (h_1^{\mathrm{ref}} - h_1(\tau))d\tau \end{aligned} \tag{10.5}$$

$$V_1 \quad = \quad \begin{cases} \text{open} & : \quad h_2 \leq h_{2L} \\ \text{close} & : \quad h_2 \geq h_{2H} \\ \text{no change} & : \quad h_{2L} < h_2 < h_{2H} \ , \end{cases} \tag{10.6}$$

where K_P and K_I are controller parameters and h_1^{ref} is the set-point for tank T_1. Equation (10.6) describes under what conditions the on-off controller changes the position of the valve from opened to closed or vice-versa. All parameters of the controllers are given in Table 10.1.

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

- Only the left and middle tank are in use, water level h_2 must be medium, the set-point for h_1 is chosen to h_1^{ref}.
- Valves $V_{12L}, V_{23L}, V_{23H}$ are closed.
- No leakage occurs ($q_L = 0$).
- The PI-controller (10.5) controls the level h_1 of tank T_1 with pump P_1 using a continuous level sensor.
- The on-off controller (10.6) controls the level h_2 of tank T_2 with valve V_1 using discrete level sensors.

Reconfiguration problem. Three different fault scenarios are given:

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

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

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

$$[h_2(k)] = medium \tag{10.7}$$

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

$$[h_1(k)] = empty \tag{10.8}$$

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

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

10.1.2 Generic component-based analysis of the three-tank system

This section applied the methods elaborated in Chapter 4 to the three-tank example.

Modelling of the field components. The three tank system is composed of the interconnection of 14 elementary components, namely

- 4 valves: V_1, V_{13}, V_2, V_{23},
- 2 pumps: P_1, P_2,
- 3 tanks: T_1, T_2, T_3,
- 3 level sensors: L_1, L_2, L_3,
- 2 controllers $C_1 C_2$.

The interconnecting pipes might also be considered as components, if the service they deliver was to be analysed.

The generic model of each of these components should include:

1. its use-mode automaton,
2. the list of services associated with each use-mode.

Valves. Consider first the valves V_i, $i = 1, ...4$, and assume they are all described by the same model, with the use-mode set = $\{V_i$_off, V_i_maintenance, V_i_automatic$\}$, associated with the service lists:

- *V_i_off: V_i_to_maintenance, V_i_to_automatic*
- *V_i_maintenance: V_i_open, V_i_close, V_i_open_manual, V_i_close_manual, V_i_to_off, V_i_ to_automatic*
- *V_i_automatic: V_i_open, V_i_close, V_i_to_off, V_i_to_maintenance*

Since only the automatic behaviour will be of interest for the valves as well as for the other components, the partial model associated with the automatic operation mode, whose services are *{V_i_ open, V_i_ close}* is the only one which will be considered.

The definition of these services in terms of consumed, produced variables and procedure is as follows, where q_i is the flow through valve i, Δp_i is the pressure drop between the input and output of the valve, and k_i is a parameter.

Service	Consumed	Produced	Procedure
V_i_open :	Δp_i	q_i	$q_i = k_i \text{ sign } (\Delta p_i)\sqrt{\lvert\Delta p_i\rvert}$
V_i_close :	Δp_i	q_i	$q_i = 0, \forall \Delta p_i$

Pumps. Each pump $P_i, i = 1, 2$ can provide the single service $deliver\_Q_i$ where Q_i is the flow parameter associated with the request for the "deliver" service, and q_i is the flow really delivered.

Service	Consumed	Produced	Procedure
$deliver\_Q_i$	Q_i	q_i	$q_i = Q_i$

Tanks. Each tank $T_i, i = 1, 2, 3$ can provide the service T_i_store, where l_i is the level in tank i , Δq_i is the difference between the input and output flows in the tank, $l_i^{\max}$ and a_i are parameters.

Service	Consumed	Produced	Procedure
T_i_store	Δq_i	l_i	$l_i(t) = \min\left\{\max\left\{0, a_i \int \Delta q_i(t)dt\right\}, l_i^{\max}\right\}$

Sensors. Each sensor $L_i, i = 1, 2, 3$ can provide the service $level\_value_i$, where l_i is the true level in tank i, h_i is its measured value (estimated by the sensor) and g is a given function.

Service	Consumed	Produced	Procedure
$level\_value_i$	l_i	h_i	$h_i = g(l_i)$

Controllers. Controller C_1 produces the Q_i parameters of the pumps. Two services are provided, namely *max_flow* which delivers the control signal for the maximum flow and *regul_flow* which delivers the control signal for a PI regulated flow, where $Q_i^{\max}$ is the maximum value of the flow which can be requested from pump P_i, w_i is the reference level for the PI controller, and K_{Pi}, (respectively K_{Ii}) are the proportional (respectively integral) coefficients of the PI regulator.

Service	Consumed	Produced	Procedure
max_flow_i :	–	Q_i	$Q_i = Q_i^{\max}$
$regul\_flow_i$:	h_i, w_i	Q_i	$Q_i = K_{Pi}(h_i - w_i) + K_{Ii} \int (h_i - w_i)dt$

Controller C_2 provides the service $control\_V_i$, which is associated with an on/off regulation, and which requests the V_i_open and V_i_close services of the valves, where h_i^-, h_i^+ are the min/max level values associated with the on/off regulator, v_i is the control request to valve i ($v_i \in \{open, close\}$) .

Service	Consumed	Produced	Procedure
$control\_V_i:$	h_i, h_i^-, h_i^+	v_i	$h_i \leq h_i^- \Longrightarrow v_i = open$ $h_i \geq h_i^+ \Longrightarrow v_i = close$

Use-modes and objectives. The definition of the use-mode set of the overall system, and for each use-mode the definition of its associated objectives directly result from the specification analysis. For the three-tank system, the different objectives are the following:

- Objective 0: No action
- Objective 1: Reach the level set points as fast as possible
- Objective 2: Regulate the levels to the set points
- Objective 3: Completely empty the system
- Objective 4: Protect the environment.

The following table gives the different use-modes and the associated objectives.

Number of use-mode	Name	Objectives
0	No_operation	0
1	Preparation	1, 4
2	Regulation	2, 4
3	End_of_production	3, 4
4	Fall_back	3

The associated use-mode management graph is given on Fig. 10.2.

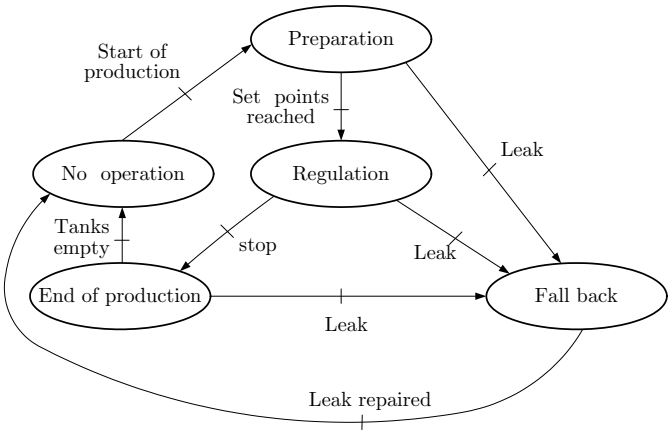


Fig. 10.3. Use-mode management graph

High-level services. Objectives are achieved from the services provided by the system elementary components. Figure 10.4 gives the pyramidal decomposition of the three-tank system, which is decomposed into three subsystems, each of them constituted of one tank and its instrumentation.

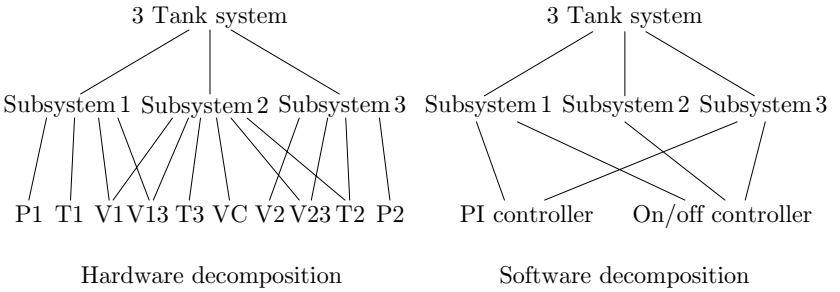


Fig. 10.4. Pyramidal decomposition

The services of the elementary components define "instructions" (a basic vocabulary) with which "programs" (words formed on this vocabulary, using given connectors) can be written to deliver services of higher level. The feasible combinations of elementary services which provide the subsystem 1 (respectively subsystem 3) services are given by the first column of table 1 (respectively table 2) where the writing

$$\text{S1.2} = \{T_1\_store, deliver\_Q_1, V_1\_close, V_{13}\_open\}$$

means that the four elementary services in the brackets are needed to provide the high-level service: "decrease level in Tank T1". Note that, according to the control architecture, this high-level service could be implemented in different ways, for example:

$$T_1\_store \;//\; deliver\_Q_1 \;//\; V_1\_close \;//\; V_{13}\_open$$

(where // means the parallel execution connector) if the pump and the valves are intelligent actuators with their own processing unit, or

$$T_1\_store \;//\; PV$$

where PV is defined by the program:

do

$deliver\_Q_1$
V_1_close
V_{13}_open

end_do

which corresponds to the "word" $deliver\_Q_1/V_1\_close/V_{13}\_open$, where / is the sequencial execution connector, if P_1, V_1, V_{13} are controlled by the same processing unit. The corresponding functional interpretations in the case where the services are

run from the regulation nominal conditions are given by the second column of the tables.

Table 10.2 Service of subsystem 1: A functional interpretation for $h_1 = 0.5$ m, and $h_3 = 0.1$ m

Feasible combination	Functional interpretation
$S1.1 =$ $\{T_1\_store, deliver\_Q_1, V_1\_close, V_{13}\_close\}$	if $Q_1 \neq 0$ increase level 1, else keep level 1 constant
$S1.2 =$ $\{T_1\_store, deliver\_Q_1, V_1\_close, V_{13}\_open\}$	decrease level 1
$S1.3 =$ $\{T_1\_store, deliver\_Q_1, V_1\_open, V_{13}\_close\}$	decrease level 1
$S1.4 =$ $\{T_1\_store, deliver\_Q_1, V_1\_open, V_{13}\_open\}$	decrease level 1

Table 10.3 Service of subsystem 3: A functional interpretation for $h_1 = 0.5$ m, and $h_3 = 0.1$ m

Feasible combination	Functional interpretation
$S3.1 = \{T_3\_store, V_1\_close, V_{13}\_close\}$	decrease level 3
$S3.2 = \{T_3\_store, V_1\_close, V_{13}\_open\}$	increase level 3
$S3.3 = \{T_3\_store, V_1\_open, V_{13}\_close\}$	increase level 3
$S3.4 = \{T_3\_store, V_1\_open, V_{13}\_open\}$	increase level 3

Services of subsystems 1 and 3 can themselves be associated to provide services at the system level. As in the previous step, feasible associations are automatically generated by the exploration of the possible service combinations taking care that the choice of a Table 10.3 service may impose the choice of a Table 10.4 service, since a part of the elementary services implied in the aggregated one are the same. For example, a service which makes use of V_1_open cannot be run simultaneously with a service which makes use of V_1_close. Feasible combinations of group 1 and 3 services to provide system services are given by the first column of Table 10.4. The corresponding functional interpretation in the case where the service is run from the regulation nominal conditions is given by the second column.

Table 10.4 High-level services: A functional interpretation for $h_1 = 0.5$ m and $h_3 = 0.1$ m

High-level service	Functional Interpretation
$S0.1 = \{S1.1, S3.1\}$	increase or keep level 1, decrease level 3
$S0.2 = \{S1.2, S3.2\}$	decrease level 1, increase level 3
$S0.3 = \{S1.3, S3.3\}$	decrease level 1, increase level 3
$S0.4 = \{S1.4, S3.4\}$	decrease level 1, increase level 3

Table 10.4 shows that there is no service which allows to keep Tank 1 and Tank 3 levels constant. Consequently the regulation objective can only be performed by requesting successively level-increasing and level-decreasing services. The regulation objective achievement requires the level set points w_1, h_3^-, h_3^+ as input and can be provided by different versions, which are given by

$$
\begin{aligned}
M2.1 &= \{S0.1, S0.3, level\_value_3, level\_value_1, regul\_flow_1, control\_V_1\} \\
M2.2 &= \{S0.1, S0.2, level\_value_3, level\_value_1, regul\_flow_1, control\_V_{13}\} \\
M2.3 &= \{S0.1, S0.4, level\_value_3, level\_value_1, regul\_flow_1, control\_V_1, \\
&\quad control\_V_{13}\}
\end{aligned}
$$

The algorithm which realises service $M2.1$ could be, for example:

Algorithm 10.1 *M2.1 algorithm*

Inputs: w_1, h_3^-, h_3^+

Do: Until end of regulation service.

1. $regul\_flow\_Q_1$: $Q_1 = f(w_1, h_1)$.
2. $control\_V_1$: $v_1 = f(h_3, h_3^-, h_3^+)$.
3. if $v_1 = open$ then S0.3 else S0.1.

Faults scenarios. When faults occur, some lower level services become unavailable or become permanent in time. The available versions of the high-level services are those which do not require the lost low-level services and in which the permanent low-level services are implied. Let us consider three examples.

Scenario 10.1 The current operation mode is UM2, and the currently used version for achieving objective 2 is the nominal one M2.1. Suppose that Valve V_1 gets blocked closed. The analysis is as follows:

- Service V_1_close gets permanent in time and service V_1_open becomes unavailable. Therefore, services S1.3, S1.4, S3.3, and S3.4 become unavailable, which implies the unavailability of services S0.3 and S0.4.

- There exists one version, namely M2.2, which can be run to achieve the regulation objective, since it does not make use of any unavailable service. The nominal version can no longer be provided but objective 2 can still be achieved, thanks to service reconfiguration.

Scenario 10.2 The current operation mode is UM2, and the currently used version for achieving objective 2 is the nominal one M2.1. Suppose that Valve V_1 gets blocked open. The analysis is as follows:

- Service V_1_open gets permanent in time and service V_1_close becomes unavailable. Therefore, services S1.1, S1.2, S3.1, S3.2 become unavailable, which implies the unavailability of services S0.1, S0.2.
- The nominal version for achieving objective 2 can no longer be provided, and no other version can be performed, since service S0.1 is common to all versions. Level 1 cannot be kept to $0.5\,\mathrm{m}$, and it makes no sense to stay in UM2 any longer. A possible solution is to move to another use-mode whose missions both do not contain the regulation one and can be fulfiled using the reduced set of remaining services. Another possible solution is to change the original mission parameters so as to make success possible, e. g. change level 1 set point from w_1 to w_1^*, where w_1^* is the level of the valve V_1 connecting pipe. With this new objective, Tables 10.2, 10.3, 10.4 becomes Tables 10.5, 10.6, 10.7 (only the functional interpretation changes) and level 3 can be regulated using one of the versions given by

$$M2.1 = \{S0.3, S0.4, level\_value_3, level\_value_1, regul\_flow\_Q_1, control\_V_{13}\}.$$

Table 10.5 Service of subsystem 1: A functional interpretation for $h_1 = w_1^*$ and $h_3 = 0.1\,\mathrm{m}$

Feasible combination	Functional interpretation
$S1.3 =$ $\{T_1\_store, deliver\_Q_1, V_1\_open, V_{13}\_close\}$	if $Q_1 \neq 0$ increase level 1, else keep level 1 constant
$S1.4 =$ $\{T_1\_store, deliver\_Q_1, V_1\_open, V_{13}\_open\}$	decrease level 1

Table 10.6 Service of subsystem 3: A functional interpretation for $h_1 = w_1^*$ and $h_3 = 0.1\,\mathrm{m}$

Feasible combination	Functional interpretation
$S3.3 = \{T_3\_store, V_1\_open, V_{13}\_close\}$	decrease level 3
$S3.4 = \{T_3\_store, V_1\_open, V_{13}\_open\}$	increase level 3

Table 10.7 High-level services: A functional interpretation for $h_1 = w_1^*$ and $h_3 = 0.1$ m

Feasible combination	Functional interpretation
$S0.3 = \{S1.3, S3.3\}$	decrease level 1, increase level 3
$S0.4 = \{S1.4, S3.4\}$	decrease level 1, increase level 3

Scenario 10.3 There is a leak in Tank T_1. The corresponding storage service becomes unavailable and the environment protection objective can no longer be fulfilled. The system has to be moved to an use-mode in which this objective does not appear, namely the fall back use-mode. In this use-mode, achieving objective 3 leads to completely empty the tanks.

As a remark, it should be noted that for the combination of Tanks T_2 and T_3 to appear as a redundant hardware allowing further reconfiguration, objective 2 should have been formulated as: "regulate levels 1 and 3 or levels 2 and 3 to the set points".

10.1.3 Solution of the reconfiguration task

The reconfiguration problem of the three-tank system includes discrete decisions that have to be made concerning the choice of the actuators, the sensors, the controller and the set-points. Therefore, it is reasonable to use a representation of the three-tank system which refers directly to these decision variables. The method presented in Section 9.7 will be applied here, where the non-deterministic automaton is abstracted from a discrete-time version of the continuous-variable model (10.2) – (10.4).

Partitioning of the signal spaces. The quantiser of the level h_2 is already given in the problem formulation, where this level is only known to assume one of the three qualitative values *low*, *medium* or *high*. The quantisers for h_1 and h_3 are introduced deliberately, because the decision concerning the reconfiguration of the controller does, in general, not depend on the precise quantitative value $\boldsymbol{x}$ but on a global assessment $[\boldsymbol{x}]$ of the state. Hence, the signal spaces of the tank levels are partitioned into the six intervals described in the following table.

$[h_i] = empty$	$h_i \in [0, 0.09\text{ m})$
$[h_i] = low$	$h_i \in [0.09\text{ m}, 0.11\text{ m})$
$[h_i] = medium$	$h_i \in [0.11\text{ m}, 0.49\text{ m})$
$[h_i] = high$	$h_i \in [0.49\text{ m}, 0.51\text{ m})$
$[h_i] = full$	$h_i \in [0.51\text{ m}, 0.60\text{ m})$
$[h_i] = overflow$	$h_i \geq 0.6\text{ m}$

Instead of the tank levels h_i only the qualitative values $[h_i]$ are assumed to be known, which form the vector

$$[\boldsymbol{x}(k)] = ([h_1(k)],\ \ [h_3(k)],\ \ [h_2(k)])'.$$

For the valves, the qualitative values *closed* and *open* correspond to the quantitative values $Pos(V) = 0$ or $Pos(V) = 1$, respectively. The set point h_1^{ref} is assumed to have one of the qualitative values $[h_1]$ of the level of Tank T_1 and the pump P_2 is assumed to have three qualitative values as shown in the following table:

$[V_i] = closed$	$Pos(V_i) = 0$
$[V_i] = open$	$Pos(V_i) = 1$
$[Q_2^{P_2}] = off$	$Q_2^{P_2} = 0$
$[Q_2^{P_2}] = medium$	$Q_2^{P_2} = 0.5\, Q_{\max}$
$[Q_2^{P_2}] = on$	$Q_2^{P_2} = Q_{\max}$

The discrete input vector $[\boldsymbol{u}(k)]$ is composed of the qualitative input values similarly as the qualitative state $[\boldsymbol{x}(k)]$.

Qualitative modelling of the tank system. A model (9.38) of the three-tank system subject to the three faults considered can be obtained by applying the abstraction method developed in Section 9.4.3. For the reconfiguration purposes, a nondeterministic automaton is used rather than a stochastic automaton. The automaton takes into account all three tanks, because the right tank has to be used in case of the fault f_3. As the levels in Tanks T_1 and T_3 are quantised into 6 intervals each and the level of the middle tank into three intervals, the automaton has $6 \cdot 3 \cdot 6 = 102$ states and cannot be shown here.

After the qualitative model has been obtained by the abstraction procedure, the controller has been found by Algorithm 9.3. With the requirements given for the three-tank system, the set $\mathcal{Z}_{\mathrm{Aim}}(f)$ of admissible operation points is given by

$$\mathcal{Z}_{\mathrm{Aim}}(f) = \{[\boldsymbol{x}] \ : \text{Eqs. (10.7) and (10.8) are satisfied}\}.$$

For fault f_1,

$$\mathcal{Z}_{\mathrm{Aim}}(f_1) = \{([h_1], medium, [h_3])' \text{ with arbitrary } [h_1], [h_3]\}$$

holds. The function k_q can be represented as a tabular showing the relation among the different faults f, the qualitative state $[\boldsymbol{x}]$ and the qualitative input $[\boldsymbol{u}]$.

Experimental results. Experiments with the three-tank system have been made with sampling time $T_s = 7\,\mathrm{s}$ and with the quantiser described above. $[\boldsymbol{x}(k)]$ is measured by capacity sensors that indicate merely whether the liquid level in the tank is above or below the sensor position. At time $k = 0$ the fault-tolerant control algorithm is informed about the current fault f, which has been applied to the tank system earlier and which has brought the tank levels to "wrong" values. Then the computer selects the control input according to the control law k_q.

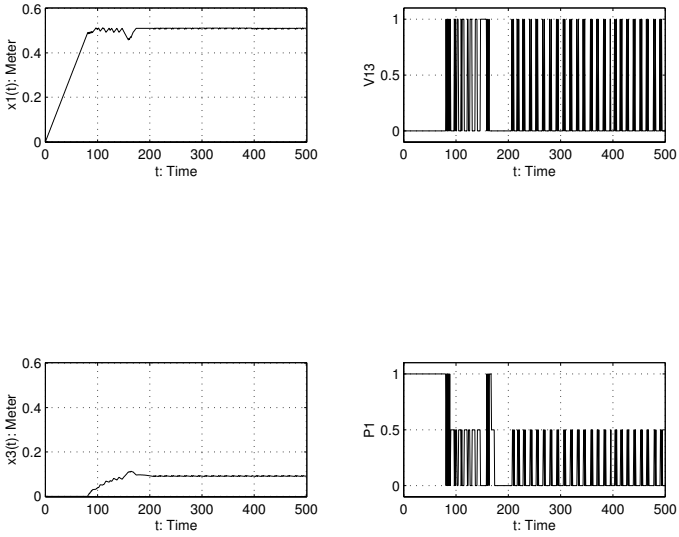


Fig. 10.5. Behaviour of the reconfigured system for fault f_1

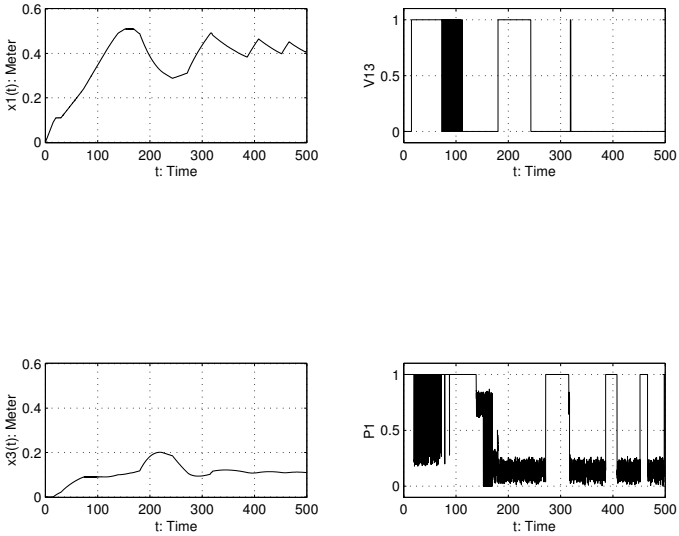


Fig. 10.6. Behaviour of the reconfigured system for fault f_2

The experimental results are shown in Fig. 10.5 for fault f_1 and the initial state $\boldsymbol{x}_0 = \mathbf{0}$, which corresponds to the extreme assumption that after the fault has occurred, the tank system is emptied until the fault has been identified. This extreme assumption is made to show the effect of the discrete controller, which works unter the influence of the fault. The controller uses the valve V_{12L} as new control input and brings the system into the required state within about $170\,\mathrm{s}$. In the experiment shown in Fig. 10.6, Fault f_2 occurred. The controller reduces the set point of the level controller of Tank T_1 to $[h_1^{\mathrm{ref}}] = medium$ and uses again the valve V_{12L} to bring the level of Tank T_2 to the required value $medium$.

10.2 Diagnosis and fault-tolerant control of a chemical process

In this section, fault diagnosis and fault-tolerant control are applied to two chemical processes, where in the first case a fault-tolerant temperature and level controller should be applied whereas in the aim of the second case is to attenuate disturbances concerning the conductivity and temperature of a liquid. Both problems are tackled by means of linearised models. The experiments with industrial equipment show impressive results with the methods developed in this book but also point to the restrictions of the fault tolerance if the process diverges considerably from the operation point and the nonlinearities shift the plant properties from the nominal ones.

10.2.1 Fault diagnosis by means of a discrete-event model

The experimental set-up used for the test of qualitative diagnostic methods is depicted in Figs. 10.7 and 10.8. Although the main modelling problems are posed by the stirred reactor, which is depicted in the middle of Fig. 10.9, the faults affect other parts of the whole system as well.

Figure 10.9 shows the part of the process that is considered further. An inflowing liquid is heated in the stirred tank reactor such that the outflowing liquid has a temperature of approximately $70^\circ\,\mathrm{C}$. The temperature is controlled by a dual-mode controller switching both heating elements simultaneously on or off such that the temperature in the reactor is held between 69 and $71^\circ\,\mathrm{C}$. The liquid level in the tank is controlled by means of a dual-mode controller that opens the outlet valve half or completely such that the level in the reactor varies between 30 and $40\,\mathrm{cm}$. In Fig. 10.13, the faultless behaviour of the process is depicted. The valves V_{20} and V_{90} are additional inputs which are not used under faultless conditions but may be used in fault-tolerant control. With these valves, water of temperature $20^o\,\mathrm{C}$ or $90^o\,\mathrm{C}$, respectively, can be put into the reactor.

Faults. In the following, three faults will be considered, which all block the controllers and, hence, necessitate a reconfiguration of the control algorithm:

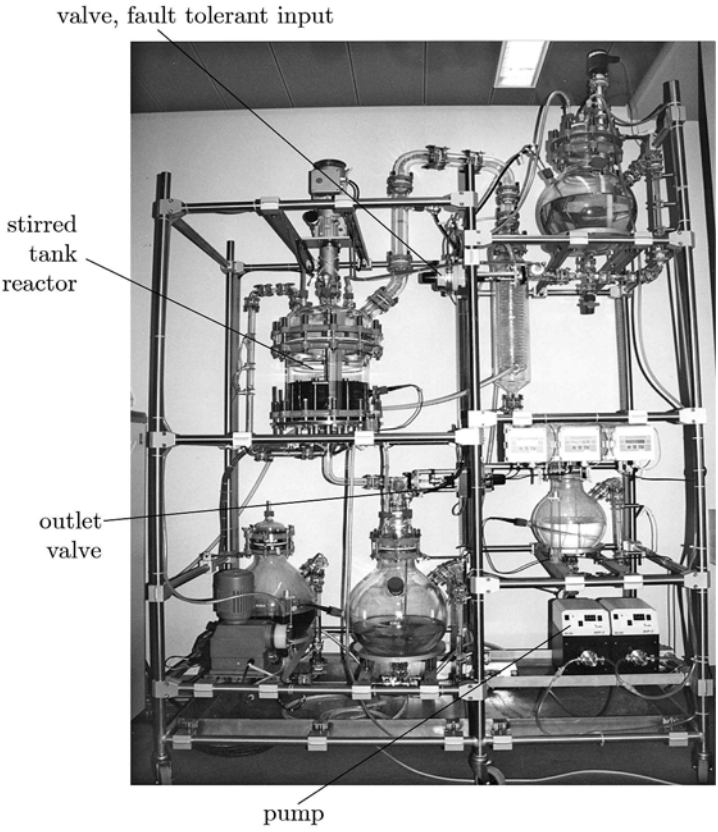


Fig. 10.7. The chemical plant TINA

- **Heating element fault.** If a single heating element ceases to operate, the heating power is insufficient to maintain a reactor temperature of 70° C.

- **Valve fault.** If the outlet valve is stuck in the completely opened position, after some time, the liquid level falls below 20 cm which is below the top of the heating elements and therefore causes a safety mechanism to turn off the heating and to deactivate the temperature control. Fault-tolerant control has to prevent the safety system from shutting off the plant.

- **Cooler fault.** The temperature T_{in} of the inflowing liquid is the output of another process including a cooler, which may fail. Under normal conditions, the temperature is 23° C. In case of the cooler fault, the temperature of the inflowing liquid rises to 90° C such that cooling rather than heating in the stirred reactor is necessary.

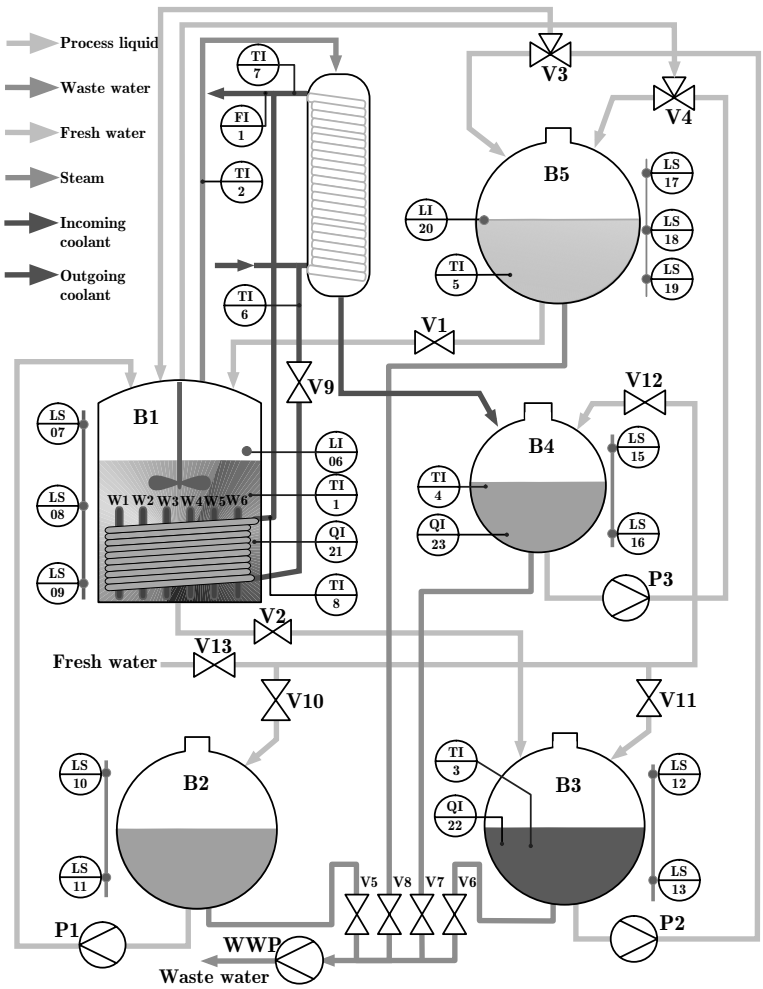


Fig. 10.8. Schematic diagram of the overall process

For the plant considered here, the first two faults are internal faults whereas the third fault is an external fault (cf. Chapter 3).

Plant model for diagnosis. For the demonstration of the diagnostic algorithm, the system under consideration is the stirred tank reactor combined with two dual-mode controllers. As the block diagram depicted in Fig. 10.10 shows, the plant has the four input signals T_{in}, $Pump$, V_{20} and V_{90}, whose interpretations are also given in Fig. 10.9. The dual-mode controller is considered as a part of the system whose faults should be diagnosed. The measured output is the temperature T and

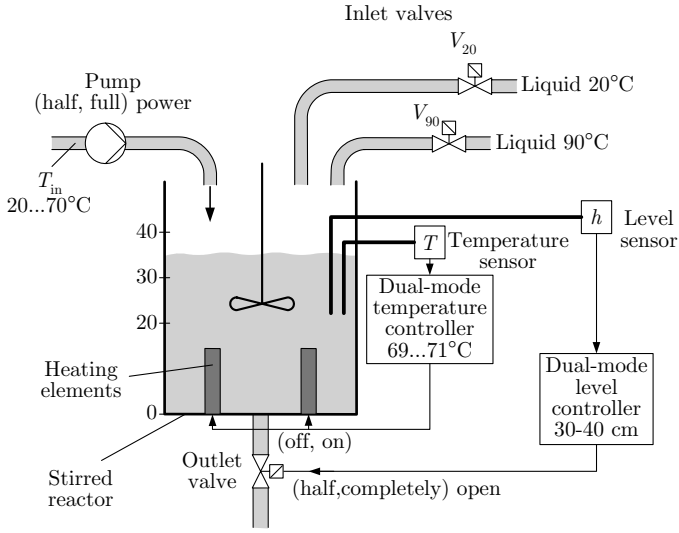


Fig. 10.9. Part of the process used for the experiments

the liquid level h of the reactor. The three faults are considered as additional inputs to the system.

The chemical plant has continuous as well as discrete signals. For example, the temperatures are continuous whereas the input signals generated by the dual-mode controllers are discrete. Therefore, the quantised systems approach explained in Chapter 9 is useful, because all signals are interpreted uniformly as discrete signals.

A state-space model of the reactor can be derived with the liquid level and the temperature as the two state variables. Two differential equations occur, where the first is analogous to the equation used in the tank example throughout this book and the second results from an enthalpy balance. These two differential equations have been combined with the switching conditions of the dual-mode controllers, which results in a hybrid model of the system to be diagnosed. Each dual-mode controller has two discrete state variables, which are considered to be immeasurable and should be observed. The overall state space consists of two continuous and two discrete state variables.

The qualitative model is abstracted from a quantitative model for a sampling time of $T_s = 120\,\mathrm{s}$. The liquid temperature and the level are the output signals. For the qualitative model, the partitioning of the continuous-variable subspace, which is identical to the partitioning of the output space, is depicted in Fig. 10.11. The size of the state sets around the set-point of $70^\circ\mathrm{C}$ is chosen smaller than in the other regions of the state space. Likewise, the input space used by the controller of the faultless system is partitioned. The additional input signals, which may be used by the fault-tolerant controller, are the positions of two valves V_{20} and V_{90}, which can be completely opened or closed, only.

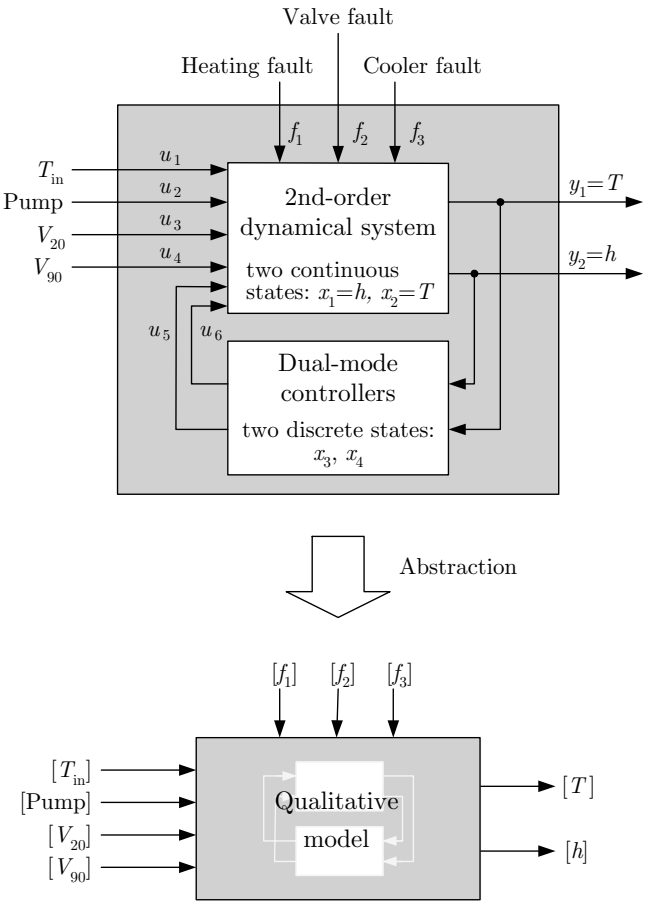


Fig. 10.10. Abstraction of the qualitative model

The result is a stochastic automaton with 400 states.

Experimental results for state observation. The observation algorithm described in Section 9.5.3 is applied to determine the discrete state of the dual-mode controllers. These discrete states coincide with the control input generated by the controllers, which switch the heating on and off and determine the position of the outlet valve. The reactor temperature $T(t)$ and the liquid level $h(t)$ are used as output measurements. The task considered here is to determine the dual-mode controller states from the quantised measurement information.

The input and output signals measured in an experiment with the faultless plant are depicted in Figs. 10.12 and 10.13. The first figure shows the pump power $Pump(t)$ and the input temperature $T_{\text{in}}(t)$. The two other input signals shown in Fig. 10.10 are constant. The measured reactor temperature T and liquid level h are

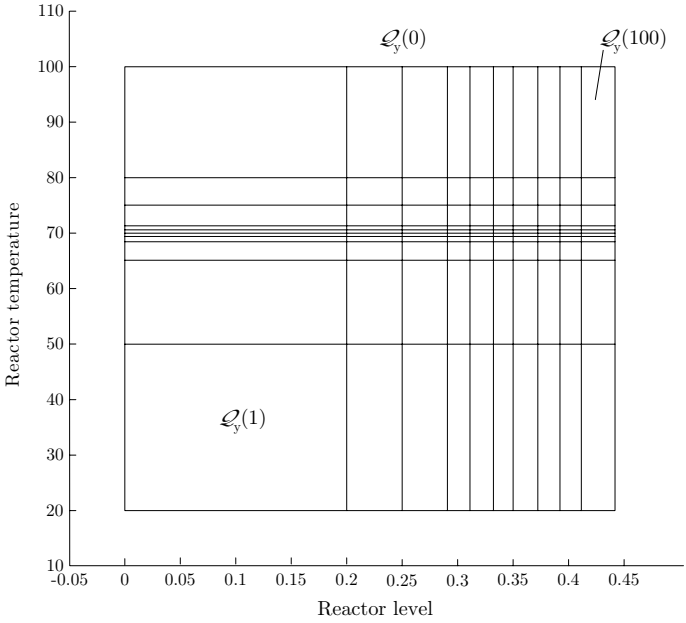


Fig. 10.11. Partitioning of the state space

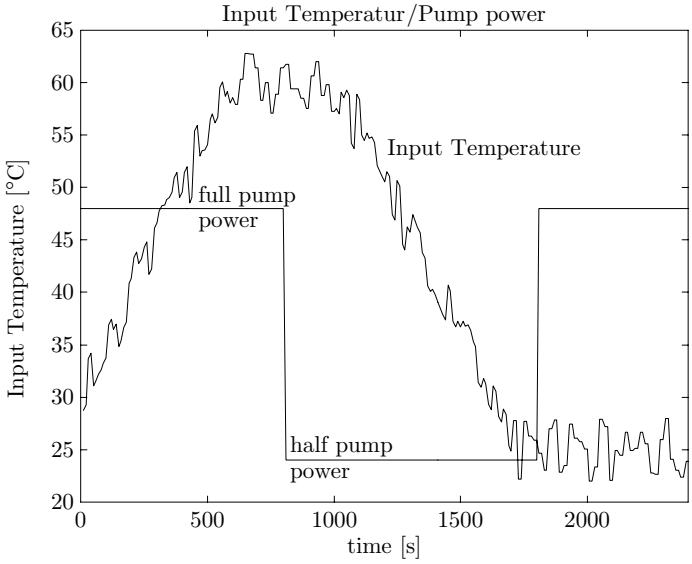


Fig. 10.12. Process input

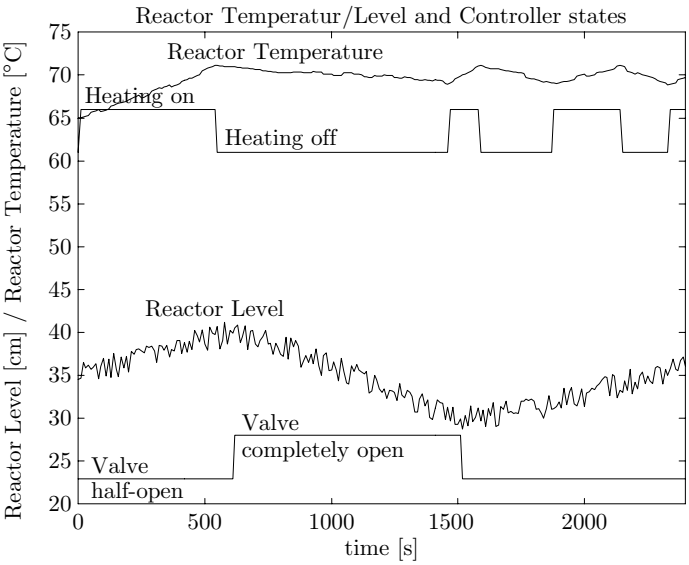


Fig. 10.13. Output measurements

depicted in Fig. 10.13. This figure also shows the true discrete states of the dual-model controllers, which correspond to the heating switching and the valve position. These discrete states have been assumed immeasurable and, therefore, to be reconstructed by the observation algorithm.

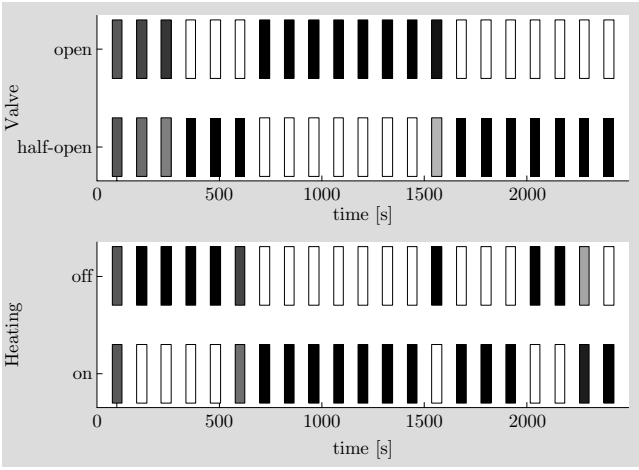


Fig. 10.14. Qualitative observation result

The observation result is shown in Fig. 10.14. The probabilities of the dual-mode controller states are depicted in grey scale. A dark colour represents a high and a light a low probability. All measurement information has been quantised according to the chosen signal space partitions before they are processed by the observation algorithm.

The grey rectangles for the initial time point show that the discrete controller states cannot be determined from the measurement obtained for the first time instant alone. The algorithm has been initialised with a uniform distribution over the qualitative states. At the second time step the heating position can be uniquely determined to be "off", whereas three quantised measurements are necessary to determine the valve position to be "half-open".

The quality of the observation can be evaluated by comparing the observation result with the true results shown in Fig. 10.13. It can be seen, that the observation algorithm provides a good approximation of the discrete controller states. This state observation is incorporated into the diagnostic algorithm described in the following.

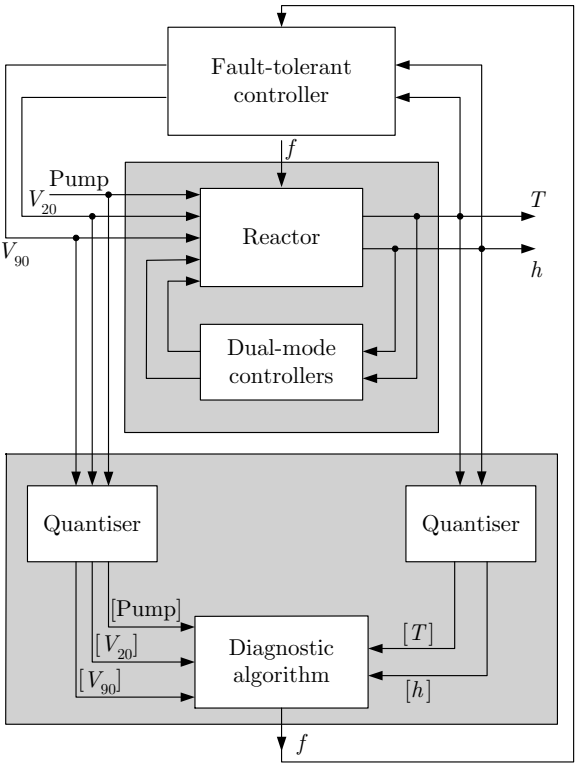


Fig. 10.15. Fault diagnosis of the reactor

Diagnostic results. The diagnostic algorithm developed in Section 9.6 is applied. As a heating fault means that the temperature control loop is opened and, hence, the control aim can no longer be met, the reactor was equipped with a fault-tolerant controller that closes a loop around the system considered so far. It uses the reactor temperature and liquid level as system output and the valves V_{20} and V_{90} as control input (Fig. 10.15). The controller gets the diagnostic result as further information. The additional controller is used here to hold the reactor in a long time interval inside its region of acceptable performance. The experiment is made to demonstrate the diagnostic algorithm.

As the diagnosis also concerns the cooler fault, the input temperature T_{in} is not used as measurement.

Figure 10.16 shows the experimental results. The upper part of the figure includes the immeasurable input temperature T_{in} and the pump power $Pump$ and the middle part depicts the reactor temperature T and the liquid level h together with the input signals V_{20} and V_{90} generated by the fault-tolerant controller, which bring about an additional cold liquid inflow or hot liquid inflow, respectively.

The first fault is a break-down of a heating element at time 800 seconds. As the second fault, at time 2700 seconds, the cooler breaks down, which leads to the increasing temperature depicted in the upper part of the figure.

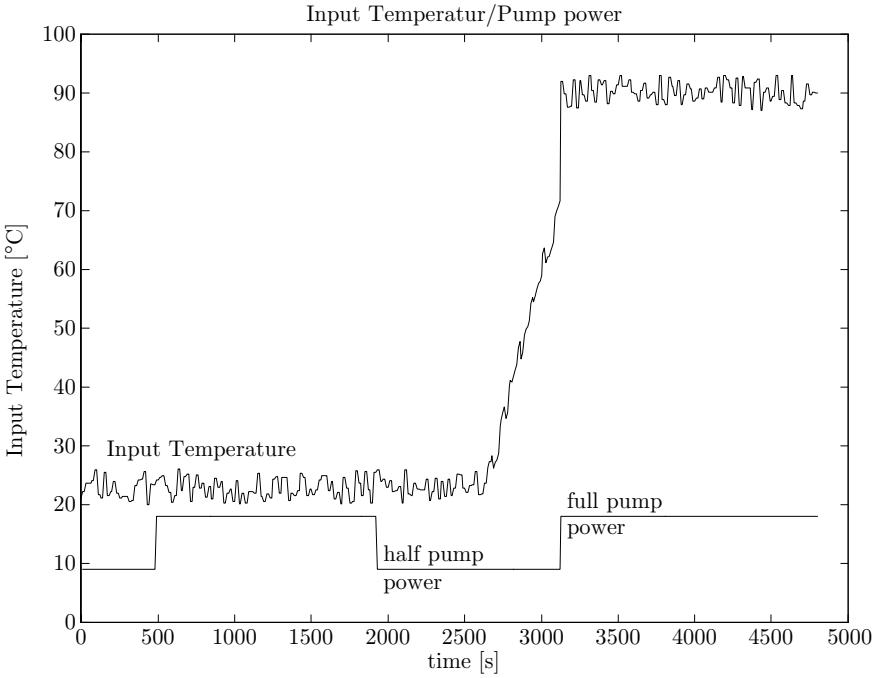
Figure 10.17 shows the diagnostic result. The algorithm is started in the start-up phase of the process, when the reactor temperature was still too low and the fault-tolerant controller opens the valve V_{90} to let hot liquid into the reactor to increase the temperature. The diagnostic algorithm provides the fault probabilities, which are also depicted in Fig. 10.17. It can be seen, that both faults, which also occur in combination, are detected quickly and uniquely, and the valve fault is explicitly excluded.

After approximately ten minutes, the heating element fault occurs. It can be seen in Fig. 10.17 that the diagnostic algorithm finds this fault at once. To prevent the reactor from leaving its region of acceptable performance, the fault-tolerant controller opens the valve V_{90} to let hot liquid into the reactor in order to stabilise the temperature despite of the reduced heating power. As a result, the experiment can be continued.

After one hour, in addition to the heating element fault, a cooler fault occurs. The input temperature starts to rise to $90°C$. Note that the diagnostic algorithm is not supplied with the depicted information about the temperature, but assumes that the input temperature is low (i.e. between 20 and $30°C$). However, the diagnostic algorithm detects the fault (Fig. 10.17). By using this information, the fault-tolerant controller starts adding cold liquid by opening valve V_{20} and thus returns the process into the region of acceptable performance.

10.2.2 Reconfiguration of a level and temperature control loop

For a demonstration of the control reconfiguration in case of an actuator failure the part of the chemical process shown in Fig. 10.18 is considered. The control



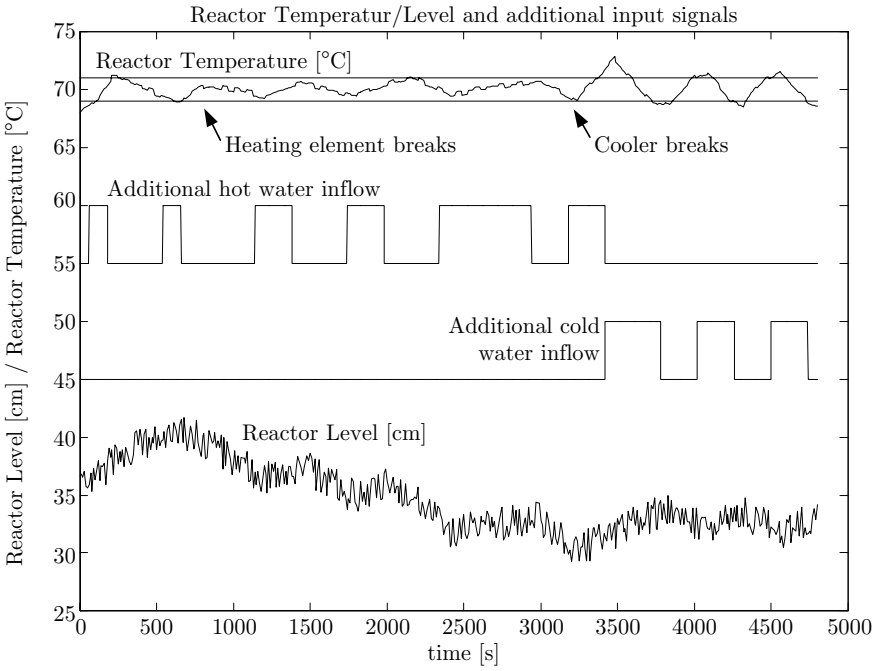


Fig. 10.16. Process input generated by the fault-tolerant controller and output measurements

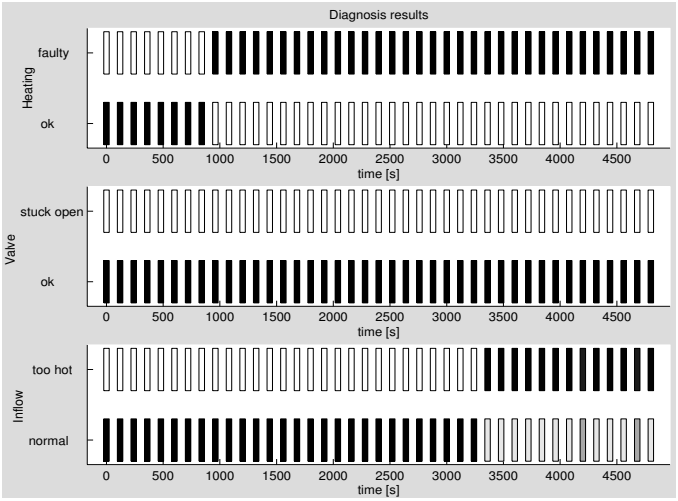


Fig. 10.17. Diagnostic result

objectives are to maintain a constant liquid level and a constant temperature in the reactor tank B_1 and, thus, producing a constant product outflow. To achieve this, hot and cold liquid can be brought into the reactor from Tanks B_2 and B_5. The main reactor B_1 can be heated and cooled.

In the nominal case the liquid level is controlled by adjusting the cold liquid inflow from Tank B_5 and the temperature by means of the heating.

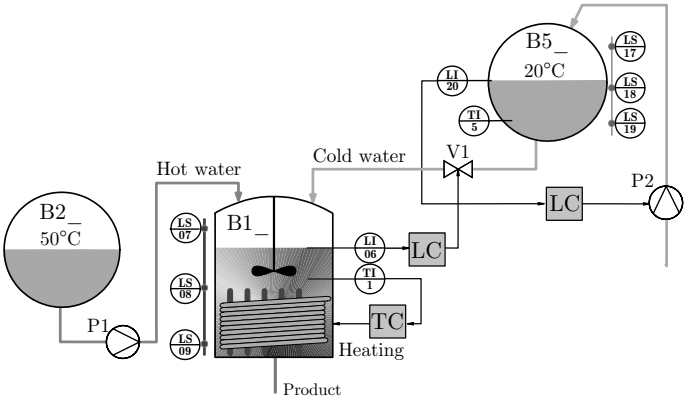


Fig. 10.18. Plant used for control reconfiguration (LC - level control, TC - temperature control

Plant model. The plant model contains three states: the reactor content V_{B1}, the reactor temperature ϑ_{B1} and the content of the cold liquid tank V_{B5}. From a mass balance, the following equations are obtained

$$\begin{aligned}
\dot{V}_{B5}(t) &= k_{P2}u_{P2}(t) - q_{51}(t) \\
\dot{V}_{B1}(t) &= q_{21}(t) + q_{51}(t) - q_{1out}(t) \\
\dot{\vartheta}_{B1}(t) &= (\vartheta_{B2}(t) - \vartheta_{B1}(t))\frac{q_{21}(t)}{V_{B1}(t)} + (\vartheta_{B5}(t) - \vartheta_{B1}(t))\frac{q_{51}(t)}{V_{B1}(t)} \\
&\quad + \frac{u_{heat}(t)k_{heat}}{V_{B1}(t)},
\end{aligned}$$

where for the liquid flows the relations

$$\begin{aligned}
q_{21}(t) &= k_{P1}u_{P1}(t) \\
q_{51}(t) &= k_{V1}\,124.5^{u_{V1}(t)}\sqrt{h_{B5}(t) + 1.07} \\
q_{1out}(t) &= k_{V2}\sqrt{\frac{V_{B1}(t)}{A_{B1}} + 1.4}
\end{aligned}$$

hold. $h_{B5}(t)$ is the liquid level in the spherical tank B_5, $u_{heat}(t)$ the heating power, k_{heat} a heating coefficient, $u_{P1}(t)$, $u_{P2}(t)$ and $u_{V1}(t)$ the control input to the two pumps and to the Valve V_1 and A_{B1} the cross-section area of the Tank B_1. After a linearisation of this nonlinear model around the operating point of $\vartheta_{B1} = 40^o\mathrm{C}$, the following linear model is obtained:

$$\begin{aligned}
\begin{pmatrix} \dot{V}_{B5}(t) \\ \dot{V}_{B1}(t) \\ \dot{\vartheta}_{B1}(t) \end{pmatrix} &= 10^{-3}\begin{pmatrix} -0.46 & 0 & 0 \\ +0.46 & -0.33 & 0 \\ -0.48 & 0.008 & -1.1 \end{pmatrix}\begin{pmatrix} V_{B5}(t) \\ V_{B1}(t) \\ \vartheta_{B1}(t) \end{pmatrix} \\
&\quad + \begin{pmatrix} 0.09 & -0.023 & 0 & 0 \\ 0 & +0.023 & +0.05 & 0 \\ 0 & -0.024 & +0.02 & 0.223 \end{pmatrix}\begin{pmatrix} u_{P2}(t) \\ u_{V1}(t) \\ u_{P1}(t) \\ u_{heat}(t) \end{pmatrix} \\
\boldsymbol{y} &= \begin{pmatrix} h_{B5}(t) \\ h_{B1}(t) \\ \vartheta_{B1}(t) \end{pmatrix}.
\end{aligned}$$

The nominal proportional controllers are defined by:

$$\begin{aligned}
u_{V1}(t) &= -0.5\,V_{B1}(t) \\
u_{heat}(t) &= -0.5\,\vartheta_{B1}(t) \\
u_{P2}(t) &= -1\,V_{B5}(t).
\end{aligned}$$

They can be represented as

$$\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{y}(t)$$

with

$$\boldsymbol{K} = \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.5 \end{pmatrix}.$$

Note that these controllers do not use the control input u_{P1}, because the matrix $\boldsymbol{K}$ has a vanishing third row.

Faults. Several severe faults can occur that open the control loops. For example, due to a heating failure, the reactor can no longer be heated, or clogging or blockage of Valve V_1 can bring the level controller out of operation. In the following the heating failure and a blockage of Valve V_1 in its nominal position will be considered.

Controller reconfiguration after a heating failure. After a heating failure has occurred, the temperature controller

$$u_{heat}(t) = -0.5\,\vartheta_{B1}(t)$$

has no influence on the process. The system in the nominal and the faulty case has the matrices

$$\begin{aligned} \boldsymbol{B} &= \begin{pmatrix} 0.09 & -0.023 & 0 & 0 \\ 0 & +0.023 & +0.05 & 0 \\ 0 & -0.024 & +0.02 & 0.223 \end{pmatrix} \\ \boldsymbol{B}_f &= \begin{pmatrix} 0.09 & -0.023 & 0 & 0 \\ 0 & +0.023 & +0.05 & 0 \\ 0 & -0.024 & +0.02 & 0 \end{pmatrix}, \end{aligned}$$

which distinguish in the last column. Both matrices have the same rank and can be related to one another by the matrix

$$\boldsymbol{N} = \begin{pmatrix} 1 & 0 & 0 & -1.72 \\ 0 & 1 & 0 & -6.72 \\ 0 & 0 & 1 & 3.09 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

such that the equation

$$\boldsymbol{B}_f\boldsymbol{N} = \boldsymbol{B}$$

holds. Hence a complete reconfiguration is possible by using the third control input, which is not used in the nominal case. The reconfigured controller

$$\boldsymbol{u}(t) = -\boldsymbol{N}\boldsymbol{K}\boldsymbol{y}(t)$$

has the controller matrix

$$\boldsymbol{NK} = \begin{pmatrix} 0.5 & 0 & -0.86 \\ 0 & 1 & -3.36 \\ 0 & 0 & 1.55 \\ 0 & 0 & 0 \end{pmatrix}.$$

Obviously, the fourth actuator is no longer used. The effect of this actuator is distributed among the three remaining actuators, which can be seen in the last column of the new controller matrix. With the reconfigured controller, the behaviour of the nominal system is completely reproduced.

Controller reconfiguration by means of a virtual actuator. The loss of the actuator V_1 does not affect the operation point, but it breaks the level control loop for the reactor B_1. The use of a reduced virtual actuator allows to keep the nominal controller while changing the control structure as little as possible.

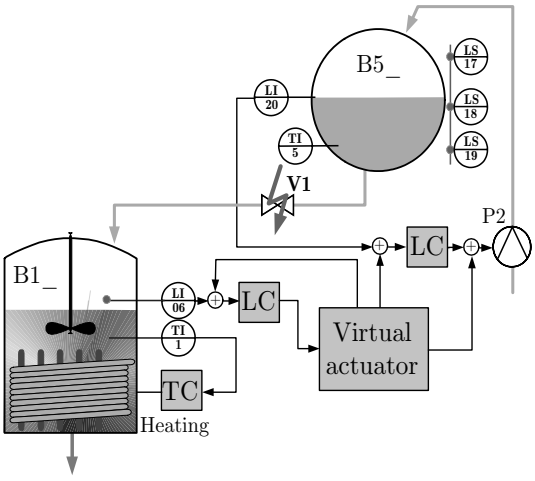


Fig. 10.19. Reconfigured controller including a virtual actuator

In the terminology of Section 7.5.3, the directly influencable part $\boldsymbol{x}_{F1}$ of the plant state is defined by V_{B5} and ϑ_{B1}, while $\boldsymbol{x}_{F2}$ is the single state variable V_{B1}:

$$\boldsymbol{x}_{f1}(t) = \begin{pmatrix} B_{B5}(t) \\ \vartheta_{B5}(t) \end{pmatrix}, \qquad \boldsymbol{x}_{f2}(t) = V_{B1}(t).$$

The $(1,\ 2)$-parameter matrix $\boldsymbol{M}$ is determined by pole placement. The element of $\boldsymbol{M}$ that is acting on ϑ_{B1} has no influence on the actuator pole and is, therefore, set to 0. The other value is chosen so that the actuator pole lies at -0.004 in order to make the influence of the virtual actuator on the closed-loop dynamics as small as possible. The application of the method explained in Section 7.5.4 to this example leads to

$$
\begin{aligned}
\dot{\hat{x}}_2(t) &= -0.004\,\hat{x}_2(t) + 0.0229\,u_{V2,R}(t) \\
\hat{\boldsymbol{u}}(t) &= \begin{pmatrix} 0.015 \\ -0.318 \\ 0 \end{pmatrix} \hat{x}_2(t) + \begin{pmatrix} -0.107 \\ 1.78 \\ 0 \end{pmatrix} u_{V2,R}(t) \\
\hat{\boldsymbol{y}}(t) &= \begin{pmatrix} -8 \\ 0 \\ 1 \end{pmatrix} \hat{x}_2(t) \ .
\end{aligned}
$$

The function of the reduced virtual actuator can be described as follows (Fig. 10.19). The input $u_{V1}(t)$ is not available to control the inflow into the main reactor, but this inflow also depends on the level in Tank B_5 and, hence, on V_{B5}. In order to reach the same effect as the broken actuator, $V_{B5}(t)$ is increased or decreased by influencing the Pump P_2 via the input $u_{P2}(t)$. As $V_{B5}(t)$ cannot be changed instantaneously, this "replacement action" is slower than the direct action of the nominal control loop on the valve V_1 and leads to a slower reaction of the system under the influence of the reconfigured controller.

In mathematical terms, the virtual actuator brings about an additional pole which yields the slower dynamics. The difference between the nominal and the new behaviour is determined by the virtual actuator and deducted from the measurements of $V_{B1}(t)$ and $V_{B5}(t)$. In this way, the additional pole remains hidden from the level controller and this controller acts like in the nominal case.

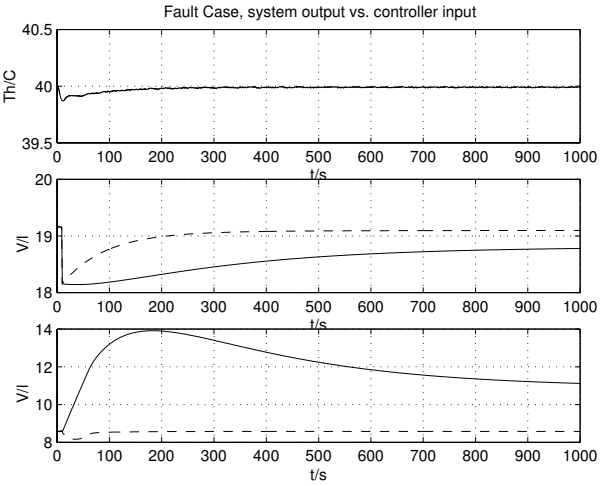


Fig. 10.20. Results of the reconfiguration experiment (Reactor temperature $\vartheta_{B1}(t)$ (top), reactor content $V_{B1}(t)$ (middle) and reactor content $V_{B5}(t)$

The experimental results are shown in Fig. 10.20. The state $V_{B1}(t)$ is disturbed by withdrawing a considerable amount of liquid until time $t = 10\,\mathrm{s}$. The virtual actuator increases the level $V_{B5}(t)$ in Tank B_5 by increasing the pump input $u_{P1}(t)$. The effect of this manipulation and of the fault is "'simulated"' by the virtual actuator, subtracted from the sensors data and, therefore, hidden from the nominal controller. After 180 seconds the tank level $V_{B5}(t)$ reaches its maximum and after another 800 seconds the state deviation has been reasonably compensated. A static deviation remains because of some modelling inaccuracies.

The dashed lines show the behaviour of the faultless closed-loop system. The slower reaction of the level controller results in the slower disturbance attenuation shown in the middle part of the figure, where the nominal system reaches the set-point of $19\,\mathrm{dm}^3$ quicker than the reconfigured system. Hence, the operation of the main reactor can be restored with a minor performance degradation.

In the lower part of the figure the different behaviour of Tank B_5 can be seen. The difference is due to the different functions that this tank has in both situations. In the faultless case the level controller of this tank adjusts the liquid content to the set-point, whereas under faulty conditions this variable is used as a means to control the inflow into Tank B_1 and, thus, to control the contents of B_1.

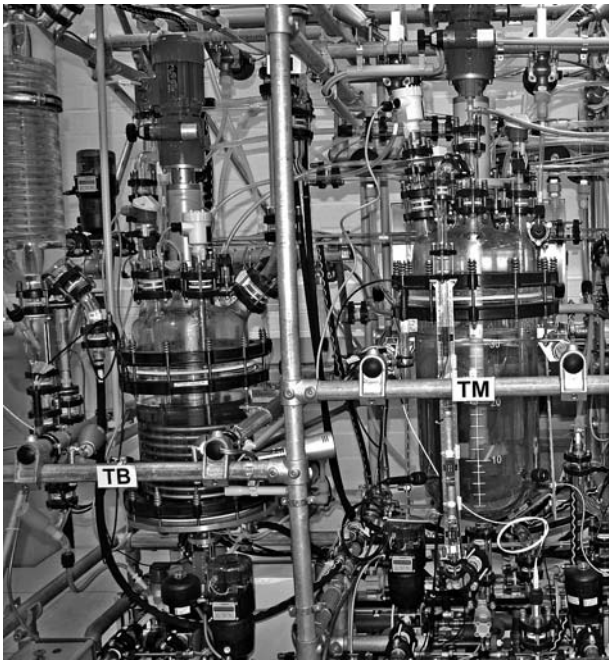


Fig. 10.21. Part of the chemical plant VERA used in the experiment

10.2.3 Reconfiguration of a conductivity control loop

The second application of the reconfiguration method that uses the virtual actuator is the fault-tolerant control of the conductivity of a liquid. Figure 10.21 shows the experimental set-up and Fig. 10.22 the schematic diagram of the three reactors involved in the control loop considered. The sequence of the two Reactors TM and TB with the Reactor TS is used to produce a liquid with prescribed temperature and conductivity. Several control loops have to be used, which are shown in the schematic diagram with the abbreviations LC for level controller, TC for temperature controller and CC for concentration controller. If actuator failures occur, these loops are brought out of operation. Typical failures concern the valves V_{TM} and V_{CW} and the heating P_{el}.

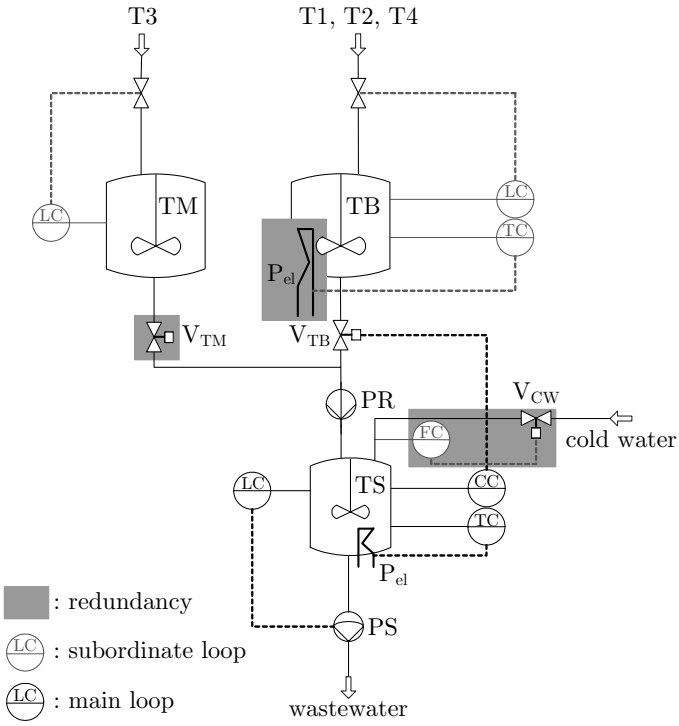


Fig. 10.22. Schematic diagram of the process

The nominal controller uses the inputs u_{PS}, u_{TS} and u_{TB}, which are subject to the three failures. The three variables to be controlled are the temperature ϑ_{TB}, the liquid level l_{TS} in the Reactor TS and the conductivity λ_{TS} of the liquid in the Reactor TS (Fig. 10.23). The block diagram also shows the redundant inputs u_{CW} and u_{TM}, which will be used for the reconfiguration.

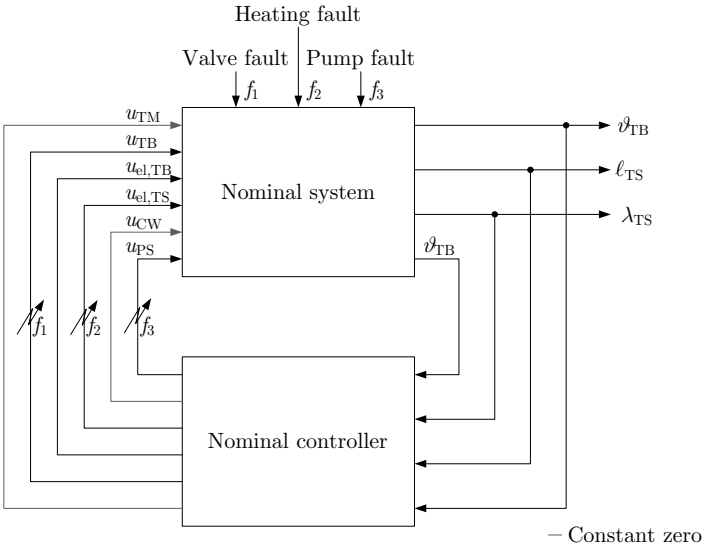


Fig. 10.23. Schematic diagram of the process

Nonlinear model. The following nonlinear model is obtained from balance equations that concern the different components of the plant. To shorten the notation of the equations, the dependency of the signals from the time t is omitted:

• Change of the liquid temperature of Reactor TS:

$$\begin{aligned}\dot{\vartheta}_{TS} &= \frac{1}{A_{TS}\rho l_{TS}}\left\{\frac{P_{el,TS}-\dot{Q}_{PL,TS}}{c_p}+\dot{m}_{TB}(\vartheta_{TB}-\vartheta_{TS})+\right.\\ &\left.\dot{m}_{TM}(\vartheta_{TM}-\vartheta_{TS})+\dot{m}_{CW}(\vartheta_{CW}-\vartheta_{TS})\right\}\end{aligned}$$

• Change of the liquid volume in Reactor TS:

$$\dot{l}_{TS}(t)=\frac{\dot{m}_{TB}(t)+\dot{m}_{TM}(t)+\dot{m}_{CW}(t)-\dot{m}_{TW}(t)}{A_{TS}\rho}$$

• Change of the concentration in Reactor TS:

$$\begin{aligned}&\dot{c}_{TS}(t)\\ &=\frac{\dot{m}_{TB}(t)(c_{TB}-c_{TS}(t))+\dot{m}_{TM}(t)(c_{TM}-c_{TS}(t))-\dot{m}_{CW}(t)c_{TS}(t)}{A_{TS}\rho l_{TS}(t)}\end{aligned}$$

• Change of the liquid temperature in Reactor TB:

$$\begin{aligned}&\dot{\vartheta}_{TB}(t)\\ &=\frac{1}{A_{TB}\rho l_{TB}}\left\{\frac{P_{el,TB}(t)-\dot{Q}_{PL,TB}(t)}{c_p}+\dot{m}_{T124}(t)(\vartheta_{T124}-\vartheta_{TB}(t))\right\}\end{aligned}$$

• Behaviour of the cold water Valve V_{CW}:

$$\begin{aligned}\dot{x}_{CW}(t) &= -\frac{1}{T_{CW}}x_{CW}(t)+\frac{1}{T_{CW}}u_{CW}(t)\\ \dot{m}_{CW}(t) &= x_{CW}(t) \qquad \text{with } T_{CW}=3,7\,\mathrm{s}\end{aligned}$$

• Actuator dynamics of the heating of the Reactor TB:

$$\begin{aligned}\dot{x}_{TB}(t) &= -\frac{1}{T_{el,TB}}x_{TB}(t)+\frac{1}{T_{el,TB}}u_{TB}(t)\\ P_{el,TB}(t) &= k_{TB}x_{TB}(t),\\ &\quad \text{with } T_{el,TB}=27\,\mathrm{s},\ \ k_{TB}=18\,\mathrm{kW}\end{aligned}$$

• Actuator dynamics of the heating of the Reactor TS:

$$\begin{aligned}\dot{x}_{TS}(t) &= -\frac{1}{T_{el,TS}}x_{TS}(t)+\frac{1}{T_{el,TS}}u_{TS}(t)\\ P_{el,TS}(t) &= k_{TS}x_{TS}(t),\\ &\quad \text{with } T_{el,TS}=65\,\mathrm{s},\ \ k_{TS}=4\,\mathrm{kW}\end{aligned}$$

Besides the state variables ϑ_{TB} and l_{TS}, the conductivity is the third variable to be controlled. This signal is obtained by the following relation:

$$\lambda_{TS}(t)=0,4469\frac{\mathrm{mS}}{\mathrm{cm}}+2047,7\frac{\mathrm{mS}}{\mathrm{cm}}c_{TS}(t).$$

All these equations use the following mass and heat flows:

• Mass flow from Rector TB towards Reactor TS:

$$\dot{m}_{TB}(t)=\begin{cases}\left(0,019\frac{\mathrm{kg}}{\mathrm{s}\sqrt{\mathrm{m}}}+0,727\frac{\mathrm{kg}}{\mathrm{s}\sqrt{\mathrm{m}}}(u_{TB}(t)-0,13)\right)\sqrt{l_{TB}+0,3\,\mathrm{m}}, & \\ \qquad\qquad \text{if } u_{TB}\geq 0,13 & \\ 0\frac{\mathrm{kg}}{\mathrm{s}}, & \\ \text{else} & \end{cases}$$

• Mass flow from Rector TM towards Reactor TS:

$$\dot{m}_{TM}(t)=\begin{cases}\left(0,047\ \frac{\mathrm{kg}}{\mathrm{s}\sqrt{\mathrm{m}}}+0{,}605\ \frac{\mathrm{kg}}{\mathrm{s}\sqrt{\mathrm{m}}}(u_{TM}(t)-0,04)\right)\sqrt{l_{TM}+0{,}3\,\mathrm{m}} & \\ \qquad\qquad \text{if } u_{TM}\geq 0{,}04 & \\ 0\frac{\mathrm{kg}}{\mathrm{s}} \qquad\qquad \text{else.} & \end{cases}$$

• Mass flow out of the Reactor TS:

$$\dot{m}_{PS}(t)=\dot{m}_{TW}(t)=0,1679\frac{\mathrm{kg}}{\mathrm{s}\sqrt{\mathrm{m}}}u_{PS}(t)\sqrt{l_{TS}(t)+0,36\,\mathrm{m}}$$

• Heat balance of the Reactor TS:

$$\dot{Q}_{PL,TS}(\vartheta_{TS}(t))=\begin{cases}\dot{Q}_{PL,TS,on}(\vartheta_{TS}(t)), & \text{if heating is on}\\ \dot{Q}_{PL,TS,off}(\vartheta_{TS}(t)), & \text{if heating is off}\end{cases}$$

with

$$\dot{Q}_{PL,TS,on}(\vartheta_{TS}(t)) = \begin{cases} 46,9403\frac{\mathrm{W}}{^\circ\mathrm{C}}(\vartheta_{TS}(t) - 22,5\,^\circ\mathrm{C}), & \text{if } \vartheta_{TS} \geq 22,5\,^\circ\mathrm{C} \\ 0\,\mathrm{W}, & \text{if } \vartheta_{TS} < 22,5\,^\circ\mathrm{C} \end{cases}$$

$$\dot{Q}_{PL,TS,off}(\vartheta_{TS}(t)) = \begin{cases} 4,8968\frac{\mathrm{W}}{^\circ\mathrm{C}}(\vartheta_{TS}(t) - 22,5\,^\circ\mathrm{C}), & \text{if } \vartheta_{TS} \geq 22,5\,^\circ\mathrm{C} \\ 0\,\mathrm{W}, & \text{if } \vartheta_{TS} < 22,5\,^\circ\mathrm{C} \end{cases}$$

• Heat balance of the Reactor TB:

$$\dot{Q}_{PL,TB}(\vartheta_{TB}(t)) = \begin{cases} \dot{Q}_{PL,TB,on}(\vartheta_{TB}(t)), & \text{if heating is on} \\ \dot{Q}_{PL,TB,off}(\vartheta_{TB}(t)), & \text{if heating is off} \end{cases}$$

$$\dot{Q}_{PL,TB,on}(\vartheta_{TB}(t)) = \begin{cases} 135,468\frac{\mathrm{W}}{^\circ\mathrm{C}}(\vartheta_{TB}(t) - 22,5\,^\circ\mathrm{C}), & \text{if } \vartheta_{TB} \geq 22,5\,^\circ\mathrm{C} \\ 0\,\mathrm{W}, & \text{if } \vartheta_{TB} < 22,5\,^\circ\mathrm{C}. \end{cases}$$

$$\dot{Q}_{PL,TB,off}(\vartheta_{TB}(t)) = \begin{cases} 4,8968\frac{\mathrm{W}}{^\circ\mathrm{C}}(\vartheta_{TB}(t) - 22,5\,^\circ\mathrm{C}), & \text{if } \vartheta_{TB} \geq 22,5\,^\circ\mathrm{C} \\ 0\,\mathrm{W}, & \text{if } \vartheta_{TB} < 22,5\,^\circ\mathrm{C} \end{cases}$$

The given equations can be lumped together to get a nonlinear state-space model (9.3), (9.4)

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{g}(\boldsymbol{x}(k), \boldsymbol{u}(k)), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0 \\ \boldsymbol{y}(k) &= \boldsymbol{h}(\boldsymbol{x}(k), \boldsymbol{u}(k)) \end{aligned}$$

with the state, input and output vectors

$$\boldsymbol{x}(t) = \begin{pmatrix} \vartheta_{TS}(t) \\ l_{TS}(t) \\ c_{TS}(t) \\ \vartheta_{TB}(t) \\ x_{CW}(t) \\ x_{TB}(t) \\ x_{TS}(t) \end{pmatrix}, \quad \boldsymbol{u}(t) = \begin{pmatrix} u_{TM}(t) \\ u_{TB}(t) \\ u_{TB}(t) \\ u_{TS}(t) \\ u_{CW}(t) \\ u_{PS}(t) \end{pmatrix}, \quad \boldsymbol{y}(t) = \begin{pmatrix} \vartheta_{TS}(t) \\ l_{TS}(t) \\ \lambda_{TS}(t) \\ \vartheta_{TB}(t) \end{pmatrix}.$$

Linearised model. A linearised state-space model

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

is obtained from the nonlinear model with the following matrices:

$$\boldsymbol{A} = 10^{-3} \cdot \begin{pmatrix} -3,46 & 0 & 0 & 1,46 & -59,12 & 0 & 39,36 \\ 0 & -0,76 & 0 & 0 & 1,41 & 0 & 0 \\ 0 & 0 & -3,15 & 0 & -0,0034 & 0 & 0 \\ 0 & 0 & 0 & -1,34 & 0 & 157,46 & 0 \\ 0 & 0 & 0 & 0 & -270,27 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -37,03 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -15,38 \end{pmatrix}$$

$$\boldsymbol{B} = 10^{-3} \cdot \begin{pmatrix} -10,62 & 0 & 0 & 0 & 0 & 0 \\ 7,11 & 8,49 & 0 & 0 & 0 & -1,98 \\ 0,0249 & 0,0235 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 270,27 & 0 \\ 0 & 0 & 37,03 & 0 & 0 & 0 \\ 0 & 0 & 0 & 15,38 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{C} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2047,7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{D} = \boldsymbol{O}.$$

The set of eigenvalues of the matrix $\boldsymbol{A}$

$$\sigma = \{-0,2703; -0,0370; -0,0154; -0,0035; -0,0032; -0,0013; -0,0008\}$$

gives an impression of the dynamical properties of the plant.

Models of the faulty system. The three actuator failures cause a change of the matrix $\boldsymbol{B}$ of the linearised state-space model:

• Failure f_1 of the Valve V_{TB}, which gets the input signal u_{TB}:

$$\boldsymbol{B}_{f_1} = 10^{-3} \cdot \begin{pmatrix} -10,62 & 0 & 0 & 0 & 0 & 0 \\ 7,11 & 0 & 0 & 0 & 0 & -1,98 \\ 0,0249 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 270,27 & 0 \\ 0 & 0 & 37,03 & 0 & 0 & 0 \\ 0 & 0 & 0 & 15,38 & 0 & 0 \end{pmatrix}$$

• Failure f_2 of the heating of the Reactor TS, which acts according to the control input u_{TS}:

$$\boldsymbol{B}_{f_2} = 10^{-3} \cdot \begin{pmatrix} -10,62 & 0 & 0 & 0 & 0 & 0 \\ 7,11 & 8,49 & 0 & 0 & 0 & -1,98 \\ 0,0249 & 0,0235 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 270,27 & 0 \\ 0 & 0 & 37,03 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

• Failure f_3 of the Pump PS, which runs according to the control input u_{PS}:

$$\boldsymbol{B}_{f_3} = 10^{-3} \cdot \begin{pmatrix} -10,62 & 0 & 0 & 0 & 0 & 0 \\ 7,11 & 8,49 & 0 & 0 & 0 & 0 \\ 0,0249 & 0,0235 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 270,27 & 0 \\ 0 & 0 & 37,03 & 0 & 0 & 0 \\ 0 & 0 & 0 & 15,38 & 0 & 0 \end{pmatrix}.$$

These matrices differ from the matrix $\boldsymbol{B}$ for the nominal model with respect to one column each, which is set to zero for the failed actuator.

Control reconfiguration by a virtual actuator. For all three fault cases, the virtual actuator described in Definition 7.7 is used for the control reconfiguration (Fig 10.24). The scheme is the same in all cases, only the matrix $\boldsymbol{B}_f$, which is a parameter of the virtual actuator, differs. This shows that the control reconfiguration is completely automatic in the sense that a general reconfiguration algorithm can be applied, which adapts the effect of the nominal controller to the failure that has occurred.

The first experiment concerns the reconfiguration with the goal to retain the stability of the closed-loop system. For this task, a virtual actuator with parameter matrix $\boldsymbol{N} = \boldsymbol{O}$ is used.

In case of the failure of the Valve V_{TB}, the virtual actuator has been designed to have the following set of eigenvalues:

$$\begin{aligned} \sigma_{VA} &\stackrel{!}{=} 25\sigma \\ &= \{-6,7568; -0,9259; 0,3846; -0,0866; -0,0790; -0,0335; -0,0190\} \end{aligned} \tag{10.9}$$

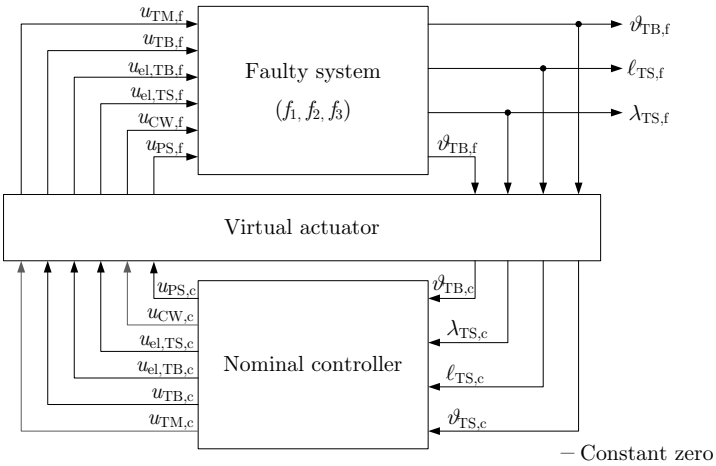


Fig. 10.24. Reconfiguration by means of a virtual actuator

This is accomplished by the feedback matrix

$$\boldsymbol{M} = \begin{pmatrix} -12{,}31 & -16{,}05 & 77{,}63 & 0{,}15 & 5{,}11 & 0{,}40 & -3{,}71 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 13{,}39 & -0{,}01 & 5770 & 90{,}07 & -23{,}71 & 178{,}06 & 15{,}41 \\ 17{,}18 & -0{,}06 & 7332 & 23{,}26 & -31{,}85 & 39{,}14 & 25{,}31 \\ -1{,}48 & -0{,}01 & -642{,}30 & -2{,}04 & 2{,}11 & -3{,}43 & -1{,}81 \\ -130{,}19 & -192{,}04 & 239{,}73 & 0{,}75 & 18{,}61 & 2{,}21 & -12{,}85 \end{pmatrix}.$$

This pole assignment is possible, because the pair $(\boldsymbol{A}, \boldsymbol{B}_{f1})$ is completely controllable. The eigenvalues are chosen with respect to the eigenvalues of the plant. They make the virtual actuator much quicker than the plant. The zero row of the matrix $\boldsymbol{M}$ ensures that the failed valve is no longer used for feedback control. Due to the separation property of the virtual actuator, the overall closed-loop system has the eigenvalues of the nominal closed-loop system and the eigenvalues given in Eq. (10.9) for the virtual actuator. Hence, the reconfigured system is stable.

Figure 10.25 approves this result. The two bars placed at time $t = 350\,\mathrm{s}$ mark the time instant at which the valve is blocked and the controller reconfigured. The temperature ϑ_{TS} and the level l_{TS} remain at the set-points, whereas the conductivity cannot follow precisely the set-point change at time $t = 300\,\mathrm{s}$ marked by the dashed line. This is due to the proportional controller used.

Figure 10.26 shows the six control inputs. After the valve V_{TB} is blocked, the signal u_{TB} shown in the top right corner of the figure does no longer change. The virtual actuator uses the input signals u_{TS}, u_{TB} and u_{PS} which are also used by the nominal controller. In addition to this, the virtual actuator exploits the input u_{CW} to the cold water Valve V_{CW}, whereas the other additional input u_{TM} is not used.

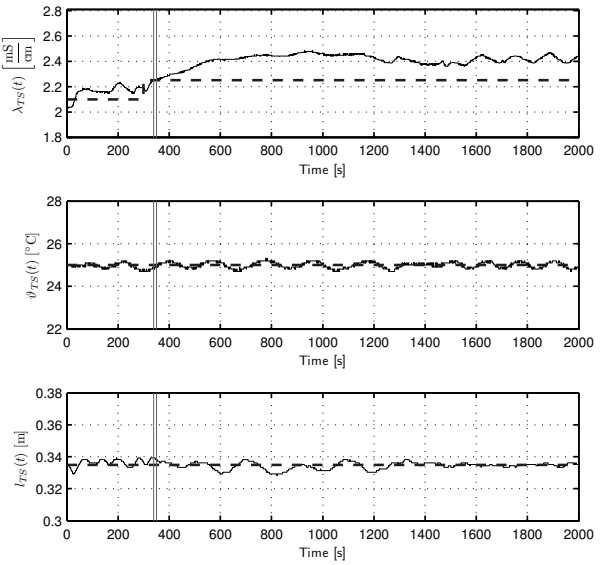


Fig. 10.25. Reconfiguration in case of the valve V_{TB}-failure

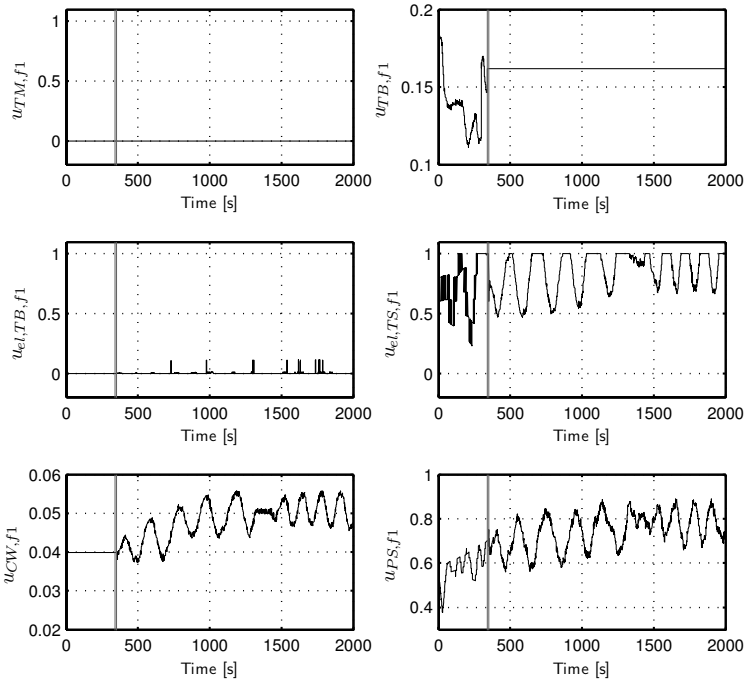


Fig. 10.26. Absolute values of the control input after the reconfiguration in case of the valve V_{TB}-failure

The choice how to distribute the effect of the blocked valve over the remaining actuators is made implicitly by the virtual actuator. No selection procedure, with a possible involvement of a human control designer, is necessary. Therefore, the concept of the virtual actuator can be applied completely automatically.

The second experiment concerns the aim to bring all variables to be controlled back to their set-points. Here the "'complete"' virtual actuator with the two parameter matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ is used. Besides the matrix $\boldsymbol{M}$ given above, the direct feedthrough is chosen as

$$\begin{aligned}\boldsymbol{N} &= \left(\boldsymbol{C}(\boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{M})^{-1}\boldsymbol{B}_f\right)^{-1}\left(\boldsymbol{C}(\boldsymbol{A}-\boldsymbol{B}_f\boldsymbol{M})^{-1}\boldsymbol{B}\right)\\ &= \begin{pmatrix} 1 & 0{,}291 & -0{,}016 & 0{,}053 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0 & -0{,}588 & 0{,}031 & -0{,}037 & 1 & 0\\ 0 & -4{,}250 & -0{,}012 & -0{,}004 & 0 & 1 \end{pmatrix},\end{aligned}$$

which ensures set-point following, because the reconfigured closed-loop system has the same static reinforcement as the nominal control loop.

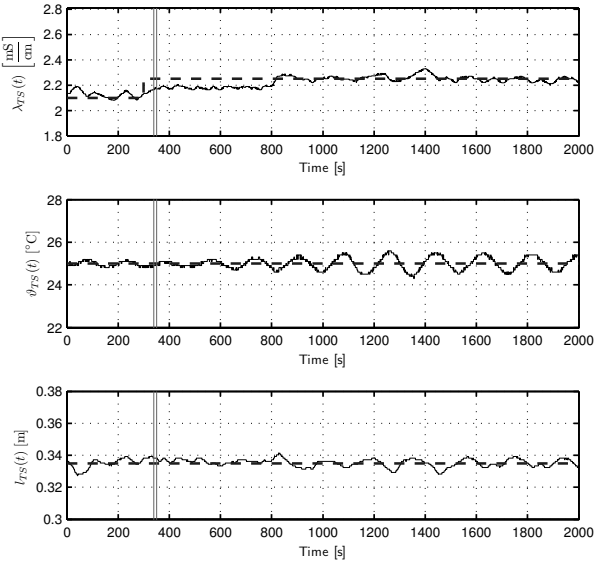


Fig. 10.27. Reconfiguration after valve V_{TB}-failure

The reconfiguration result is depicted in Fig. 10.27. The same experiment has been made as before, but now all three control outputs are moved back to their set-points.

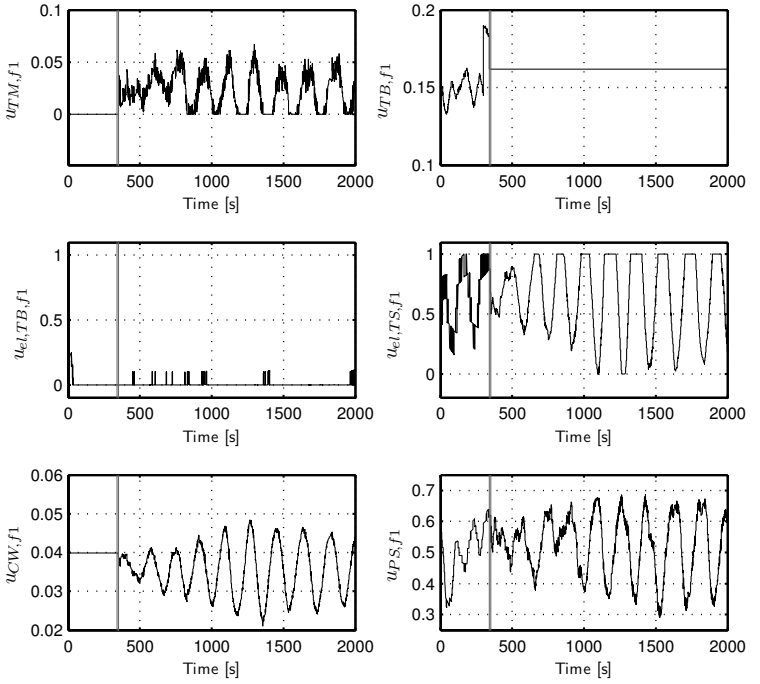


Fig. 10.28. Control input after the reconfiguration for valve V_{TM}-failure

As Fig. 10.28 shows, the virtual actuator uses now the additional inputs u_{CW} and u_{TM}. The reconfiguration is completely successful including the restoration of the set-point.

10.3 Diagnosis and control of a ship propulsion system

10.3.1 Structure of the ship propulsion system

Faults in a ship propulsion system and its associated automation system can cause a dramatic reduction in the ship's ability to propel and manoeuvre itself, and effective means are needed to prevent faults to develop into a failure. Various algorithms and methods from different research areas can be used to analyse the system and subsequently detect, isolate, and accommodate the faults. The ship propulsion system described in this section was presented as an international benchmark and was used as a platform for development of new ideas and comparison of methods.

The topics selected for this section are based on structural analysis. It is shown how residual generators are directly deduced from analysis of structure and how fault-tolerance can be obtained. Diagnostic methods and a supervisor logic to obtain

fault-tolerant control are implemented and tested against recorded time-history data which were manipulated to include well defined faults.

The dynamics of the propulsion system is non-linear. Furthermore, one essential fault is non-additive. The implication is that some residual generators become non-linear. This section illustrates how such real-life phenomena can be handled in the general framework developed in this book and where slight extensions are needed.

The propulsion system example origins from studies and manoeuvering trials with the Danish intercity ferry MF Dr. Ingrid, a 10.000 tons combined passenger and train ferry. Detailed modelling and data recorded from manoeuvering trials with the ferry give a realistic scenario for test of diagnostic methods and techniques to obtain fault tolerance.

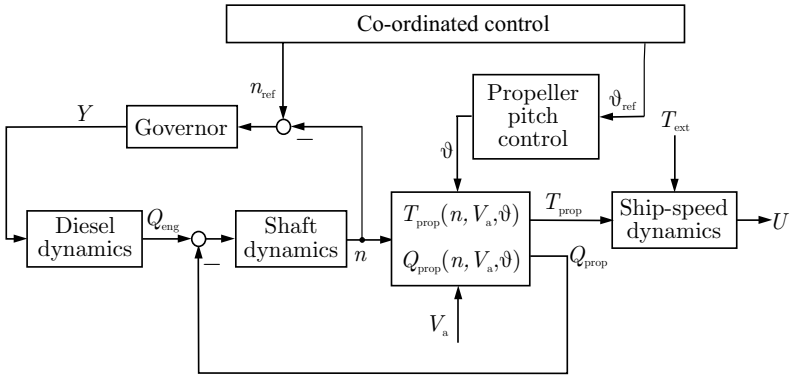


Fig. 10.29. Block diagram of the ship propulsion system

Ship propulsion system. An outline of the propulsion system chosen for the benchmark is shown in Fig. 10.29 (of Table 10.8 for a list of symbols). The main components are described by the following blocks:

- *Diesel dynamics* gives engine torque to drive the propeller shaft.
- *Shaft dynamics* provides shaft speed given diesel and propeller torques.
- *Propeller characteristics* provide propeller thrust and load torque from shaft speed n, propeller pitch ϑ and water speed V_a;
- *Ship speed dynamics* determines ship speed from propeller thrust and external forces.
- *Propeller pitch and shaft speed controllers* (governor) control the propeller pitch and shaft speed.
- *The coordinated control level* calculates set-points for shaft speed and propeller pitch controllers.

Table 10.8 List of symbols used in the ship propulsion system

Symbol	Unit	Explanation
I_t	kgm^2	Total inertia
K_y	Nm	Torque coefficient
$n(t)$	rads^{-1}	Shaft speed
$R(U)$	N	Hull resistance
$T_{prop}(t)$	N	Propeller thrust
$T_{ext}(t)$	N	External force
$1-t_T$	-	Thrust deduction factor
$U(t)$	ms^{-1}	Ship speed
$V_a(t)$	ms^{-1}	Flow at propeller
$1-w$	-	Wake fraction
$Q_{eng}(t)$	Nm	Diesel torque
Q_f	Nm	Shaft friction
$Q_{prop}(t)$	Nm	Propeller torque
$Y_d(t)$	0...1	Fuel index
$\vartheta(t)$	-1...1	Propeller pitch

The coordinated control level is detailed in Fig. 10.30. The following functions are included:

- **Combinator**: Gives a set of command values: $n_{\rm com}(t)$ and $\vartheta_{\rm com}(t)$ as functions of the command handle position.
- **Efficiency optimiser**: A module to optimise propulsion efficiency determines $n_{\rm com}(t)$ and $\vartheta_{\rm com}(t)$, based on measured values of $Y(t)$, $n(t)$, $\vartheta(t)$ and $U(t)$.
- **Speed control**: A ship speed-control module maintains a command value of ship speed $U_{\rm ref}$, using measured values of $Y(t)$, $n(t)$, $\vartheta(t)$ and $U(t)$ as input.
- **Overload control**: Modifies $n_{\rm com}(t)$ and $\vartheta_{\rm com}(t)$ to prohibit the prime mover from reaching its torque limits. The fuel index is used to determine an approaching overload condition.

10.3.2 Models of the propulsion system

The overall function of the propulsion system is to maintain the ship's ability to propel itself and to manoeuvre. Propulsion requires thrust ahead whereas manoeuvres require ahead and astern thrust ability. With a positive shaft speed n, this is

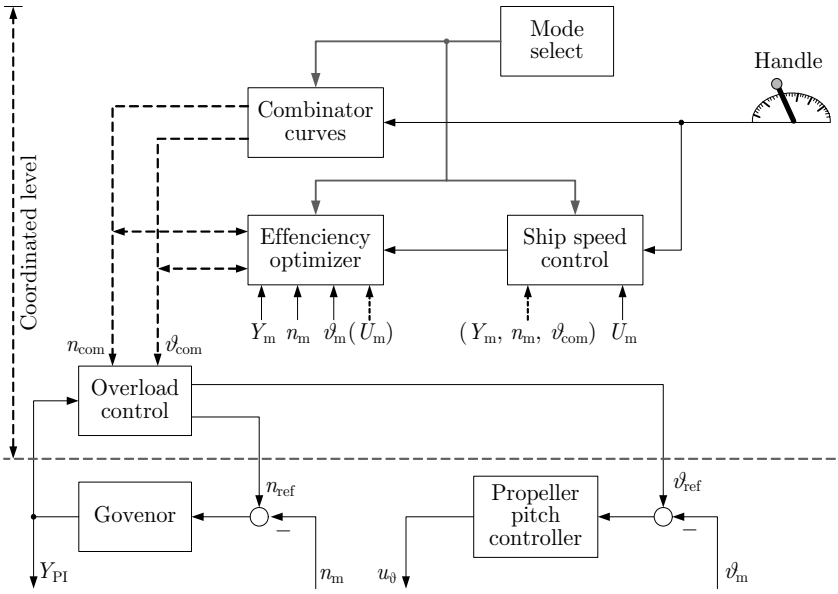


Fig. 10.30. Hierarchy of controllers for the propulsion system. The handle gives input to a combinator, efficiency optimiser, and ship speed control module. Lower level controls are shaft speed (governor), propeller pitch and diesel overload blocks.

obtained by an appropriate change of the propeller pitch ϑ, which is the angle that the propeller blades are twisted.

The component hierarchy is treated as belonging to two levels. Lower level components are the diesel engine with shaft speed controller, the propeller with the pitch controller, and the ship's speed dynamics. The upper level comprises coordinated control for the lower level components and overall command to the propulsion system. Reconfiguration will take place at the upper level, but lower-level controllers should be fault-tolerant, if possible, to maintain their primary services.

Upper-level components. Upper level components are the following:

- **Command handle**: A command handle's position constitutes the main man-machine interface (MMI).

- **Combinator**: Use-modes with different interpretations of handle position are available:
  - **Manoeuvring**: Handle position determines $n(t)$ and $\vartheta(t)$;
  - **Economy**: Handle position determines $n_{\mathrm{com}}(t)$ and $\vartheta_{\mathrm{com}}(t)$;
  - **Set speed**: Maintain a set ship speed using measured ship speed $U(t)$.

- **Efficiency optimiser**: The efficiency optimiser determines the set of $n(t)$ and $\vartheta(t)$ that achieves the desired ship speed $U_{\mathrm{ref}} = f_{sc}(h(t))$ as determined by the handle position, without ship speed feedback.

- **Ship speed control**: Ship speed control aims at maintaining a set ship speed within a narrow margin. This component uses measured ship speed as one of its input variables.

- **Diesel overload control**: Overload is avoided by reducing the propeller pitch if diesel torque is close to maximum at a given shaft speed.

Lower-level components. The lower level consists of the shaft speed and the propeller-pitch controllers and the physical components of the propulsion system. In a component-based analysis, the physical components related to the pitch control function are lumped together to a new entity called *propeller pitch control*.

Propeller pitch control. The pitch control is an aggregated component that comprises a large hydraulic actuator turning the propeller blades, the feedback from a pitch sensor, a controller and the drive electronics. In its original implementation, this component has only one version of the use-mode um_1, which denotes the automatic mode. In order to obtain fault-tolerant properties, other versions are added. The concise definition of the component is given in the following equations:

$$
\begin{aligned}
&< \mathbf{Propeller\ pitch\ control} > \quad ::= \\
&\qquad < M(0,1) > \quad ::= \quad < um_0\ (manual),\ um_1\ (automatic) > \\
&\qquad < um_o\ (manual) > \quad ::= \quad < s_o > \\
&\qquad < um_1\ (automatic) > \quad ::= \quad < s_o,\ s_1 > \\
&\qquad < \mathbf{s_0}\ (\mathbf{3-state}) > \quad ::= \quad < v_0\ (up-down) > \\
&\qquad < v_0\ (up-down) > \quad ::= \quad < consumed >, < produced >, \\
&\qquad\qquad\qquad\qquad\quad < procedure >, < request >, \\
&\qquad\qquad\qquad\qquad\quad < activation >, < resources > \\
&\qquad < consumed > \quad ::= \quad < command\ \in [up,\ nil,\ down], > \\
&\qquad < produced > \quad ::= \quad < pitch\ angle\ of\ blades, > \\
&\qquad < procedure > \quad ::= \quad < plant\ dynamics\ \text{Eq. (10.12)}\ with \\
&\qquad\qquad\qquad\qquad\quad control\ (10.13), > \\
&\qquad < request > \quad ::= \quad < select\ 3-state, > \\
&\qquad < activation > \quad ::= \quad < none, > \\
&\qquad < resources > \quad ::= \quad < hydraulic\ oil\ supply,\ CP\ propeller >
\end{aligned}
$$

$$
\begin{array}{rcl}
< \boldsymbol{s_1}\,(\boldsymbol{continuous}) > & ::= & < v_1\,(normal),\ v_2\,(ftc-a) > \\
 & & \\
< v_1\,(normal) > & ::= & < consumed >, < produced >, \\
 & & < procedure >, < request >, \\
 & & < activation >, < resources > \\
< consumed > & ::= & < pitch\ angle\ command, > \\
< produced > & ::= & < pitch\ angle\ of\ blades, > \\
< procedure > & ::= & < plant\ dynamics \text{ Eq. (10.12)} \ with\ control \\
 & & (10.14), > \\
< request > & ::= & < select\ automatic,\ normal, > \\
< activation\ condition > & ::= & < hydraulic\ pressure\ present, > \\
< resources > & ::= & < angle\ sensor,\ hydraulic\ oil\ supply, \\
 & & CP\ propeller > \\
 & & \\
< v_1\,(ftc-a) > & ::= & < consumed >, < produced >, \\
 & & < procedure >, < request >, \\
 & & < activation >, < resources > \\
< consumed > & ::= & < pitch\ angle\ command, > \\
< produced > & ::= & < pitch\ angle\ of\ propeller, > \\
< procedure > & ::= & < \text{Eq. (10.12)} \ and\ fault-tolerant\ control \\
 & & results, > \\
< request > & ::= & < select\ automatic,\ ftc-a, > \\
< activation > & ::= & < hydraulic\ pressure\ present, > \\
< resources > & ::= & < hydraulic\ oil\ supply, CP\ propeller > . \\
< FPA\ input \mid um_0 \vee um_1 > & ::= & < sensor\ fault\ f_1 = \Delta\vartheta,\ \text{leak}\ f_2 = \Delta\dot{\vartheta}_{inc}, \\
 & & e_{i1} = \vartheta_{\text{com}} > \\
< FPA\ output \mid um_0 \vee um_1 > & ::= & < e_{o1} = \vartheta L > \\
< FPA\ description \mid um_o > & ::= & < M_{pc-um0} > \\
< FPA\ description \mid um_1 > & ::= & < M_{pc-um1} >
\end{array}
$$

The mathematical model for the physical parts of the component is composed of the following equations:

$$\vartheta_m(t) = \vartheta(t) + \nu_\vartheta(t) + \Delta\vartheta(t) \tag{10.10}$$

$$\dot{\vartheta}(t) = \max\left(\dot{\vartheta}_{\min}, \min\left(u_{\dot{\vartheta}}(t), \dot{\vartheta}_{\max}(t)\right)\right) + \Delta\dot{\vartheta}_{\text{inc}} \tag{10.11}$$

$$\vartheta(t) = \max\left(\vartheta_{\min}, \min\left(\vartheta(t),\ \vartheta_{\max}\right)\right). \tag{10.12}$$

The control signal $u_{\dot{\vartheta}}(t)$ is generated according to the version running. In version v_o the control signal is obtained from

$$u_{\dot{\vartheta}}(t) = k_t u_{\text{cmd}}(t), \quad u_{\text{cmd}} \in [-1, 0, 1]. \tag{10.13}$$

In versions v_1 and v_2,

$$u_{\dot{\vartheta}}(t) = k_t\left(\vartheta_{\text{ref}}(t) - \vartheta_m(t)\right) \tag{10.14}$$

holds. Here, $\vartheta_m(t)$ is the measured propeller pitch, $[\dot{\vartheta}_{\min},\ \dot{\vartheta}_{\max}]$ the rate interval set by the hydraulic pump capacity and geometry, and $[\vartheta_{\min},\ \vartheta_{\max}]$ is the physical

interval for propeller-blade travel. $\nu_\vartheta(t)$ is the measurement noise. Two faults are included in the model: leakage $\Delta\dot{\vartheta}_{\mathrm{inc}}(t)$, and pitch sensor fault $\Delta\vartheta(t)$. It is noted that the control signal $u_{\dot{\vartheta}}(t)$ is not measured.

With $(e_i) \in [low,\ high,\ fluc,\ undef]$ we get

$$\boldsymbol{M}_{pc-umo} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}'$$

$$\boldsymbol{M}_{pc-um1} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}'.$$

Shaft speed control. The input to the shaft speed controller, which is called the governor, is given by the shaft speed reference $n_{\mathrm{ref}}(t)$ and the measured shaft speed $n_m(t)$. The output is the throttle of the diesel engine, which is proportional to the fuel index $Y(t)$. The governor is a PI controller. Anti-windup is part of the integrating action, and K is the anti-windup gain.

$$\begin{aligned}
&< \textbf{Shaft speed controller} > \quad ::= \\
&\qquad < M(0,1) > \quad ::= \quad < um_1\ (automatic) > \\
&\\
&\qquad < um_1\ (automatic) > \quad ::= \quad < s_o,\ s_1 > \\
&\\
&\qquad < \boldsymbol{s_0}\ (\textbf{constant}) > \quad ::= \quad < v_0\ (constant) > \\
&\qquad < v_0\ (constant) > \quad ::= \quad < consumed >, < produced >, \\
&\qquad\qquad\qquad\qquad\quad < procedure >, < request >, \\
&\qquad\qquad\qquad\qquad\quad < activation >, < resources > \\
&\qquad < consumed > \quad ::= \quad < logic\ command >, \\
&\qquad < produced > \quad ::= \quad < diesel\ index\ Y >, \\
&\qquad < procedure > \quad ::= \quad < Y = Y_{ta}\ time\ of\ activation >, \\
&\qquad < request > \quad ::= \quad < constant >, \\
&\qquad < activation > \quad ::= \quad < none >, \\
&\qquad < resources > \quad ::= \quad < diesel\ engine > \\
&\\
&\qquad < \boldsymbol{s_1}(continuous) > \quad ::= \quad < v_1\ (normal),\ v_2(ftc-n) > \\
&\qquad < v_1\ (normal) > \quad ::= \quad < consumed >, < produced >, \\
&\qquad\qquad\qquad\qquad\quad < procedure >, < request >, \\
&\qquad\qquad\qquad\qquad\quad < activation >, < resources >
\end{aligned}$$

$$
\begin{array}{rcl}
< consumed > & ::= & < n_m,\ n_{\rm com},\ Y_m >, \\
< produced > & ::= & <\mathrm{Y}>, \\
< procedure > & ::= & < \text{Eqs. (10.15) } and \text{ (10.16)} >, \\
< request > & ::= & < automatic >, \\
< activation > & ::= & < none >, \\
< resources > & ::= & < n_m,\ Y_m,\ n_com,\ power, mainengine >
\end{array}
$$

$$
\begin{array}{rcl}
< v_2\ (ftc-n) > & ::= & < consumed >, < produced >, \\
 & & < procedure >, < request >, \\
 & & < activation >, < resources > \\
< consumed & ::= & < \hat{n},\ n_{\rm com},\ Y_m >, \\
< produced > & ::= & < Y >, \\
< procedure > & ::= & < \text{Eq. (10.15) } with\ n_m = \hat{n}\ and \\
 & & \text{Eq. (10.16)} >, \\
< request > & ::= & < automatic >, \\
< activation > & ::= & < automatic,\ ftc-n >, \\
< resources > & ::= & < \hat{n},\ n_com,\ Y_m,\ power,\ main\ engine >
\end{array}
$$

$$
\begin{array}{rcl}
< FPA\ in\ \mid\ um_1,\ v_0 > & ::= & < n_m,\ n_{\rm com},\ Y_m > \\
< FPA\ in\ \mid\ um_1,\ v_1 > & ::= & < n_m,\ n_{\rm com},\ Y_m > \\
< FPA\ in\ \mid\ um_1,\ v_2 > & ::= & < \hat{n},\ n_{\rm com},\ Y_m > \\
< FPA\ out > & ::= & < Y > \\
< FPA\ in\ \mid\ um_1,\ v_0 > & ::= & < Mat0_{4,12} > \\
< FPA\ out\ \mid\ um_1,\ v_0 > & ::= & < \boldsymbol{M}_{gov-v1} > \\
< FPA\ mat\ \mid\ um_1\ v_0 > & ::= & < \boldsymbol{M}_{gov-v2} >
\end{array}
$$

The controller is given by

$$
\begin{array}{rcl}
n_m(t) & = & n(t) + \nu_n(t) + \Delta n(t) \\
\dot{Y}_i(t) & = & \dfrac{k_r}{\tau_i}\big((n_{\rm ref}(t)\ -\ n_m(T)) - K(Y_{PIb}(t) - Y_{PI}(t))\big) \\
Y_{PIb}(t) & = & Y_i(t)\ +\ k_r.(n_{\rm ref}(t)\ -\ n_m(t)) \\
Y_{PI}(t) & = & \min\left(\max\left(Y_{PIb}(t),\ Y_{lb}\right),\ Y_{ub}\right).
\end{array}
\tag{10.15}
$$

Y_{lb} and Y_{ub} are the lower and upper bounds for the integrator part of the governor, and $\Delta n(t)$ the measurement fault. The governor comprises fuel index limits to keep the diesel engine within its allowed envelope of operation. These limits are given below

$$
y_{\max} = \left\{ \begin{array}{ll}
0.4 & n_m \leq 40\,\% \text{ of } n_{\rm max,a} \\
1 & n_m \geq 80\,\% \text{ of } n_{\rm max,a} \\
\dfrac{1.5 n_m}{n_{\max,\,a}} - 0.2 & \text{otherwise}
\end{array} \right\}
\tag{10.16}
$$

$$
Y(t) = \max(0,\ \min(Y_{PI}(t),\ y_{\max})),
$$

where $n_{\max,a}$ is the maximum generated shaft speed allowed, n_m the measured shaft speed, $y_{\max}$ the hard limit specified for the engine, and $Y \in [0,\, Y_{\max}]$ denotes the command fuel index.

A similar formal description can be made for each physical component, but this is omitted for brevity.

Diesel engine. The diesel engine generates a torque Q_{eng}, which is controlled by its fuel index Y, to drive the shaft. The diesel engine dynamics can be divided into two parts. The first part describes the relation between the generated torque and the fuel index. It is given by the transfer function

$$Q_{\text{eng}}(s) = \frac{K_y + \Delta K_y}{1 + \tau_c\, s} Y(s), \tag{10.17}$$

where K_y is the gain constant and τ_c is the time constant corresponding to torque build-up from cylinder firings.

The second part expresses the torque balance of the shaft:

$$I_m \dot{n}(t) = Q_{\text{eng}}(t) - Q_{\text{prop}}(t) - Q_f. \tag{10.18}$$

$Q_{\text{eng}}(t)$ is the torque developed by the diesel engine, $Q_{\text{prop}}(t)$ is the torque developed from the propeller, and Q_f is the friction torque.

Propeller thrust and torque. A controllable pitch propeller (CP) has blades that can be turned by means of a hydraulic mechanism. The propeller pitch ϑ can be changed from $100\,\%$ (full ahead) to $-100\,\%$ (full astern).

The propeller thrust and torque are determined by the following bilinear relations:

$$T_{\text{prop}}(t) = T_{|n|n}(\vartheta)\,|n(t)|n(t) + T_{|n|V_a}(\vartheta(t))\,|n(t)|V_a(t) \tag{10.19}$$

$$Q_{\text{prop}}(t) = Q_{|n|n}(\vartheta)\,|n(t)|n(t) + Q_{|n|V_a}(\vartheta(t))\,|n(t)|V_a(t). \tag{10.20}$$

V_a is the velocity of the water passing through the propeller disc

$$V_a(t) = (1 - w)U(t),$$

where w is a hull-dependent parameter called the wake fraction. The coefficients $T_{|n|n}$, $T_{|n|V_a}$, $Q_{|n|n}$ and $Q_{|n|V_a}$ are complex functions of the pitch $\vartheta(t)$. T_{prop} and Q_{prop} are calculated by interpolating between tables of data measured in model propeller tests. Figure 10.31 shows graphically the dependencies of T_{prop} and Q_{prop} on n and V_a for different values of the pitch. K_T and K_Q, in these figures denote thrust and torque coefficients

$$T_{\text{prop}} = K_T\,\rho\,D^4\,|n|n \tag{10.21}$$

$$Q_{\text{prop}} = K_Q\,\rho\,D^5\,|n|n,$$

where D is the propeller diameter and ρ the mass density of water.

Ship speed dynamics. The following non-linear differential equation approximates the ship speed dynamics:

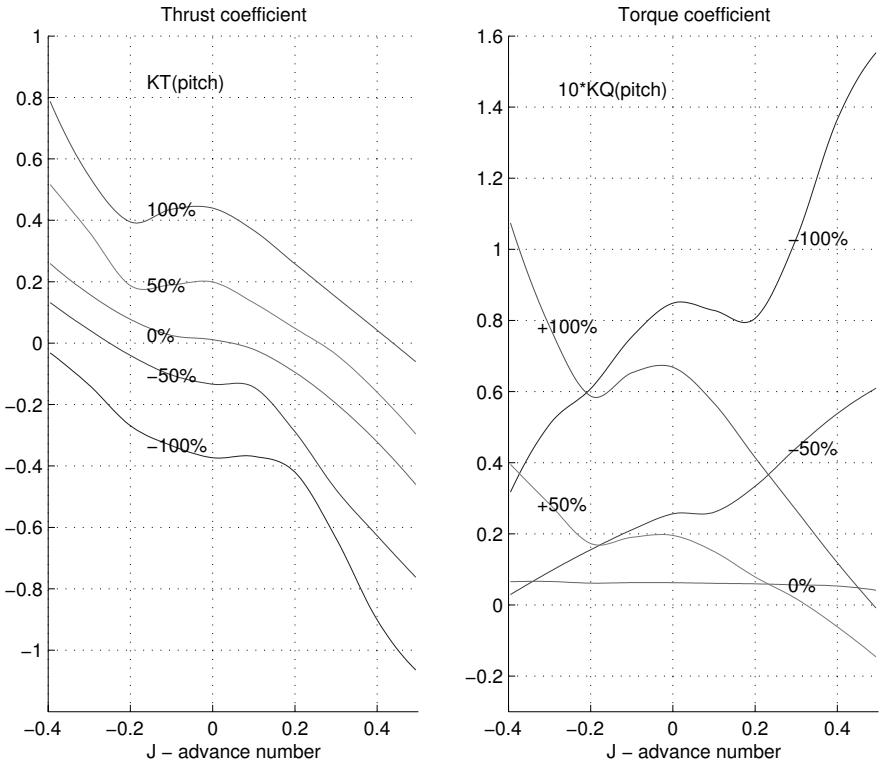


Fig. 10.31. Propeller torque as functions of advance number $J = 2\pi V_a/n$ for different values of pitch

$$\begin{aligned}(m - X_{\dot{U}})\dot{U}(t) &= R(U(t)) + (1 - t_T)T_{\text{prop}}(t) + T_{\text{ext}}(t) \\ U_m(t) &= U(t) + \nu_U(t)\,.\end{aligned} \tag{10.22}$$

The term $R(U)$ describes the resistance of the ship in the water. It is a negative quantity. Figure 10.32 shows the hull resistance as a function of the speed for two given load conditions. $X_{\dot{U}}$ represents the added mass in surge, which is negative. The thrust deduction $1 - t_T$ represents the net thrust lost due to the propeller-generated flow at the ship's stern. T_{ext} is the external force brought about by the wind and the waves. ν_U is the measurement noise.

10.3.3 Fault scenarios and requirements on the diagnosis

Fault scenario. The faults are summarised in Table 10.9.

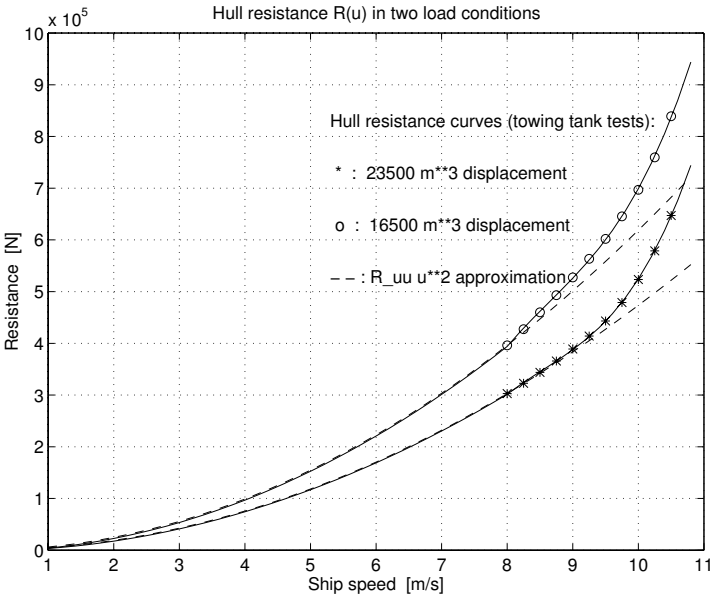


Fig. 10.32. The magnitude of the hull resistance in dependence upon the ship speed and its loading conditions. The commonly adopted square law curve is not valid in the relevant range of operation (8.5 m/s to 10.5 m/s)

Table 10.9 Faults considered

Fault	Symbol	Type
Sensor faults	$\Delta\vartheta$	additive - abrupt
Hydraulic leak	$\Delta\dot{\vartheta}_{\mathrm{inc}}$	additive - incipient
Sensor faults	Δn	additive - abrupt
Diesel fault	ΔK_y	multiplicative - abrupt

A formal analysis of fault propagation shows that they have different degrees of severity. Some are very serious and need rapid fault detection and accommodation to avoid serious accidents if the component failure occurs during a critical manoeuvre. The time to detect and reconfigure is hence essential. Some of the faults described are based on actual events that have caused serious damage due to the lack of fault-tolerant features in existing propulsion control systems. Figure 10.33 locates the generic faults of the benchmark in the system block diagram. The faults are:

1. Faults $\Delta\vartheta$ related to the propeller pitch:
   - $\Delta\vartheta_{\mathrm{high}}$: This fault can occur due to an electrical or mechanical defect in the pitch sensor or its interface.
   - $\Delta\vartheta_{\mathrm{low}}$: This fault can occur due to an electrical or mechanical fault in the pitch sensor or its interface.

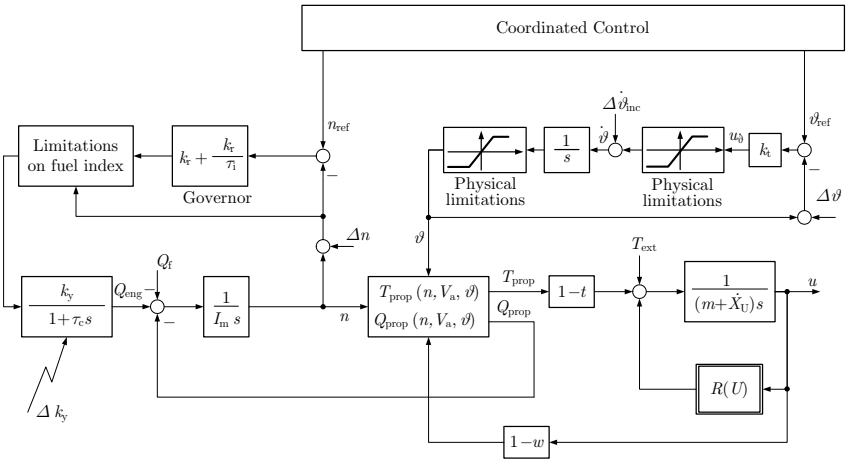


Fig. 10.33. Block diagram of the propulsion system with saturation phenomena shown for shaft speed and pitch controller. The generic faults of the benchmark are indicated.

- $\Delta\dot{\vartheta}_{\mathrm{inc}}$: A leakage can occur in the (hydraulic) actuation part of the control system; in practice, often in an over-pressurised valve.

2. Faults Δn related to shaft speed measurement. A dual pulse pick-up is used for measuring the shaft speed. The followings faults are considered:
   - a maximum signal Δn_{high} (disturbance on one pick-up), and
   - a minimum signal Δn_{low} (loss of both pick-up signals).
3. Faults related to the diesel engine (ΔK_y). The generated shaft torque can be lower than expected for the following reasons: reduced air inlet, reduced fuel oil inlet, or a cylinder is not working.

To determine how faults affect the system operation, the fault-propagation analysis methodology presented in Chapter 4 was employed. This analysis shows the end-effects for each fault. Combining end effects, the severity level for each fault was assessed. The results are shown in Table 10.10.

Table 10.10 Consequences and severity levels for the benchmark faults

Fault	Consequence	Severity
$\Delta\vartheta_{\text{high}}$	$deceleration \Rightarrow manoeuvring\ risk$	high
$\Delta\vartheta_{\text{low}}$	$acceleration \Rightarrow collision\ risk$	very high
$\Delta\dot{\vartheta}_{\text{inc}}$	$gradual\ speed\ change \Rightarrow cost\ increase$	medium
Δn_{high}	$deceleration \Rightarrow manoeuvring\ risk$	high
Δn_{low}	$acceleration \Rightarrow collision\ risk$	very high
ΔK_y	$diesel\ overload \Rightarrow wear, slowdown$	medium

Requirements on the diagnosis. The requirements for our diagnostic methods are as follows, where T_s is the sampling time in the particular control loop:

- *Time-to-detect* (T_d): For the sensor feedback faults ($\Delta\vartheta_{\text{low}}$, $\Delta\vartheta_{\text{high}}$, Δn_{low}, and Δn_{high}): $T_d < 2T_s$, the incipient fault $\Delta\dot{\vartheta}_{\text{inc}} : T_d < 100\,T_s$, the gain fault Δk_y : $T_d < 5T_s$.
- *Unknown input*: A time-varying external drag force from weather and shallow water is a potential source of false detection. Diagnosis should be insensitive to this.
- *False detection probability:* $P_f < 0.01$. Fault-free real data, including harbour manoeuvres, are provided to allow realistic testing of algorithms with respect to false detection.
- *Missed-detection probability:* $P_m < 0.001$. Due to the high severity level of faults, which can result in endangering the ship (and its crew), the probability of not detecting them when they do occur should be as low as possible.
- *Robust design:* Several sources of model uncertainty exist: slowly increasing hull resistance $R(U)$ due to growth ($0\,\%$ increasing to $20\,\%$ of $R(U)$), or varying external force from sea and wind ($\pm 10\,\%$ of $R(U)$), *a-priori* uncertainty in propeller thrust and torque ($\pm\,10\,\%$) and engine friction (from $5\,\%$ to $8\,\%$), and general uncertainty on other physical parameters ($\pm\,2\,\%$). Fault diagnosis should be robust to these. *A-posteriori* data for parameters may be identified and used for diagnosis.

Requirements on fault handling. Fault handling should incorporate appropriate steps to accommodate the faults. The remedial actions should primarily use the re-

configuration at the coordination level. Performance in a reconfigured mode can be lower than under faultless conditions. Large transients should be avoided when changing to a reconfigured mode. Bump-less transfer is not required, but is a desirable feature.

Test scenario. The test sequences constitute recordings from manoeuvres with the intercity ferry MF Dr. Ingrid. Faults are superimposed on the recorded data using the high-fidelity simulation model. Time stamps for different fault events are given in Table 10.11. The total simulation time is 3500 seconds. The fault ΔK_y corresponds to a $20\,\%$ drop in the diesel engine gain K_y.

Diagnosis. The main task is to find fault diagnosic algorithms that make it possible to detect and isolate the faults mentioned above. The algorithms should be robust to model uncertainty, load changes, and external forces.

The known and measured variables are propeller pitch set-point $\vartheta_{\mathrm{ref}}(t)$, propeller pitch measurement $\vartheta_m(t)$, shaft speed set-point $n_{\mathrm{ref}}(t)$, shaft speed measurement $n_m(t)$, ship speed $U_m(t)$, and the fuel index $Y_m(t)$. The data are obtained by sampling every 1 second.

Table 10.11 Fault events and their corresponding activation time intervals

Event	Start time	End time
$\Delta\vartheta_{\mathrm{high}}$	$180\,\mathrm{s}$	$210\,\mathrm{s}$
$\Delta\dot{\vartheta}_{\mathrm{inc}}$	$800\,\mathrm{s}$	$1700\,\mathrm{s}$
$\Delta\vartheta_{\mathrm{low}}$	$1890\,\mathrm{s}$	$1920\,\mathrm{s}$
Δn_{high}	$680\,\mathrm{s}$	$710\,\mathrm{s}$
Δn_{low}	$2640\,\mathrm{s}$	$2670\,\mathrm{s}$
ΔK_y	$3000\,\mathrm{s}$	$3500\,\mathrm{s}$

10.3.4 Structural analysis of the propulsion system

After having made an assessment of the set of high severity faults that need to be handled to obtain a fault-tolerant propulsion system, the next step is to provide relations for use in design of residual generators. Structural analysis is employed for this purpose. A prerequisite for structural analysis is that the set of constraints are listed. For the open loop system, involving the shaft and ship dynamics, the related constraints are the following:

$$
\begin{array}{lcll}
c_1 & : & c_1(\vartheta, \vartheta_m) = 0 & : \quad \vartheta = \vartheta_m \\
c_2 & : & c_2(n, n_m) = 0 & : \quad n = n_m \\
c_3 & : & c_3(Y, Y_m) = 0 & : \quad Y = Y_m \\
c_4 & : & c_4(k_y, K_y) = 0 & : \quad k_y = K_y \\
c_5 & : & c_5(Y, k_y, Q_{\text{eng}}) = 0 & : \quad Q_{\text{eng}} + \tau_c \dot{Q}_{eng} = k_y Y \\
c_6 & : & c_6(\dot{Q}_{eng}, Q_{\text{eng}}) = 0 & : \quad \dot{Q}_{eng} = \frac{dQ_{\text{eng}}}{dt} \\
c_7 & : & c_7(Q_{\text{eng}}, Q_{\text{prop}}, n) = 0 & : \quad I_m \dot{n} = Q_{\text{eng}} - Q_{\text{prop}} \\
c_8 & : & c_8(\dot{n}, n) = 0 & : \quad \dot{n} = \frac{dn}{dt} \\
c_9 & : & c_9(n, \vartheta, U, Q_{\text{prop}}) = 0 & : \quad Table^1 \\
c_{10} & : & c_{10}(n, \vartheta, U, T_{\text{prop}}) = 0 & : \quad Table^2 \\
c_{11} & : & c_{11}(\dot{U}, R(U), T_{\text{prop}}) = 0 & : \quad m\dot{U} = R(U) - (1 - t_T) T_{\text{prop}} \\
c_{12} & : & c_{12}(R(U), U) = 0 & : \quad Table^3 \\
c_{13} & : & c_{13}(\dot{U}, U) = 0 & : \quad \dot{U} = \frac{dU}{dt} \\
c_{14} & : & c_{14}(U, U_m) = 0 & : \quad U = U_m
\end{array}
\tag{10.23}
$$

The propeller developed torque $Q_{\text{prop}}(t)$ and trust $T_{\text{prop}}(t)$ are functions of $n(t)$, $\vartheta(t)$, and $U(t)$, and are calculated by interpolating between data that are described in the given tables. For the sake of clarity, the following simplifications are made:

- measurement noises in the system are not represented,
- disturbances, Q_f and T_{ext} are disregarded,
- $X_{\dot{U}}$ is negligible and the trust deduction $1 - t_T$ is known.

Furthermore, the propeller pitch dynamic is described by the following constraints:

$$
\begin{array}{lcll}
c_{15} & : & c_{15}(u_{\dot{\vartheta}}, \vartheta_{\text{ref}}, \vartheta_m) = 0 & : \quad u_{\dot{\vartheta}} = k_t(\vartheta_{\text{ref}} - \vartheta_m) \\
c_{16} & : & c_{16}(u_{\dot{\vartheta}}, \dot{\vartheta}) = 0 & : \quad u_{\dot{\vartheta}} = \dot{\vartheta} \\
c_{17} & : & c_{17}(\dot{\vartheta}, \vartheta) = 0 & : \quad \dot{\vartheta} = \frac{d\vartheta}{dt}
\end{array}
\tag{10.24}
$$

These constraints are valid during normal operational conditions when the control and system's physical limits are not violated.

The system structure is

$$
\begin{aligned}
\mathcal{S} &= \bigcup_{i=1}^{17} c_i, \\
\mathcal{C} &= \{c_1, c_2, \cdots, c_{17}\}, \\
\mathcal{K} &= \{\vartheta_m, n_m, Y_m, U_m, K_y, \vartheta_{\text{ref}}\}, \\
\mathcal{X} &= \{U, \vartheta, n, Y, k_y, \dot{Q}_{\text{eng}}, Q_{\text{eng}}, Q_{\text{prop}}, T_{\text{prop}}, u_{\dot{\vartheta}}, R(U), \dot{U}, \dot{n}, \dot{\vartheta}\}, \\
\mathcal{Z} &= \mathcal{K} \bigcup \mathcal{X}.
\end{aligned}
$$

As the measurement noise is disregarded in analysis of structure, the relations $\vartheta_m = \vartheta$ and $n = n_m$ etc. hold. A bi-partite graph representation of the system structure is depicted in Fig. 10.34.

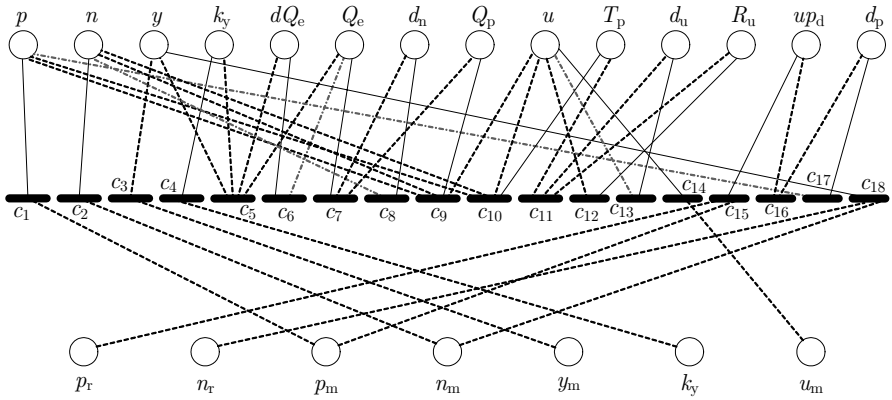


Fig. 10.34. Structural model of the ship propulsion system. The matching is illustrated by the thick lines.

Matching. The edges of a matching are identified by a thick line in the graph representation (Fig. 10.34) and by 'o' in the following incidence matrix.

As it is illustrated, a complete matching with respect to the unknown variables are obtained. There are three unmatched constraints that can be used for detecting the faults. Each unmatched constraint involves a different part of the system as follows:

Subsystem 1: Engine dynamics and propeller torque characteristics. The unmatched constraint c_5 involves engine and shaft dynamics and the propeller torque characteristics. The involved constraints are:

$$
\begin{aligned}
c_1 &: \vartheta(t) = \vartheta_m(t) \\
c_2 &: n(t) = n_m(t) \\
c_3 &: Y(t) = Y_m(t) \\
c_4 &: k_y(t) = K_y(t) \\
c_5 &: Q_{\mathrm{eng}}(t) + \tau_c \dot{Q}_{\mathrm{eng}}(t) = k_y(t) Y(t) \\
c_6 &: \dot{Q}_{\mathrm{eng}}(t) = \frac{dQ_{\mathrm{eng}}(t)}{dt} \\
c_7 &: I_m \dot{n}(t) = Q_{\mathrm{eng}}(t) - Q_{\mathrm{prop}}(t) \\
c_8 &: \dot{n}(t) = \frac{dn(t)}{dt} \\
c_{14} &: U(t) = U_m(t)
\end{aligned}
$$

The faults that affect the dynamics of this subsystem are obviously faults in pitch and shaft measurements as well as the fault in the diesel engine.

Subsystem 2: Ship speed dynamics and propeller thrust characteristics. The unmatched constraint c_{11} involves ship speed dynamics and the propeller thrust

$\leftrightarrow$	ϑ_{m}	n_{m}	Y_{m}	U_{m}	K_{y}	ϑ_{ref}	U	ϑ	n	Y	k_{y}	$\dot{Q}_{\mathrm{eng}}$	Q_{eng}	Q_{prop}	T_{prop}	$R(U)$	$\dot{U}$	$\dot{n}$	u_Φ	$\dot{\vartheta}$
c_1	1							①												
c_2		1							①											①
c_3			1							①										
c_4					1						①									
c_5										1	1	1	1							
c_6												①	x							
c_7													①	1					1	
c_8									x										①	
c_9							1	1	1					①						
c_{10}							1	1	1						①					
c_{11}															1	1	1			
c_{12}							1										①			
c_{13}							x											①		
c_{14}				1			①													
c_{15}	1					1													①	
c_{16}																			1	①
c_{17}								x												1

Fig. 10.35. Incidence matrix of the ship example

characteristics. The involved constraints are:

$$\begin{aligned}
c_1 &: \vartheta(t) = \vartheta_m(t) \\
c_2 &: n(t) = n_m(t) \\
c_{10} &: Table^2 \\
c_{11} &: m\dot{U}(t) = R(U(t)) - (1 - t_T)T_{\text{prop}}(t) \\
c_{13} &: \dot{U}(t) = \frac{dU(t)}{dt} \\
c_{14} &: U(t) = U_m(t)
\end{aligned}$$

The involved dynamics will be affected by fault in shaft speed and pitch measurements.

Subsystem 3: Propeller pitch dynamics. The unmatched constraint c_{17} involves conctraints related to the propeller pitch dynamics. The involved constraints are:

$$\begin{aligned}
c_1 &: \vartheta(t) = \vartheta_m(t) \\
c_{15} &: u_{\dot{\vartheta}}(t) = k_t(\vartheta_{\text{ref}}(t) - \vartheta_m(t)) \\
c_{16} &: u_{\dot{\vartheta}}(t) = \dot{\vartheta}(t) \\
c_{17} &: \dot{\vartheta}(t) = \frac{d\vartheta(t)}{dt}
\end{aligned}$$

This subsystem dynamics will be affected by fault in pitch measurement and the hydraulic fault.

10.3.5 Fault diagnosis using the parity space approach and state observation

Residual generation. As it has been extensively argued in previous chapters, the parity space and observer-based approaches are commonly used to generate residuals. In this section, these approaches are applied to the propulsion system to obtain an expression for residuals.

Subsystem 3. Using the relevant constraints, a parity equation can be set up, which involves only known variables of this subsystem. The obtained parity equation is:

$$\frac{d\vartheta_m(t)}{dt} = k_t(\vartheta_{\text{ref}}(t) - \vartheta_m(t)).$$

Based on this, a residual can be defined as

$$r_\vartheta(t) = \frac{d\vartheta_m(t)}{dt} - k_t(\vartheta_{\text{ref}}(t) - \vartheta_m(t)).$$

The residual should have a vanishing mean value under normal conditions, i.e. when no sensor fault has occurred. The residual's dynamical behaviour shall change in the presence of faults ($\Delta\vartheta_{\text{high}}$, $\Delta\vartheta_{\text{low}}$, and $\Delta\dot{\vartheta}_{\text{inc}}$).

Subsystem 1. Since sensor measurements for propeller pitch, shaft speed and ship speed are available, the value of the propeller torque Q_{prop} can be computed means of $Table^1$. Hence, $Q_{\text{prop}}(\vartheta_m,\, n_m,\, U_m)$ is known. Through constraint c_7 we obtain

$$Q_{\text{eng}}(t) = I_m \dot{n}_m(t) - Q_{\text{prop}}(\vartheta_m(t), n_m(t), U_m(t))$$

Now, it is possible to set up a parity equation by using the constraints c_3, c_4, c_5, and c_6, i.e.

$$\begin{aligned} K_y Y_m(t) &= Q_{\text{eng}}(t) + \tau_c \frac{dQ_{\text{eng}}(t)}{dt} \\ &= I_m \dot{n}_m(t) - Q_{\text{prop}}(\vartheta_m(t), n_m(t), U_m(t)) \\ &+ \tau_c \frac{d(I_m \dot{n(t)}_m - Q_{\text{prop}}(\vartheta_m(t), n_m(t), U_m(t)))}{dt}. \end{aligned}$$

A residual expression can be defined, by using the obtained parity equation, in a straightforward manner:

$$\begin{aligned} r_Q(t) &= I_m \dot{n(t)}_m - Q_{\text{prop}}(\vartheta_m(t), n_m(t), U_m(t)) \\ &+ \tau_c \frac{d(I_m \dot{n(t)}_m - Q_{\text{prop}}(\vartheta_m(t), n_m(t), U_m(t)))}{dt} - K_y Y_m(t). \end{aligned}$$

Actual computation of this residual, in present from, requires employing numerical methods (for instance Euler method or central-difference formula of order 2). The computed residual should have a mean value equal zero under normal conditions. The residual's dynamical behaviour shall change in presence of sensor measurement faults ($\Delta\vartheta_{\text{high}}$, $\Delta\vartheta_{\text{low}}$, Δn_{high}, Δn_{low} and the engine gain fault ΔK_y.

Residual generator obtained by matching. This subsection employs the flexibility of structural analysis to generate a residual with desired properties. It would be desirable if the diesel engine gain fault could be detected independently of a fault in the shaft speed measurement. Since the constraint c_2 is not valid if the shaft speed fault is present, remove this constraint from the original set. Further, make the following simplifications: The diesel engine dynamics is much faster than the shaft speed and ship speed dynamics. This assumption is manifested by removing the constraint c_6 and changing the equation in the constraint c_5 to $Q_{\text{eng}} = k_y Y$.

Example 10.1 *Matching on revised system*

Matching on the open loop system that follows from removing c_2 and c_6 from the set of constraints in (10.23) and c_5 modified, makes us determine the unmatched constraints that are necessary to detect the diesel engine gain fault. This is left as an exercise.

The obtained fault-free subsystem can be written as:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{g}(\boldsymbol{x}(t)) + \boldsymbol{g}_u(\boldsymbol{x}(t))\boldsymbol{u}(t) \\ y(t) &= U(t) \end{aligned}$$

where

$$\boldsymbol{x}(t) = \begin{pmatrix} n(t) \\ U(t) \end{pmatrix}$$

$$
\begin{aligned}
\boldsymbol{g}(\boldsymbol{x}) &= \begin{pmatrix} 0 \\ \frac{1}{m}R(U) + \frac{1-t_T}{m}\,T_{|n|V_a}(1-w)nU \end{pmatrix} \\
\boldsymbol{g}_u(\boldsymbol{x}) &= \begin{pmatrix} \frac{1}{I_m}\,k_y & -\frac{1}{I_m}\,\left[Q_{|n|n}n^2 + Q_{|n|V_a}(1-w)nU\right] \\ 0 & \frac{1-t_T}{m}\,T_{|n|n}n^2 \end{pmatrix} \\
\boldsymbol{u}(t) &= \begin{pmatrix} Y_m(t) \\ \vartheta_m(t) \end{pmatrix}.
\end{aligned}
$$

The diesel engine dynamics is very fast (small τ_c), i.e. Eq. (10.17) has been used to employ a static relation for engine torque $Q_{eng} = (k_y + \Delta k_y)Y_m$. $Table^1$ and $Table^2$ are represented by their bilinear approximations (10.19) and (10.20). Q_0 is disregarded in the equations. □

Detection of the diesel engine gain fault ΔK_y. To return to the original notation of the ship dynamics the following notation is used for this subsystem

$$
\begin{aligned}
\dot{n}(t) &= \frac{1}{I_m}\,K_y\,Y_m - \frac{1}{I_m}\,\left[Q_{|n|V_a}(1-w)\,n\,U + Q_{|n|n}\,n^2\right]\,\vartheta \\
\dot{U}(t) &= \frac{1}{m}R(U) + \frac{1-t_T}{m}\,T_{|n|V_a}(1-w)\,n\,U + \frac{1-t_T}{m}\,T_{|n|n}\,n^2\,\vartheta \\
y(t) &= U(t)
\end{aligned}
$$

with the fuel index measurement Y_m and the pitch measurement ϑ_m as external input. For this system an observer can be given in the following form:

$$
\begin{aligned}
\dot{\hat{n}}(t) &= \frac{1}{I_m}\,K_y\,Y_m - \frac{1}{I_m}\,\left[Q_{|n|V_a}(1-w)\hat{n}\hat{U} + Q_{|n|n}\hat{n}^2\right]\,\vartheta_m + K^{\hat{n}}_{\Delta k_y}(U_m - \hat{U}) \\
\dot{\hat{U}}(t) &= \frac{1}{m}R(\hat{U}) + \frac{1-t_T}{m}\,[T_{|n|V_a}(1-w)\hat{n}\hat{U} + T_{|n|n}\hat{n}^2\vartheta_m] + K^{\hat{U}}_{\Delta k_y}(U_m - \hat{U}) \\
\hat{y}(t) &= \hat{U}(t).
\end{aligned}
$$

A residual can be obtained by using the output (ship speed estimate) of the observer and the ship speed measurement U_m in the following way:

$$r_{TQ} = U_m - \hat{U}.$$

The residual r_{TQ} is by construction only affected by the gain fault ΔK_y (when considering the pitch signal to be fault-free). As the observer is stable, the residual behaves in the fault-free case such that $r_{TQ} \to 0$ holds for $t \to \infty$. For the estimation errors

$$e^1_n(t) = n(t) - \hat{n}(t) \quad \text{and} \quad e^1_U(t) = U(t) - \hat{U}(t)$$

the following error dynamics can be given (with $\vartheta = \vartheta_m$):

$$
\begin{aligned}
\dot{e}^1_n = {} & \frac{1}{I_m}\,\Delta k_y\,Y - \frac{1}{I_m}\,\left[Q_{|n|V_a}(1-w)(nU - \hat{n}\hat{U}) + Q_{|n|n}(n^2 - \hat{n}^2)\right]\vartheta_m \\
& - K^{\hat{n}}_{\Delta k_y}(U_m - \hat{U}) \qquad (10.25) \\
\dot{e}^1_U = \dot{r}_1 = {} & \frac{1}{m}(R(U) - R(\hat{U})) + \frac{1-t_T}{m}[T_{|n|V_a}(1-w)(nU - \hat{n}\hat{U}) + \\
& + T_{|n|n}(n^2 - \hat{n}^2)\vartheta_m] - K^{\hat{U}}_{\Delta k_y}(U_m - \hat{U}). \qquad (10.26)
\end{aligned}
$$

Looking at the estimation error dynamics (10.25) one could think that an occurring gain fault ΔK_y would have a direct impact on $\dot{e}_n^1$ leading to a growing estimation error: $n \neq \hat{n}$. As can be seen from Eq. (10.26) this would then also affect the shaft speed estimate, i.e. $U \neq \hat{U}$. Hence, the first residual would deviate from zero in case of a gain fault: $r_{TQ} \neq 0$. However, with the coupled nonlinear equations at hand this argumentation is not correct. Simulation results given below show that the gain fault ΔK_y does indeed affect the residual r_{TQ}.

The observer offers also a second possibility to generate a residual when using the shaft speed measurement n_m:

$$r_{TQ_2}(t) = n_m(t) - \hat{n}(t).$$

Obviously, this residual is also affected by the shaft speed sensor fault Δn. The residual dynamics can be stated as follows

$$\dot{r}_{TQ_2}(n_m(t), \hat{n}(t)) = \dot{e}_n^1(t) + \dot{\Delta} n(t),$$

where $\dot{e}_n^1$ is described by Eq. (10.25).

Simulation results. The gains and initial conditions for the observer are chosen as follows:

$$\begin{aligned} K_{\Delta k_y}^{\hat{n}} &= 0.001, \\ K_{\Delta k_y}^{\hat{U}} &= 0.01, \\ \hat{n}(t=0) &= 9\ \mathrm{rad/s}, \\ \hat{U}(t=0) &= 0.1\ \mathrm{m/s}. \end{aligned}$$

The simulation result is shown in Fig. 10.36.

The second residual, i.e. r_{TQ_2}, is generated by choosing the gains and the initial conditions of the observer as:

$$\begin{aligned} K_{\Delta k_y}^{\hat{n}} &= 0.001, \\ K_{\Delta k_y}^{\hat{U}} &= 0.01, \\ \hat{n}(t=0) &= 9\ \mathrm{rad/s}, \\ \hat{U}(t=0) &= 0.1\ \mathrm{m/s}. \end{aligned}$$

The simulation result is shown in Fig. 10.37.

Comparing Fig. 10.36 and 10.37, it is clear that residual r_{TQ} is not affected by shaft sensor fault as it was the idea that was described in the start of section 10.3.5, whereas residual r_{TQ_2} is affected by both the shaft speed fault and the engine gain fault. Small deviations (from zero) in both residuals are due to the sudden change in measurement/input signals combined with the nonlinear behaviour of the system and can be handled by choosing an appropriate threshold.

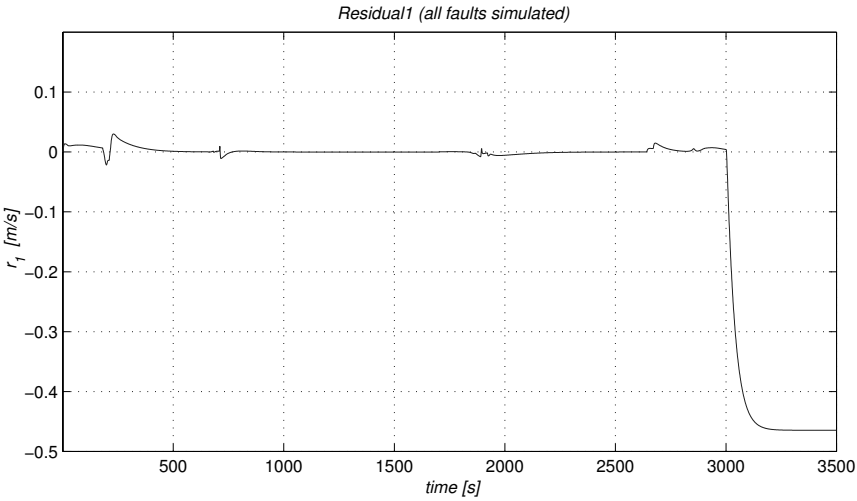


Fig. 10.36. Residual $r_{TQ} = U_m - \hat{U}$. Simulation including all faults and no measurement noise.

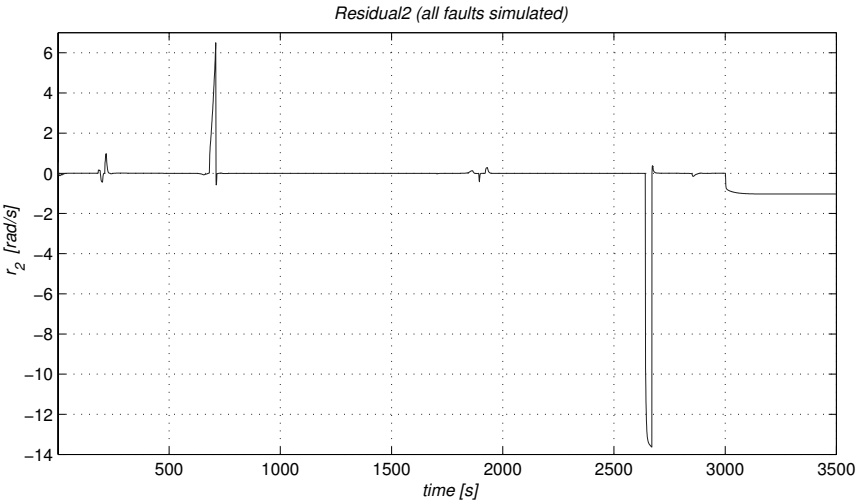


Fig. 10.37. Residual $r_{TQ_2} = n_m - \hat{n}$. Simulation including all faults and no measurement noise.

10.3.6 Quantised systems approach to the diagnosis of the pitch control loop

As an alternative approach to the diagnosis of the ship propulsion system, the quantised systems approach explained in Chapter 9 is applied to the pitch control loop, which is shown in Fig. 10.33 in the right upper corner. The loop is re-drawn as the block diagram shown in Fig. 10.38, where the input $\theta_{\rm ref}$ is represented by the left arrow and the measured pitch angle θ_m by the right arrow. The two faults, which affect this loop, are shown by the arrows labeled as $\Delta\theta$ and $\Delta\dot{\theta}_{inc}$ where the first

represents the sensor fault, which may have the two values $\Delta\theta_{low}$ or $\Delta\theta_{high}$, and the second a fault in the hydraulic system.

This figure depicts how the structure proposed in Fig. 9.3 on p. 452 is applied to a part of the ship propulsion problem. The following investigations will show that the faults change the behaviour of the pitch control loop in such a way that they can be detected by means of rough measurement sequences that result from a quantisation of the numeric data that represent the evolution of θ_{ref} and θ_m. This diagnostic method is robust against model uncertainties and measurement noise, because it is based on rough data.

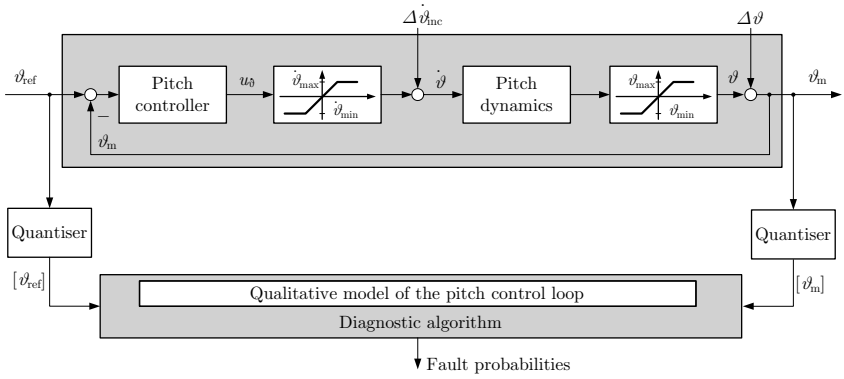


Fig. 10.38. Quantised systems approach to the diagnosis of the pitch control loop

Qualitative modelling of the pitch control loop. According to Section 9.4, the pitch control loop together with the quantisers can be described by a stochastic automaton, which is obtained by the abstraction algorithm. The quantitative model is given by Eqs. (10.12), (10.14). It is used here in a discrete-time version for the sampling time $T_s = 1\,\mathrm{s}$. The quantisers are chosen so as to get, on the one hand, a low number of symbolic input and output values but, on the other hand, to obtain enough information about the performance of the pitch control loop.

Figure 10.39 shows how the quantisation intervals of the measured pitch angle have been chosen. The quantisation of the input, which is the reference input for the measured pitch angle, is chosen accordingly. The size of the intervals are comparable with the size of the measurement noise. Due to this noise, a smaller quantisation resolution is useless. On the other hand, the diagnostic results will show that the resolution used is small enough for the diagnostic purposes, i.e., the quantised input and output sequences provide enough information for detecting and identifying the faults.

With these quantisers, the abstraction algorithm developed in Section 9.4.3 has been applied. The result is a stochastic automaton, which cannot be shown here due

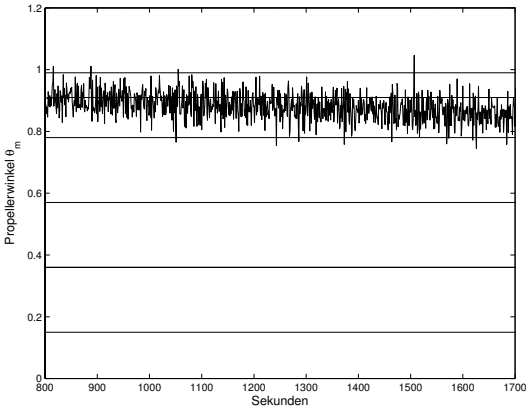


Fig. 10.39. Measured pitch angle in the quantised output space

to its size. This automaton is a complete model of the pitch control loop together with the chosen quantisers.

Diagnostic results. The diagnostic algorithm has been tested for the pitch control loop by using input and output sequences, which have been obtained by quantising input and output signals used in simulation studies of the ship propulsion system. In a long simulation run, the three different fault situations, which concern the pitch control loop, have been simulated during the time intervals given in the table.

Fault symbol	Numerical fault value	Activation time	Final time
$\Delta\theta_{high}$	1	180 s	210 s
$\Delta\theta_{low}$	-0.7	1880 s	1920 s
$\Delta\dot{\theta}_{inc}$	$\frac{0.00001}{s+0.0001}$	800 s	1700 s

Figure 10.40 shows the trajectory of the measured pitch angle θ_m in a time interval, in which the first fault was present. The diagnostic algorithm has no access to this measurement values, but obtains merely the quantised version of them. The figure is used here to explain the diagnostic result.

The diagnostic algorithm finds the fault $\Delta\theta = \Delta\theta_{high}$, which represents a positive measurement error, at once. This result is visible in Fig. 10.41, where the rectangle showing this fault (called "Sensorfault high") is black from the second time instant shown, which means that the diagnostic result associates with this fault a high probability (which is nearly one over the whole time interval where the fault is present).

At time 210 seconds, the fault disappears. Hence the measured pitch angle is much lower than before (cf. Fig. 10.40), which imitates the contrasting fault $\Delta\theta_{low}$. This behaviour explaines why the diagnostic algorithm associates with this fault $\Delta\theta_{low}$

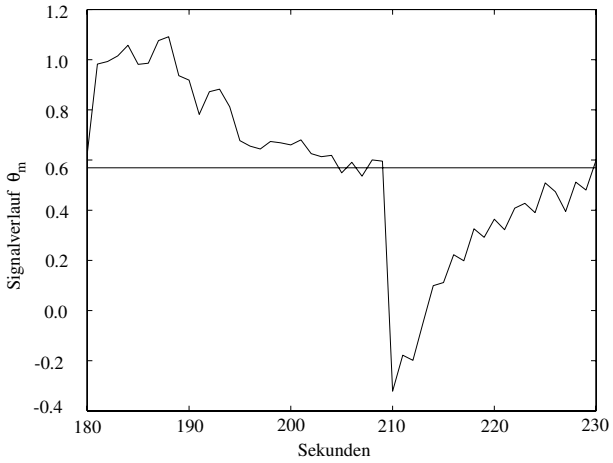


Fig. 10.40. Quantitative trajectory of the measured pitch angle

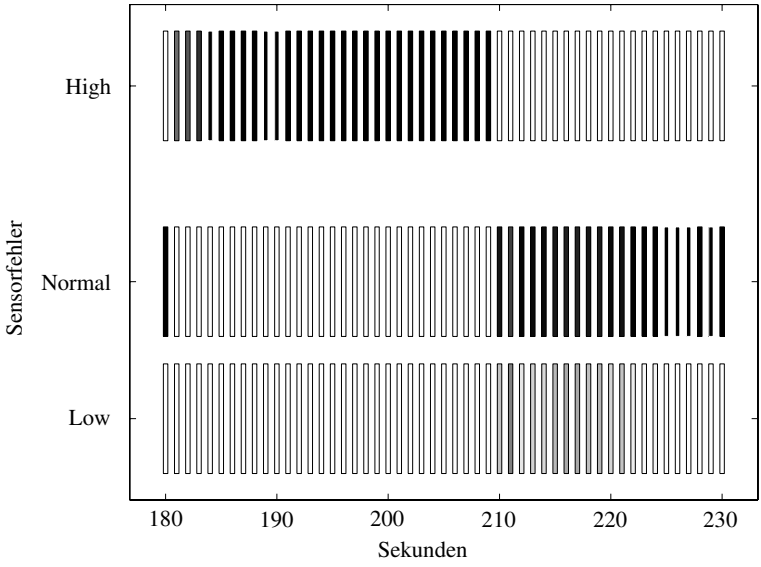


Fig. 10.41. Diagnostic result for sensor fault $\Delta\theta_{high}$ between 180 and 210 seconds

Fig. 10.42. Quantitative trajectory of the measured pitch angle for sensor fault $\Delta\theta_{low}$

a low (but positive) probability at the time instant 211. However, as the diagnostic algorithm does not only use the system output, but checks the consistency of the (quantised) input and output sequences with the stochastich automaton, it finds out that the decrease of the measurement values is the results of a disappearance of the fault $\Delta\theta_{high}$ and not due to the low-measurement fault $\Delta\theta_{low}$. Hence, the diagnostic algorithm gets the right result showing that the system is faultless.

For the second fault $\Delta\theta_{low}$, the pitch angle behaves approximately in the opposite way as before (cf. Fig. 10.42). The diagnostic algorithm finds this fault at once and behaves similarly as in the first fault case, when the second fault disappears at time instant 1920 seconds (Fig. 10.43).

Finally, the diagnosis of the pitch control loop subject to the hydraulic fault is considered. As Fig. 10.44 shows, the measured pitch angle changes very slowly. Hence, the effect of the fault is detectable rather late. Therefore, the figures are drawn with a much higher sampling time as before.

Figure 10.45 shows that the fault is detected at time around 1400 seconds.

The results show that the faults in the pitch control loop can by found by a diagnostic method that uses only quantised measurement signals. Both sensor faults are identified at once, whereas the hydraulic fault, which changes the system behaviour slowly, is identified rather late. If the hydraulic fault appears as an incipient fault (cf. Table 10.9), this method is not quick enough for its identification. However, even with precise numerical measurements, this fault turns out to be hard to detect (cf. Section 10.3.5).

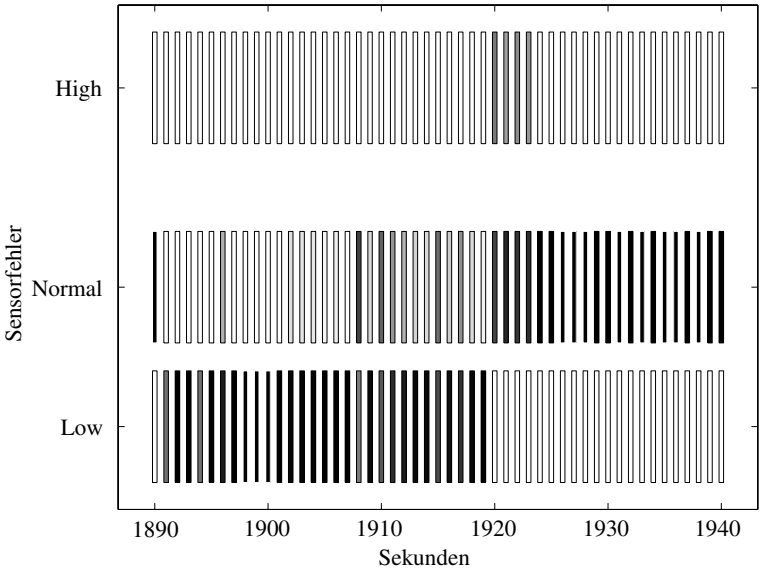


Fig. 10.43. Diagnostic result for sensor fault $\Delta\theta_{low}$ between 1880 and 1920 seconds

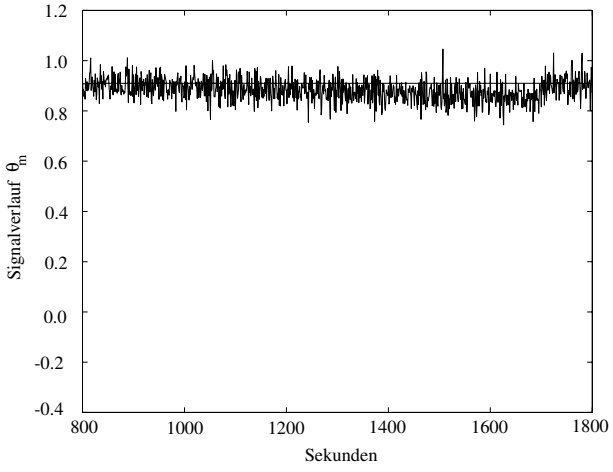


Fig. 10.44. Quantitative trajectory of the measured pitch angle for hydraulic fault $\Delta\dot{\theta}_{inc}$

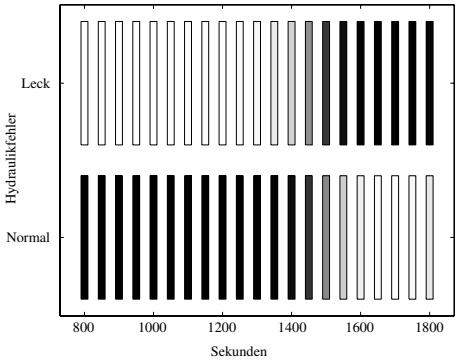


Fig. 10.45. Diagnostic result for hydraulic fault $\Delta\dot{\theta}_{inc}$ between 800 and 1700 seconds

10.3.7 Fault-tolerant propulsion

Active reconfiguration is achieved at the lower level of the ship propulsion system. Figure 10.46 shows how software redundancy, in the case where shaft speed measurement has failed, is implemented. The redundant module consists of the nonlinear observer that uses the fuel index Y_m and the propeller pitch measurement ϑ_m as input and the ship speed measurement, U_m, as the system output. As mentioned earlier, this observer is independent of faults in the shaft speed measurement. However, it can correctly provide an estimate of the shaft speed, which in turn is used as a (software) redundant information when a fault occurs.

The block containing the observer and fault isolation modules in Fig. 10.46 is the fault detection and isolation block and originally contains several residual generation and fault identification modules, which are not all explained here.

A shaft speed sensor fault occurs at time $680\,\mathrm{s}$ Fig 10.47 shows a zoom of the time responses of the shaft speed and the fuel index when the fault occurs at the worst instant in time, during a transient behaviour caused by alteration of the handle command simultaneously with the diesel torque being close to the maximum limit. A statistical fault detection method (CUSUM) detects the sensor failing high at time 681 s and generates a fault event. The state in the decision logic changes to *n_faulty* and the fault is accommodated on 682^{th} s where the effector alters the faulty measurement signal with the estimate generated by the nonlinear observer.

The transient behaviour following the fault causes the overload controller rapidly to reduce the pitch angle. This rapid load reduction makes it difficult for the shaft speed controller to reduce the shaft speed. Even in this extreme case, the overshoot in shaft speed is some $5\,\%$, which is below the critical limit of over-speed shut down (is $9\,\%$) of the main engine. The transient thus has no critical effects and is certainly acceptable compared with the alternative, which would be instant loss of propulsion of the ship. In each of the figures, the curves represent the following cases: normal case (solid line), faulty case (dash dotted line), and re-configured case (dashed line), reference signal (dotted line). The resulting overshoot is now well below the critical

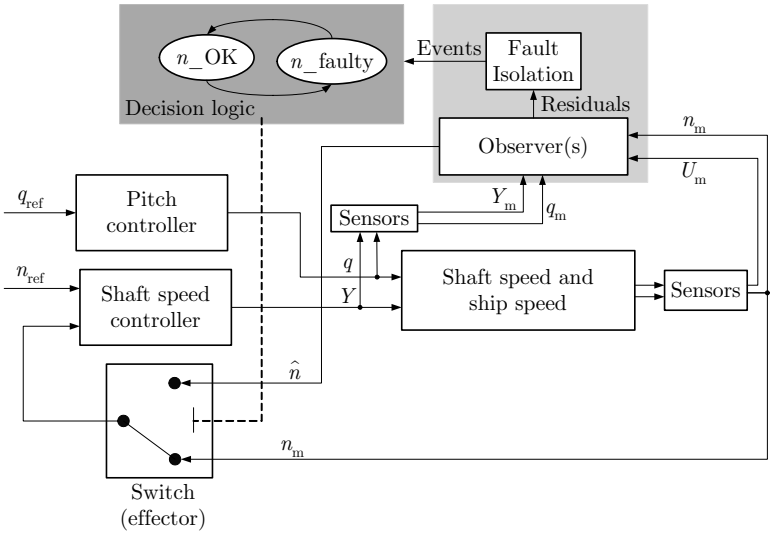


Fig. 10.46. Active reconfiguration scheme for the shaft speed when the non-linear observer is used

over-speed shut down limit and the effort of designing the adaptive observer was well spent.

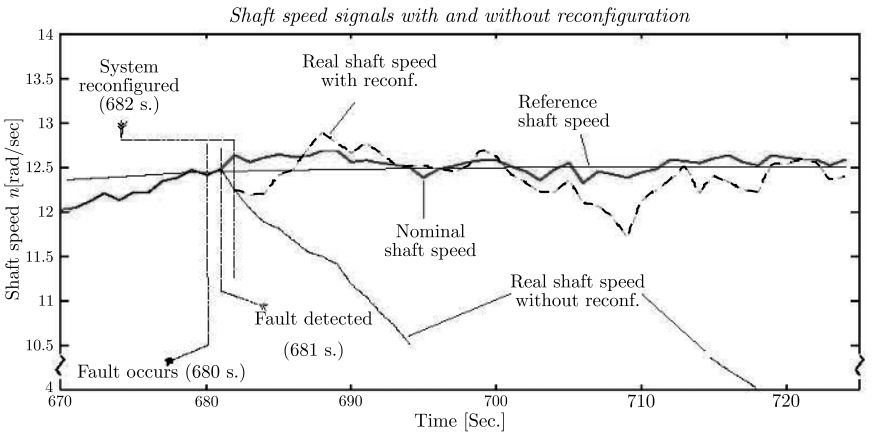


Fig. 10.47. A zoom of the shaft speed and fuel index values in the worst case, using the observer-generated shaft speed (Experiment 1)

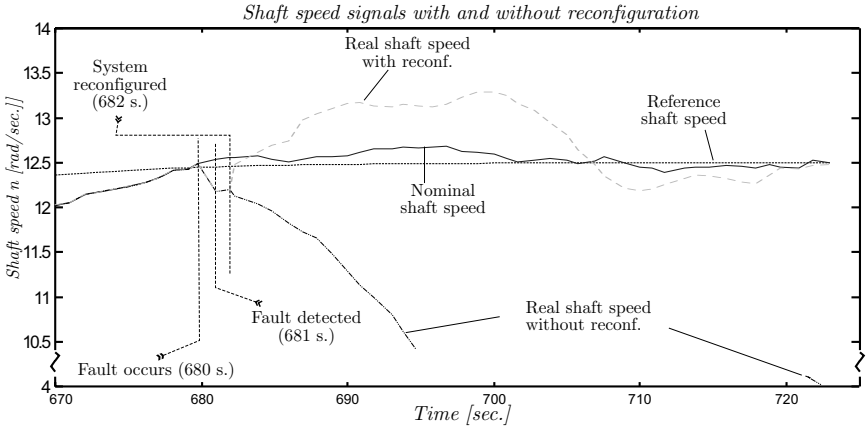


Fig. 10.48. A zoom of the shaft speed and fuel index values in the worst case, using the observer-generated shaft speed (Experiment 2)

10.4 Supervision of a steam generator

10.4.1 Description of the process

This section shows how a diagnostic algorithm can be found for a steam generator by using structural analysis.

The steam generator is a pilot process available at the University of Lille (France). A general view of the installation is given on Fig. 10.49.

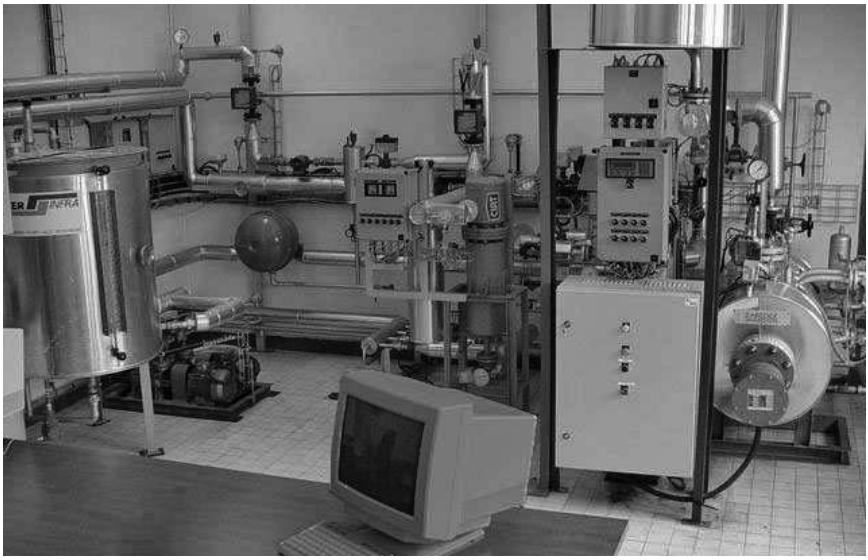


Fig. 10.49. General view of the steam generator

The energy of the primary loop is produced by an electric heating, while the energy of the steam (which would be really used in industrial applications, e.g. by providing it to a turbo-alternator in a power station), is here simply dissipated by a cooling system that is composed of a set of modulating valves and a condenser coupled with a heat exchanger.

In the following a part of the installation which is composed of the water feeding circuit and of the 175 litres boiler together with the heating is considered. The technological description is given by Fig. 10.50.

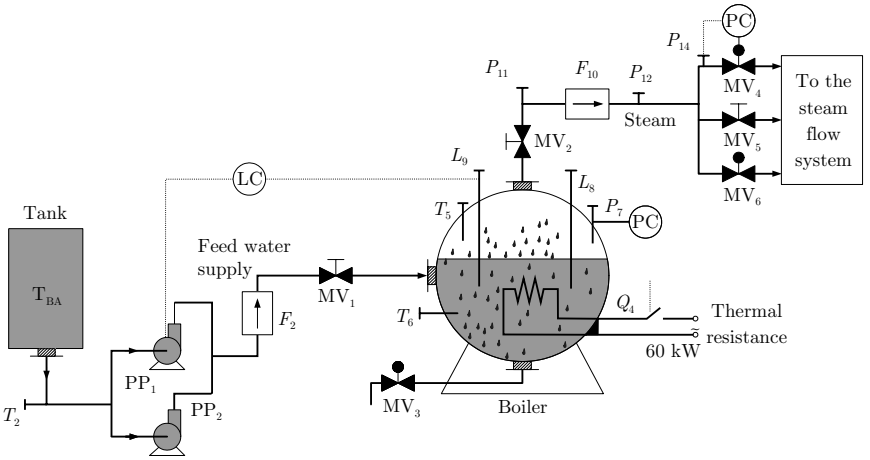


Fig. 10.50. Scheme of the process

The different system variables are labelled as:

F: massic flow (kg/s)
T: temperature ($^\circ C$)
P: pressure (bar)
L: level (litre).

Labels with a numerical index denote a sensor output while labels with a litteral index stand for the model variables. For example, P_7 is the output of the pressure sensor while P_{GV} is the pression as computed from the model. Thermal power (measured in J/s) which is transfered by convection or conduction, respectively, is labelled by $\dot{H}$ and $\dot{Q}$.

Water feeding the steam generator is stored in the tank at temperature T_{BA} and then fed into the boiler through the pump PP_1 at constant speed. Pump PP_2 is in parallel with PP_1 and stands as redundant hardware. The water level L_{GV} and the pressure P_{GV} in the boiler are measured by the sensors L_8, L_9 and P_7. They are regulated by the two on-off regulators PC (pressure control) and LC (level control). These controllers act on the heating power of the thermal resistor $\dot{Q}_{TH}$, which is measured by sensor Q_4, and on the flow of pump F_P which is measured by sensor

F_2. The produced steam, whose nominal flow is $F_{VG} = 60\,\mathrm{kg/h}$ (measured by sensor F_{10}), is subject to an isenthalpic depressurisation , due to a set of modulating valves $MV_{4,}$ MV_5 and MV_6, until the fixed pressure P_{VD} is reached. This pressure, measured by sensors P_{12} and P_{14} just ahead of the modulating valves, is regulated by a pressure controller (PC). The pressure drop between them being neglectible, the two measurements are redundant. The steam and saturated water temperatures in the boiler (both equal to T_{GV}), are measured by the thermocouples T_5 and T_6. The manual valve MV_3 on the emptying pipe of the boiler allows to create a water leakage in the steam generator, while the manual valve MV_2 on the output steam pipe allows to create a pipe clogging. MV_5 is a by-pass valve which also allows to create a steam leakage.

10.4.2 Modeling of the steam generator

The steam generator is a dynamical nonlinear system. It has energy of three domains, namely hydraulic energy (flows of fluids in pipes), electric energy (heating power), thermal energy (production of steam, thermal exchanges) and mechanical energy (pumps, valves). Three subsystems are distinguished:

1. the feedwater circuit (pump, valve, pipe)
2. the steam generation process
3. the thermal resistor .

The modelling hypotheses are as follows:

- Water and steam are saturated. Thermodynamical properties are calculated at equilibrium (this is justified by the assumption that the water-steam mixture is homogeneous).
- The water-steam mixture is at a uniform pressure P_{GV} i.e. the effect of the superficial tension of steam bubbles is neglected.
- More generally, all variables have uniform values, due to the small size of the boiler.
- The steam generator has known thermal capacity and it suffers from thermal losses by conduction to the external environment.
- The liquid in the feeding circuit is uncompressible.
- The produced steam is compressible.

Feedwater circuit. The feedwater circuit is composed of a number of pipes, of a hydraulic restriction, of two parallel pumps, and a manual valve MV_1. Water is pushed by the on-off controlled hydraulic pumps according to the water level L_{GV} in the steam generator.

Hydraulic model. The hydraulic model is intended to determine the water flow F_{AL0} in the boiler feeding pipe. This flow is obtained by the intersection of the

chraracteristics of the pump F_{PA} (given by the provider) and of the pipe F_{AL} (Fig. 10.51).

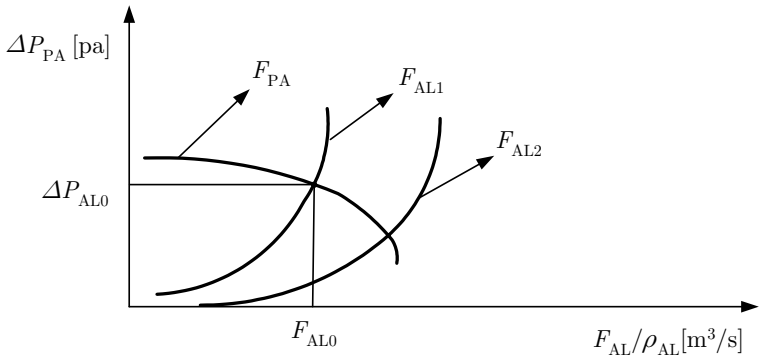


Fig. 10.51. Determination of the feeding flow

The curves F_{AL1} and F_{AL2} represent the pipes characteristics for different values of its hydraulic resistance. The pump characteristics is given as a function which links the pressure difference ΔP_{PA} (pascal) between the input and the output, and the volumic flow F_{PA} ρ_{AL}.

The determination of the massic flow F_{AL} as a function of the output pressure in the Tank P_{SB} and of the pressure in the steam generator P_{GV} just consists in solving the following system of equations:

$$\begin{cases} \frac{F_{AL}}{\rho_{AL}} = \left(-8,4948 \cdot 10^{-10} \Delta P_{PA} + 9,722 \cdot 10^{-4}\right) b_1 \\ \frac{F_{AL}}{\rho_{AL}} = \sqrt{\frac{(P_{PA}-P_{GV})10^5}{K_D(z_{V1})}}. \end{cases}$$

ΔP_{PA} is the pressure drop between the input and the output of the pump:

$$\Delta P_{PA} = 10^5 \left(P_{PA} - P_{SB}\right).$$

P_{SB} is the (known) pressure at the output of the tank and P_{PA} is the pressure at the output of the pump. The pressure drop between the tank and the input of the pump is negligible compared with the drop between the output of the pump and the boiler.

ρ_{AL} is the water density, P_{GV} the pressure inside the boiler and $K_D(z_{V1})$ the pressure drop coefficient associated with the configuration of the pipe (lenght, number of turns, etc). The latter also depends on the opening degree of the manual valve MV_1. b_1 is a Boolean variable associated with the output of the controller, which depends on the level difference between the water level in the steam generator L_{GV} and the reference water level L_{GV_ref} , and on Δ, the dead zone of the relay:

$$b_1 = F(\ \mathrm{sign}\ (L_{GV\_ref} - L_{GV}), \Delta).$$

Thermal model. Water is fed into the boiler through a calorifugated pipe, into which it arrives at ambiant temperature T_{Al}. The operating regime is discontinuous, and thus the thermal losses, by radiation, convection and conduction can be neglected. Therefore, the enthalpy flow $\dot{H}_{AL}$ (J/s) carried by the feedwater flow is equal to that at the output of the tank $\dot{H}_{SB}$:

$$\dot{H}_{AL} = \dot{H}_{SB} = F_{AL} \cdot h_{AL}.$$

The specific enthalpy of the water is h_{AL} (J/kg)

$$h_{AL} = c_{pe} T_{AL},$$

where c_{pe}, the specific heat of the water, is practically constant but depends weakly on the temperature and varies from 4180 to $4200\,\mathrm{J}\cdot\ \mathrm{kg}^{-1}.^{\circ}\mathrm{C}^{-1}$ for temperature between 15 and 100°C. Since the feedwater temperature varies between 30 and 60°C, the model is written as

$$\dot{H}_{AL} = 4200 F_{AL} T_{AL}.$$

It should be noticed that the state of the water has to be evaluated with respect to a basic reference temperature, which is taken at $T_0 = 0°C$. The specific enthalpy of liquid water is thus considered to be $h_{AL} = 0$ at $T_{AL} = 0$.

Hydraulic model of steam accumulation. The variation of the mass of the water-steam mixture M_{GV}(kg) in the boiler is

$$\frac{d}{dt}(M_{GV}) = \dot{M}_{GV} = F_{AL} - F_{VG},$$

where F_{VG} represents the massic flow of the steam at the output of the boiler, which flows through the manual valve MV_2.

Many models can be used to compute the massic flow F of a compressible fluid, depending on the operating regime of the physical process. A well known model is given by

$$\begin{aligned} F &= K_D \tfrac{P_1}{\sqrt{T_1}} \;\; if \;\; P_2 < 0.5 P_1 \\ F &= K_{D1} \sqrt{(P_1 - P_2)\, \tfrac{P_2}{T_1}} \qquad if\, P_2 > 0.5 P_1, \end{aligned} \tag{10.27}$$

where K_D is some flow coefficient, and P_1, T_1, T_2 and P_2 are respectively the pressures and temperatures at the input (the output) of the pipe restrictions. It is worth noticing that Bernoulli's law (which links the flow only with the difference of pressures) is valid only for uncompressible fluids.

In the present case, the pressures at the input and output of valve MV_2 are respectively the relative pressure within the steam generator P_{GV} (P_1) and the pressure at the input of the depressurisation valves P_{VD} (P_2). This pressure is measured by sensors P_{12} and P_{14}. Taking into account relation (10.27), it follows

$$F_{VG} = \sqrt{K_D(z_{V2}).\frac{(P_{GV} - P_{VD})\, P_{VD}}{T_{GV}}}. \tag{10.28}$$

In Equation (10.28), $K_D(z_{V2})$ is the pressure drop coefficient associated with the steam output pipe. It depends on the valve MV_2 position and it is experimentally determined.

Thermal model of the boiler. The energy balance equation in the steam generator is given by

$$\frac{d(H_{GV})}{dt} = \dot{H}_{GV} = \dot{Q}_{TH} + \dot{H}_{AL} + \dot{E}_W - \dot{H}_{VG} - \dot{Q}_{PG},$$

where h_{GV} or H_{GV} are, respectively, the specific enthalpy and the total enthalpy in the boiler, and $\dot{Q}_{TH}$ is the thermal heating power

$$\dot{Q}_{TH} = b_2.Q_4 = b_2.60000 \text{ W}.$$

This power is generated by a $60\,\text{kW}$ heating resistor, measured by sensor Q_4, and controlled by the on/off regulator, which acts on the pressure P_{GV} in the steam generator. b_2 (the relay output) is the Boolean control, which depends on the difference between the pressure in the steam generator P_{GV} and its reference value P_{GV_ref}, taking into account the relay's dead zone Δ.

$$b_2 = F(\text{ sign } (P_{GV\_ref} - P_{GV}), \Delta).$$

$\dot{E}_W$ is the power (J/s) provided by the work of the pressure forces, which is computed from the relation

$$\dot{E}_W = V_{GV}\dot{P}_{GV}$$
$$\dot{P}_{GV} = \frac{dP_{GV}}{dt}.$$

V_{GV} (m^3) is the geometric volume of the boiler, namely $0,175\,\text{m}^3$.

The total enthalpy flow carried by the steam is proportional to the specific enthalpy of the steam h_V and to its massic flow F_{VG}:

$$\dot{H}_{VG} = F_{VG} \cdot h_V.$$

$\dot{Q}_{PG}$ is the thermal power dissipated by conduction from the water-steam mixture to the metal body of the boiler. It is calculated below, in the thermal losses.

Modeling the thermal losses. Let c_{MG} be the thermal capacity of the metal body of the boiler. Two cases are in general considered when modelling such systems:

- In the first case, the thermal capacity is taken into account, but the isolation between the metal body and the ambiant atmosphere is considered as perfect, so that the losses are neglected.

- In the second case, radiation/conduction is present between the metal body and the outside world.

In the Lille process, experiments have shown that thermal losses (by radiation and conduction) could not be neglected, since several parts of the pipes, sensors, valves

are not calorifugated. Moreover, the mass of the metal body of the boiler cannot be neglected (102 kg).

The energy balance associated with the metal body of the boiler, whose volume is V_{MG} and density is ρ_{MG} is given by

$$\rho_{MG} V_{MG} c_{MG} \frac{dT_{MG}}{dT} = \dot{Q}_{PG} - \dot{Q}_{EX},$$

where

$$\dot{Q}_{PG} = K_{GM} \left(T_{GV} - T_{MG} \right).$$

$\dot{Q}_{EX}$ is the dissipated thermal power to the external world, whose temperature T_{EX} is supposed to be constant. Parameters ρ_{MG} and c_{MG} depend on the kind of metal which constitutes the body of the boiler

$$\dot{Q}_{EX} = K_{EX} \left(T_{MG} - T_{EX} \right).$$

K_{GM} and K_{EX} are heat exchange coefficients. K_{GM} depends on the length l_C of the pipe, of its external and internal diameters D_{CE} and D_{CL}, on the volumic mass ρ_V of the steam, on the thermal conductivity coefficient λ_V, on the dynamical viscosity μ_V, and on the radiation coefficient R_{AY}. K_{EX} depends on the ambiant atmosphere properties.

Description of the two-phase water-steam mixture. Two possibilities exist for modelling a two-phase mixture:

1. Write the balance equations for each of the two phases, and write the equations which model the exchanges between them. However, these equations are not well known, and this leads to a very huge number of equations.
2. Consider global equations which apply to the two phases simultaneously. This approach, although not very rigourous, is the only one which allows to obtain practical models, particularly because it involves the so-called *quality of the steam* X, which represents the ratio between the steam and the water in the mixture.

In a water-steam mixture, i.e. in the case of saturated steam, temperature and pressure are not independent. The pressure P_{GV} and the steam quality X are determined by the following mixture equation

$$\begin{cases} h_{GV} & = & \frac{H_{GV}}{M_{GV}} = h_V(P_{GV}) \cdot X + h_L(P_{GV}) \cdot (1 - X) \\ \\ \nu_{GV} & = & \frac{V_{GV}}{M_{GV}} = \nu_V(P_{GV}) \cdot X + \nu_L(P_{GV}) \cdot (1 - X), \end{cases}$$

where $h_L(P_{GV})$, $h_V(P_{GV})$, $\nu_L(P_{GV})$ and $\nu_V(P_{GV})$ are polynomial thermodynamical fonctions of the pressure P_{GV}, of the specific enthalpies h and of the massic volumes ν of the liquid and of the steam. They are determined (for a given operating regime) by a least squares optimisation approach from the water-steam equilibrium tables:

$$
\begin{aligned}
h_V(P_{GV}) &= -0,74P_{GV}^2 - 17,21P_{GV} + 2680 \\
h_L(P_{GV}) &= -0,0243P_{GV}^4 + 0,8487P_{GV}^3 - 11,9P_{GV}^2 - 99,97P_{GV} + 347 \\
\nu_V(P_{GV}) &= -5,3 \cdot 10^{-5}P_{GV}^5 + 0,00207P_{GV}^4 - 0,032P_{GV}^3 \\
&\quad +0,2498P_{GV}^2 - 1,03P_{GV} + 2,166 \\
\nu_L(P_{GV}) &= -3,59 \cdot 10^{-7}P_{GV}^3 + 1,2456 \cdot 10^{-5}P_{GV}^2 + 1,03 \cdot 10^{-3}
\end{aligned}
$$

This system thus allows to determine the steam quality X, the pressure in the water-steam mixture P_{GV} as well as the temperature T_{GV} using the following thermodynamical function:

$$T_{GV} = -0,4594P_{GV}^2 + 12,7243P_{GV} + 99,003.$$

10.4.3 Design of the diagnostic system

Specifications. Faults can occur in the process itself, or in the sensors and actuators, e.g. as a change in the parameters of some component. The specifications are intended to list those faults which have to be considered. Each fault is then associated with one or several equations of the system model, i.e. with variables or parameters of that model.

The diagnostic system will be associated with given detection and isolation quality levels, evaluated e.g. by the false alarm and missed-detection probabilities, the detection delay, the possibility to find out which fault really occured, among the different possible ones.

The main goal of a steam generator is to deliver a steam flow with given pressure and temperature. The first considered fault is thus associated with the output steam flow, i.e. with the (partial or total) clogging of the output pipe, which can be modelled as a change in the pressure drop coefficient K_D.

In industrial power plants it is important to completely separate the primary and the secondary loops. This means that no leakage is allowed. Therefore, the second considered fault is a leakage in the steam generator. Such a fault will influence the thermal energy H_{GV} and the mass M_{GV} accumulated in the boiler.

The measurements are essential for the control of the process, thus leading to consider sensor faults $\{T_2, F_3, Q_4, T_5, T_6, P_7, F_{10}, P_{11}, P_{12}, P_{14}\}$ in our list.

Design of the analytic redundancy relations. The system model can be decomposed into two subsets of relations. The behaviour of the process is described by the relations of the above model (RM), which obviously concern only letter-indexed variables. The knowledge available in real time about the behaviour is described by the relations RC which are just a way of showing which system variables are measured.

Behaviour model. From the previous equations, the behaviour model is as follows

$$
\begin{aligned}
\boldsymbol{RM1}: \quad & T_{GV} && = -0,4594P_{GV}^2 + 12,7243P_{GV} + 99,003 \\
\boldsymbol{RM2}: \quad & h_{GV} && = h_V X + h_L\,(1-X) \\
\boldsymbol{RM3}: \quad & \nu_{GV} && = \nu_V X + \nu_L(1-X) \\
\boldsymbol{RM4}: \quad & h_L && = -0,0243P_{GV}^4 + 0,8487P_{GV}^3 - 11,9P_{GV}^2 + \\
& && +99,97P_{GV} + 347 \\
\boldsymbol{RM5}: \quad & h_V && = -0,74P_{GV}^2 + 17,21P_{GV} + 2680 \\
\boldsymbol{RM6}: \quad & \nu_L && = -3,59\cdot 10^{-7}P_{GV}^3 + 1,2456\cdot 10^{-5}P_{GV}^2 + 1,03\cdot 10^{-3} \\
\boldsymbol{RM7}: \quad & \nu_V && = -5,3\cdot 10^{-5}P_{GV}^5 + 0,00207P_{GV}^4 - 0,032P_{GV}^3 \\
& && +0,2498P_{GV}^2 - 1,03P_{GV} + 2,166 \\
\boldsymbol{RM8}: \quad & \nu_{GV} && = \tfrac{V_{GV}}{M_{GV}} \\
\boldsymbol{RM9}: \quad & h_{GV} && = \tfrac{H_{GV}}{M_{GV}} \\
\boldsymbol{RM10}: \quad & F_{VG} && = \sqrt{K_D(z_{V2})\cdot \tfrac{(P_{GV}-P_{VD})P_{VD}}{T_{GV}}} \\
\boldsymbol{RM11}: \quad & \dot{M}_{GV} && = \tfrac{dM_{GV}}{dt} \\
\boldsymbol{RM12}: \quad & \dot{M}_{GV} && = F_{AL} - F_{VG} \\
\boldsymbol{RM13}: \quad & \dot{H}_{GV} && = \tfrac{dH_{GV}}{dt} \\
\boldsymbol{RM14}: \quad & \dot{H}_{GV} && = \dot{Q}_{TH} + \dot{H}_{AL} + \dot{E}_W - \dot{H}_{VG} - \dot{Q}_{PG}
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{RM15}: \quad & \dot{H}_{AL} && = 4200F_{AL}T_{AL} \\
\boldsymbol{RM16}: \quad & \dot{H}_{VG} && = F_{VG}.h_V \\
\boldsymbol{RM17}: \quad & \dot{Q}_{PG} && = K_{GM}\,(T_{GV} - T_{MG}) \\
\boldsymbol{RM18}: \quad & \rho_{MG} && V_{MG}c_{MG}\cdot \dot{T}_{MG} + T_{MG}\,(K_{EX} + K_{GM}) \\
& && = K_{GM}T_{GV} + K_{EX}T_{EX} \\
\boldsymbol{RM19}: \quad & \dot{T}_{MG} && = \tfrac{dT_{MG}}{dt} \\
\boldsymbol{RM20}: \quad & \dot{E}_W && = V_{GV}\dot{P}_{GV} \\
\boldsymbol{RM21}: \quad & \dot{P}_{GV} && = \tfrac{dP_{GV}}{dt}
\end{aligned}
$$

This model is a set of 21 constraints (nonlinear algebraic and differential equations), which link 25 unknown variables.

From the physical analysis of the process, the state vector is of dimension three. Two state variables are associated with the accumulation in the boiler, namely the thermal energy H_{GV} and the mass M_{GV}. The third state variable is associated with the accumulation of the thermal energy in the metal body of the boiler. It is here represented by the temperature T_{MG}.

In general, the initial conditions being unknown, only derivative causality can be used for the determination of the analytic redundancy relations. However, in this case, the initial conditions can be derived at each operation of the steam generator. Indeed, the initial mass $M_{GV}(0)$ of the mixture follows from the relation

$$
\begin{aligned}
M_{GV}(0) &= V_L(0)\rho_L(0) + V_V(0)\rho_V(0) \\
&= V_L(0)\rho_L(0) + [V_{GV} - V_L(0)]\rho_V(0),
\end{aligned}
$$

where $V_L(0)$ is the initial volume of the liquid in the boiler, which is fixed (for a given pressure $P_{GV}(0)$) by the reference of the level regulation system ($0.146\,\mathrm{m}^3$ in our application). The volume $V_V(0)$ of the steam then follows from V_{GV}, the total volume of the boiler, given by its geometry. The volumic masses of the steam, $\rho_V(0)$ and of the liquid $\rho_L(0)$ are determined from the thyermodynamical tables at pressure $P_{GV}(0)$.

The total initial enthalpy accumulated in the steam generator is then

$$\begin{aligned} H_{GV}(0) &= M_{GV}(0)h_{GV}(0) \\ &= M_{GV}(0)[h_V(0).X(0) + h_L(0)(1 - X(0))]. \end{aligned}$$

The initial specific enthalpy of the mixture, $h_{GV}(0)$, is determined as a function of the initial steam quality, $X(0)$ which is computed by

$$X(0) = \frac{M_V(0)}{M_V(0) + M_L(0)} = \frac{M_V(0)}{M_{GV}(0)}$$

and of the steam – $h_V(0)$ – and water – $h_L(0)$ – enthalpies, determined from thermodynamical tables. The initial temperature of the boiler body $T_{MG}(0)$ results from the differential equation $RM18$ in static regime, where the initial temperature in the boiler $T_{GV}(0)$ depends on the saturation pressure $P_{GV}(0)$

$$T_{MG}(0) = \frac{K_{GM}T_{GV}(0) + K_{EX}T_{EX}}{K_{GM} + K_{EX}}.$$

Sensors. The constraints associated with the sensors are the following:

$$\begin{array}{rrll} \boldsymbol{RC1}: & T_2 & = T_{AL} & (T_2,\ T_{AL}:\ ^\circ\,\mathrm{C}) \\ \boldsymbol{RC2}: & \frac{F_3}{3600} & = F_{AL} & (F_3:\ \mathrm{kg/h},\ F_{AL}:\ \mathrm{kg/s}) \\ \boldsymbol{RC3}: & Q_4 \cdot 1000 & = \dot{Q}_{TH} & (Q_4:\ \mathrm{kW}, \dot{\mathrm{Q}}_{TH}:\ \mathrm{W}) \\ \boldsymbol{RC4}: & T_5 & = T_{GV} & (T_5,\ T_{GV}:\ ^\circ\,\mathrm{C}) \\ \boldsymbol{RC5}: & T_6 & = T_{GV} & (T_6,\ T_{GV}:\ ^\circ\,\mathrm{C}) \\ \boldsymbol{RC6}: & P_7 & = P_{GV} & (P_7,\ P_{GV}:\mathrm{bar}) \\ \boldsymbol{RC7}: & \frac{F_{10}}{3600} & = F_{VG} & (F_{10}:\ \mathrm{kg/h},\ F_{VG}\ \mathrm{kg/s}) \\ \boldsymbol{RC8}: & P_{11} & = P_{GV} & (P_{11},\ P_{GV}:\ \mathrm{bar}) \\ \boldsymbol{RC9}: & P_{12} & = P_{VD} & (P_{12},\ P_{VD}:\ \mathrm{bar}) \\ \boldsymbol{RC10}: & P_{14} & = P_{VD} & (P_{14},\ P_{VD}:\ \mathrm{bar}). \end{array}$$

10.4.4 Structural analysis

The structural model is intended to analyse the structural properties of the system and to provide the means of creating residuals. Once these residuals have been identified, the behaviour model has to be used to determine the actual functions which have to be implemented for their real-time computation.

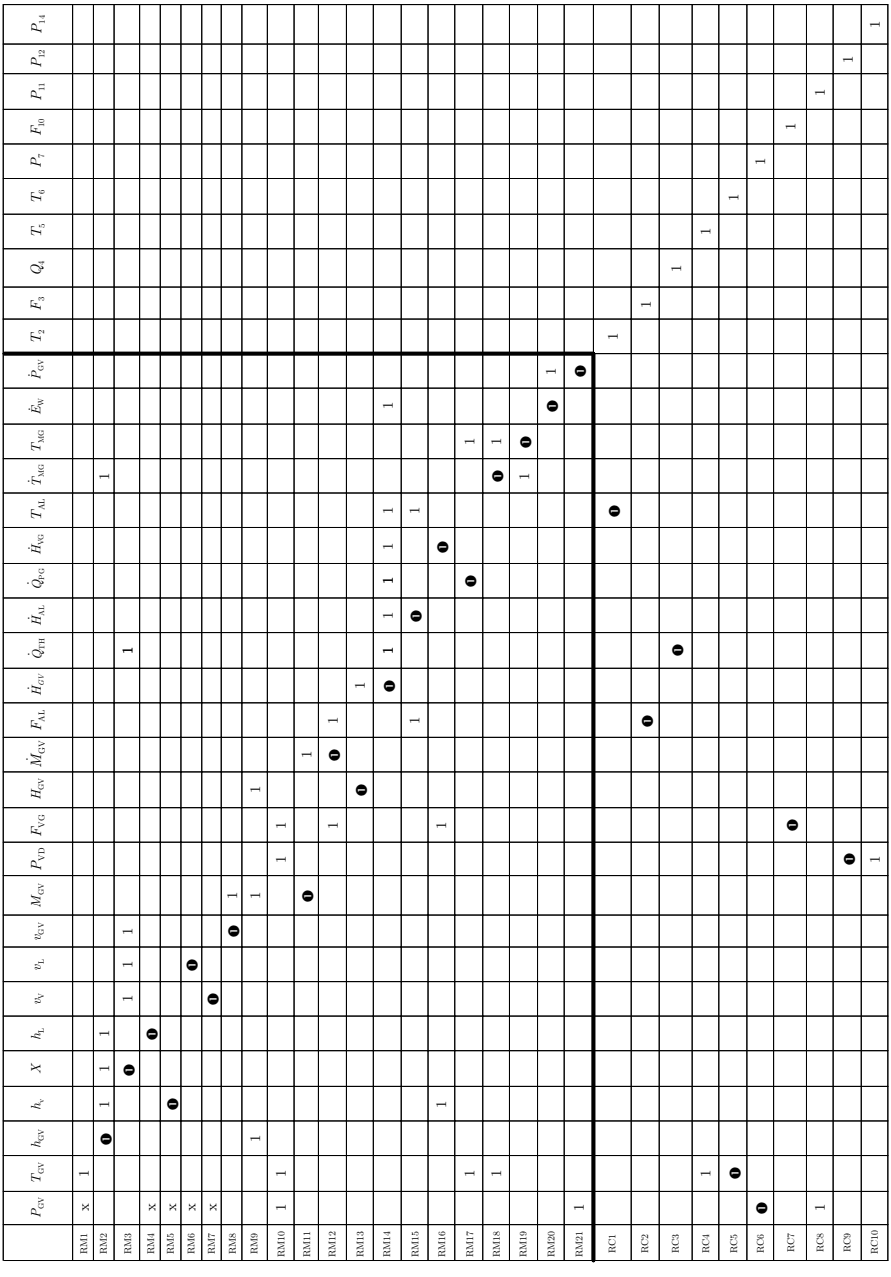


Fig. 10.52. Incidence matrix of the system structural graph

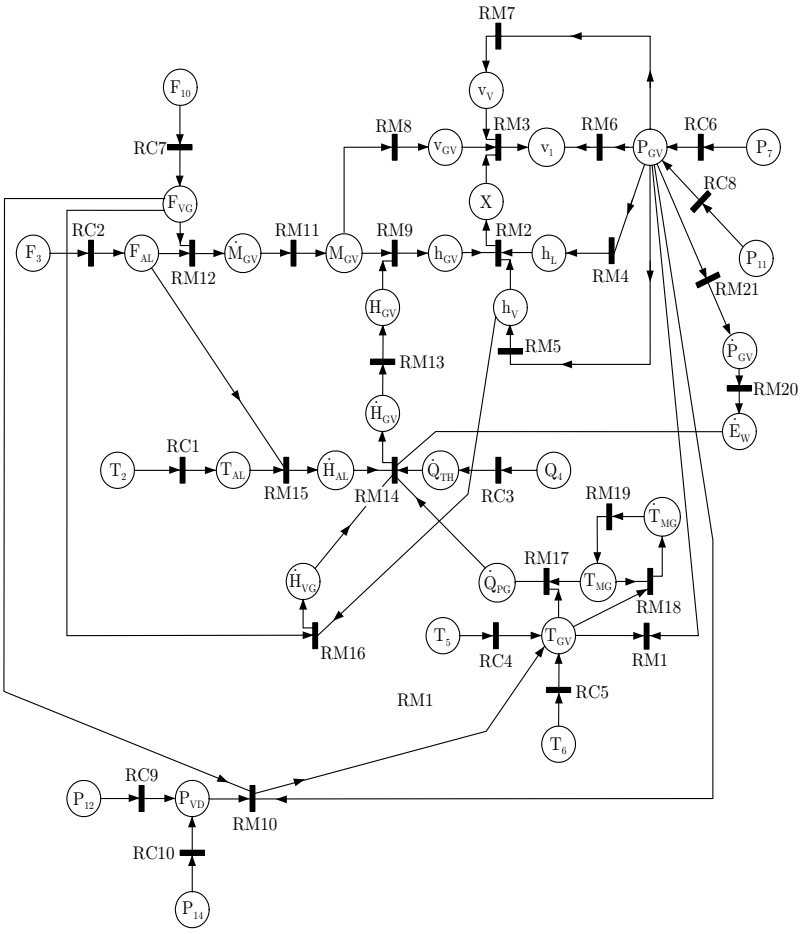


Fig. 10.53. Oriented structure graph of the boiler

The incidence matrix of the system structural graph is given by Fig. 10.53.

The analytic redundancy relations express the compatibility conditions of this system of 31 equations and 25 unknowns. They are exhibited by performing a complete matching with respect to the unknown variables, as shown on the incidence matrix, and illustrated by Fig. 10.53. The non-matched constraints are

$$NMC = \{RM1, RM9, RM10, RC4, RC8, RC10\}\,.$$

Computing the unknown variables as functions of the known ones by means of the matching, and then putting the result into the constraints of NMC, one obtains six relations which link only known variables, i.e. six analytic redundancy relations.

ARR1. The first ARR ($ARR1$) results from $RM1$, $RC5$ and $RC6$, and the associated alternated chain is shown on Fig. 10.54.

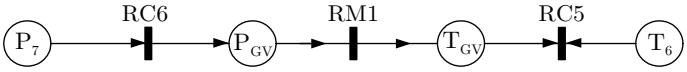


Fig. 10.54. ARR1 subgraph

This leads to the following computations:

$$\begin{array}{lll} \boldsymbol{RM1} & \Rightarrow T_{GV} & = -0,4594P_{GV}^2 + 12,7243P_{GV} + 99,003 \\ \boldsymbol{RC5}: & T_6 & = T_{GV} \\ \boldsymbol{RC6}: & P_7 & = P_{GV}. \end{array}$$

Replacing P_{GV} and T_{GV} by their measures P_7 and T_6 into expression $RM1$, one obtains $ARR1$

$$T_6 = -0,4594P_7^2 + 12,7243P_7 + 99,003. \tag{10.29}$$

ARR2. The computation form of $ARR2$ is:

$$\begin{array}{lll} \boldsymbol{RM10} & \Rightarrow F_{VG} & = \sqrt{K_D(z_{V2})\dfrac{(P_{GV} - P_{VD}) \cdot P_{VD}}{T_{GV}}} \\ \boldsymbol{RM1}: & T_{GV} & = -0,4594P_{GV}^2 + 12,7243P_{GV} + 99,003 \\ \boldsymbol{RC6}: & P_7 & = P_{GV} \\ \boldsymbol{RC7}: & \frac{F_{10}}{3600} & = F_{VG} \\ \boldsymbol{RC9}: & P_{12} & = P_{VD} \\ \boldsymbol{RM10} & \Rightarrow & \dfrac{(3600)^2(P_7 - P_{12}) \cdot K_D \cdot P_{12}}{-0,4594P_7^2 + 12,7243P_7 + 99,003} - F_{10}^2 = 0 \\ & r_2 & = F_{10}^2 - \dfrac{(3600)^2(P_7 - P_{12}) \cdot K_D \cdot P_{12}}{-0,4594P_7^2 + 12,7243P_7 + 99,003}. \end{array}$$

r_2 is in $\mathrm{kg}^2/\mathrm{s}^2$, it represents the link between the steam flow F_{VG}, as computed by P_{GV}, P_{VD} and T_{GV}, and F_{10}, the measured value of this flow. It is represented on Fig. 10.55.

ARR3. The successive steps for the computation of $ARR3$ are as follows.

Step 1. From constraints $RC1 - RC10$ and $RM13 - RM17$, an elimination procedure provides the following expression of H_{GV}:

$$H_{GV}^{(1)} = \int (Q_4 + 4200T_2F_3 - K_{GM}(T_5 - T_{MG}) - F_{10}h_V(P_7)dt) + H_{GV}(0).$$

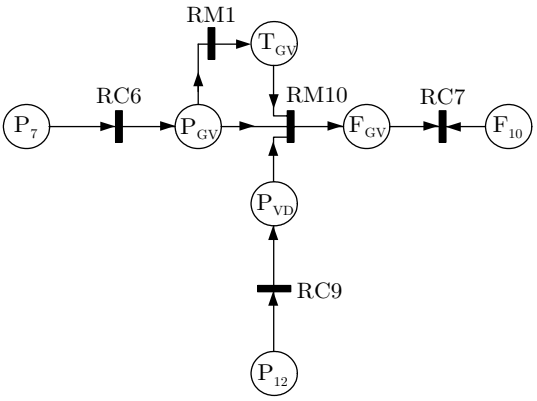


Fig. 10.55. ARR2 subgraph

Step 2. From $RM11 - RM13$, $RC2$ and $RC7$ the following relation is obtained:

$$M_{GV} = \int (F_3 - F_{10})/3600dt + M_{GV}(0).$$

Step 3. The expression of X (taken from $RM3$) is obtained using $RM8$ and the above expression of M_{GV}

$$X = \frac{0,175/\left(\int (F_3 - F_{10})/3600dt + M_{GV}(0)\right) - \nu_L(P_7)}{\nu_V(P_7) - \nu_L(P_7)}.$$

Step 4. Another expression of H_{GV} is obtained by substitution, replacing X and M_{GV} by their respective expressions in constraint $RM9$:

$$H_{GV}^{(2)} =$$

$$\left[\left(\frac{0,175/\left(\int (F_3 - F_{10})/3600dt + M_{GV}(0)\right) - \nu_L(P_7)}{\nu_V(P_7) - \nu_L(P_7)} \cdot (h_V(P_7) - h_L(P_7))\right)\right.$$

$$\left. + h_L(P_7)\right] \cdot \left(\int (F_3 - F_{10})/3600dt + M_{GV}(0)\right).$$

The residual is then

$$r_3 = H_{GV}^{(2)} - H_{GV}^{(1)}$$

$$\boldsymbol{r}_3 = \left[\left(\frac{0,175/\left(\int (F_3 - F_{10})/3600dt + M_{GV}(0)\right) - \nu_L(P_7)}{\nu_V(P_7) - \nu_L(P_7)}\right.\right.$$

$$\left.\left. \cdot\ (h_V(P_7) - h_L(P_7))\right) + h_L(P_7)\right] \cdot \left(\int (F_3 - F_{10})/3600dt + M_{GV}(0)\right)$$

$$-\left(\int (Q_4 + 4200T_2F_3 - K_{GM}(T_5 - T_{MG}) - F_{10}h_V(P_7))dt + H_{GV}(0)\right).$$

r_3 is an energy (in J) whose structure is shown on Fig. 10.56.

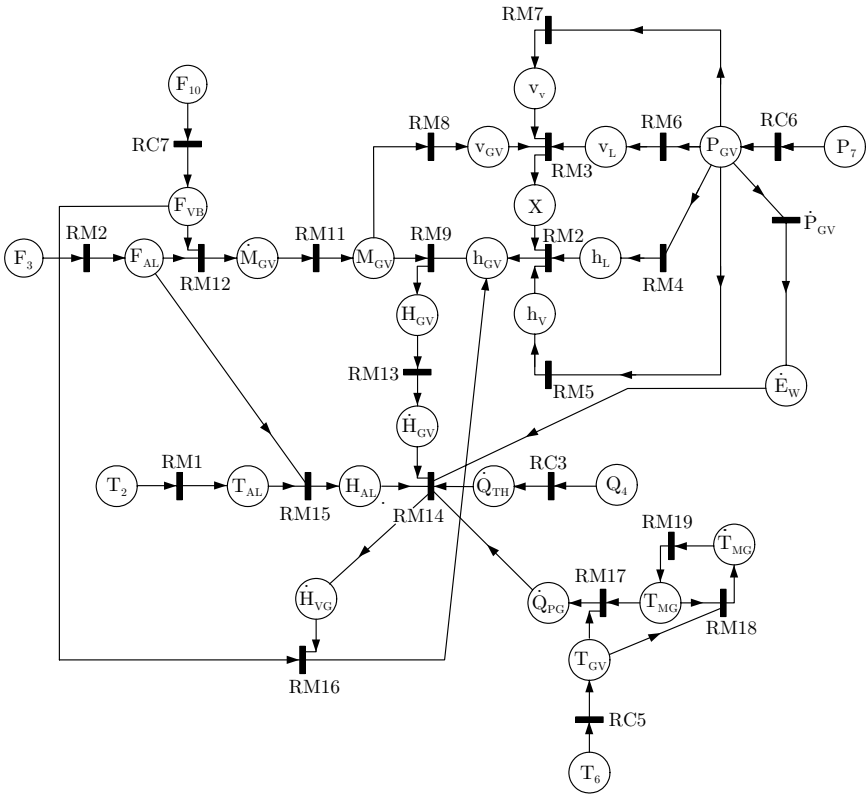


Fig. 10.56. ARR3 subgraph

Some remarks on $ARR3$ follow.

Remark 10.1 *Homogeneity*
For clarity, there is no coefficient in this expression to insure its homogeneity. In fact, the flows should be in kg/s while the sensors F_3 and F_{10} provide measurements in kg/h. The energy should be in J while sensor Q_4 provides measurements in kJ. Besides, $h_V(P_7)$ and $h_L(P_7)$ must be in J/kg. □

Remark 10.2 *Complexity*
The computation form of this residual makes use of many constraints. This follows from the fact that many variables cannot be measured (enthalpy, volumic mass, energy, quality of the steam). □

ARRs 4, 5 and 6. The three last ARRs are associated with hardware redundancy, since several sensors are measuring the same system variables.

Sensors T_5 and T_6 are both located in the boiler, but T_6 is slightly higher than T_5 and thus it measures the temperature of the steam. Under the hypothesis that the mixture is really homogeneous, both sensors should provide the same value

$$r_4 = T_5 - T_6.$$

Sensor P_7 is located in the boiler while sensor P_{11} is located at the output. Neglecting the pressure drop in the (small) pipe, both should provide the same value

$$r_5 = P_7 - P_{11}.$$

Sensors P_{12} and P_{14} are both located before the two parallel modulating valves at the beginning of the depressurisation circuit

$$r_6 = P_{12} - P_{14}.$$

10.4.5 Fault signatures

Remember that the signature of a fault is the subset of the redundancy relations which are influenced by the fault. The faults to be detected and isolated are faults in the sensors $\{T_5, T_6, P_7, P_{11}, P_{12}, P_{14}, F_{10}\}$ and in $\{M_{GV}, K_D\}$. For the system of 6 redundancy relations, the resulting signature table is given in Table 10.12.

Table 10.12 Fault signatures

$\nearrow$	T_2	F_3	Q_4	T_5	T_6	P_7	F_{10}	P_{11}	P_{12}	P_{14}	K_D	M_{GV}
r_1					1	1						
r_2						1	1		1		1	
r_3	1	1	1		1	1	1					1
r_4				1	1							
r_5						1		1				
r_6									1	1		

All twelve column vectors are different from zero, and thus all faults are detectable. Note that faults on sensors T_2, F_3 and Q_4 which were not in the specifications, can also be detected (it will be seen below that this is not exactly true, the actual sensitivity being too low). The physical interpretation of the theoretical signatures is as follows:

- Residual r_1 links the temperature T_{GV} and the pressure P_{GV} of the steam-water mixture. A clogging or a leakage should not act on this residual.
- Residual r_2 links the steam massic flow F_{VG}, the output pressure P_{VD}, the pressure in the boiler P_{GV}, and the temperature T_{GV}. A clogging should show

through this residual since coefficient K_D is explicitely present in the computation form.

- Residual r_3 links the different energies exchanged. It should be sensitive to a water leak because M_{GV} appears explicitely in its expression.
- Residuals r_4, r_5, and r_6 express hardware redundancy and thus are sensitive to the faults in the associated sensors. They should be independent of the clogging or the leakage faults.

Note that it is always possible to obtain new $ARRs$ by combining the existing ones. This can be done by substituting one variable in a relation by its expression obtained from another relation. For example, a new residual r_7 can be created from r_1 and r_5 by eliminating P_7 and, therefore, r_7 would only depend on T_6 and P_{11}.

The decision procedure now works as follows:

1. First compute the residuals and check whether they are or not different from zero (residuals being stochastic variables, this calls for hypotheses testing of change point detection algorithms).
2. Construct the system signature by putting a zero for every zero residual and a 1 for every non-zero residual, and compare the system signature with the theoretical ones given by Table 10.12.

Of course, since decision errors are always possible, the system signature might be different from any theoretical signature in the table. In that case, the nearest one is in general taken. Thus, the detection and isolation procedure will be less sensitive to errors as the distances between the theoretical signature is larger. Table 10.13 gives the Hamming distances between the 12 theoretical signatures.

It can be noticed that the distance between the signatures associated with T_2, F_3, Q_4 and M_{GV} is zero. This means that any of these faults has the same signature, and thus, it is impossible to isolate which one has occured. Namely, a fault of one of the three sensors canot be distinguished from a leakage in the boiler.

Table 10.13 Hamming distances

$\nearrow$	T_2	F_3	Q_4	T_5	T_6	P_7	F_{10}	P_{11}	P_{12}	P_{14}	K_D	M_{GV}
T_2	0	0	0	2	2	3	1	1	3	2	2	0
F_3		0	0	2	2	3	1	1	3	2	2	0
Q_4			0	2	2	3	1	1	3	2	2	0
T_5				0	2	5	3	2	3	2	2	2
T_6					0	3	3	4	5	4	4	2
P_7						0	2	3	4	5	3	3
F_{10}							0	3	2	3	1	1
P_{11}								0	3	2	2	1
P_{12}									0	1	1	3
P_{14}										0	2	2
K_D											0	2
M_{GV}												0

10.4.6 Experimental results

Normal operation. In normal operation, the residuals should be zero in the ideal case (no modelling error, no uncertainties, no measurement noise). This is not the case, since modelling errors and noises are really present. However, taking into account the relative values of the residuals with respect to the corresponding signals shows that the precision is not bad, as illustrated now for residual r_3, whose behaviour in normal operation is shown by Fig. 10.57.

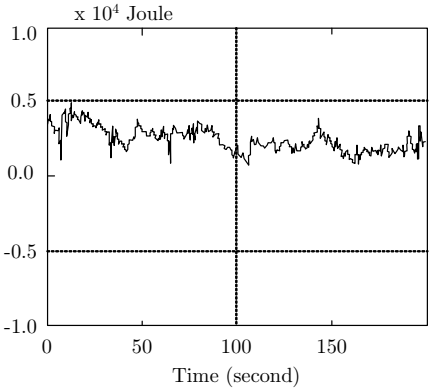


Fig. 10.57. Residual r_3 in normal operation

This residual expresses the difference between the two available ways of computing H_{GV}. Table 10.14 gives the normalised values of r_3 (%). It is seen that this residual, although nonzero in normal operation, can still be used, since its average value $\left(2,58 \cdot 10^3 \text{ J}\right)$ is very small with respect to the average value of H_{GV}, $\left(7,19^5 \text{ J}\right)$, the ratio being $0,36\,\%$.

Table 10.14 Residual r_3 analysis

	Residual r_3 value	Percentage with respect to H_{GV}
Average value	$2,5842 \cdot 10^3$	$0,36$
Standard deviation	$833,0997$	$0,12$
Maximum value	$5,033 \cdot 10^3$	$0,70$
Minimum value	$874,9006$	$0,12$

10.4.7 Fault scenarios

Sensor faults. Sensor faults are created by adding a $15\,\%$ extra signal to their output on the time window [75 s, 150 s]. T_2, F_3 and Q_4 act only on residual r_3 but this one

does not react. The reason is that although it is structurally sensitive to the faults, its actual (numerical) sensitivity is much too low (the energy associated with the faults is neglectible with respect to the energy of the generator). T_5 and T_6 are present in residuals r_1, r_3 and r_4. r_3 does not react to these faults, for the same reason as already explained. But r_1 and r_4 are really sensitive as shown by Fig. 10.58.

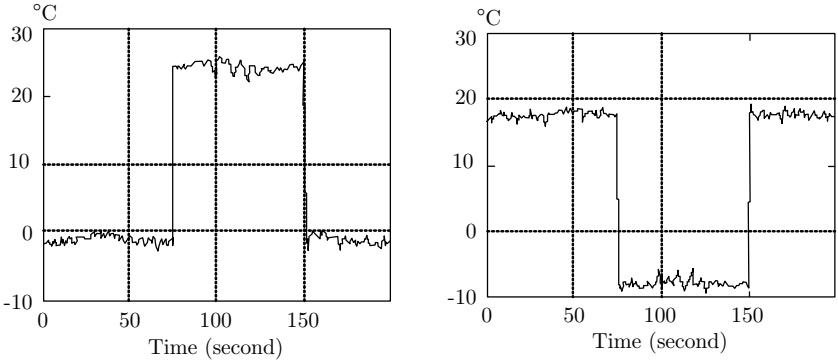


Fig. 10.58. Residual r_1 with $+15\,\%$ on sensor T_6 (left) and residual r_4 with $+415\,\%$ on sensor T_5 (right)

Faults on sensors P_7 and P_{11} affect residuals r_1, r_2 and r_3. Experimental results are given on Fig. 10.59 and 10.60.

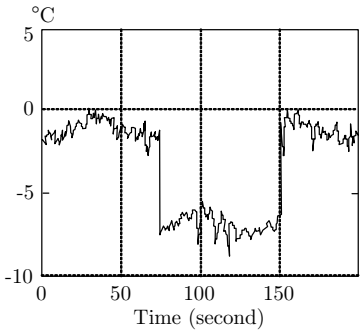


Fig. 10.59. Residual r_1 with $+15\,\%$ on P_7

It is seen that the sensitivity of r_3 to P_7 is real, and thus it reacts to the fault. Similarly, residuals r_2 et r_3 are really sensitive to faults of sensor F_{10} and residuals r_2 , r_6 are really senstive to faults of P_{12} and P_{14}.

Thus, all sensor faults can be detected since at least one residual is really senstivie to each of them.

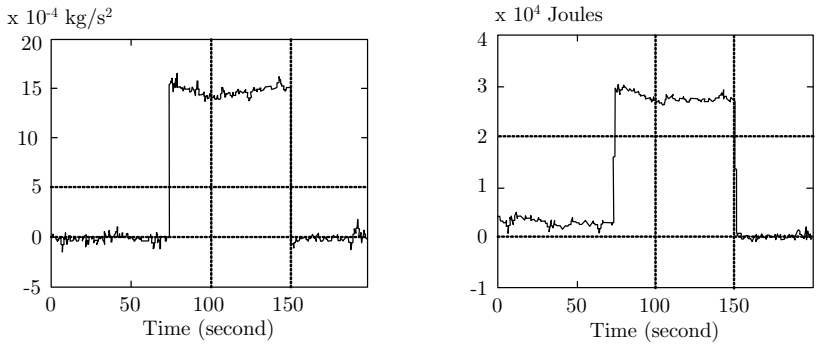


Fig. 10.60. Residual r_2 (left) and r_3 (right) with 15 % on P_7

Process faults. Two experiments are made with the faulty process: the clogging of the output valve (between time 60 s and 100 s), and a water leakage in the boiler (only for 3 seconds, from time 125 s to 128 s, since the experiment is very dangerous).

As expected, residual r_1 is not sensitive to any of these faults. Residual r_2 is sensitive only to the clogging (Fig. 10.61) because it is associated with the steam flow F_{VG} (measured by F_{10}) which is not affected by the fault in the considered operating conditions (the boiler still contains water enough).

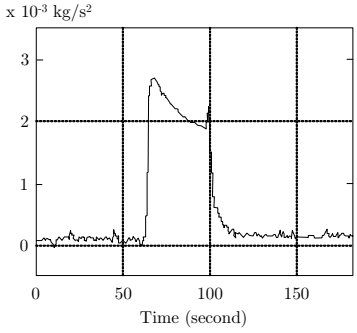


Fig. 10.61. Residual r_2 clogging of the output pipe

Residual r_3 (Fig. 10.62) is only senstive to the leak. The physical interpretation is that since the level regulator tries to compensate the leak, the boiler fills up with more cold water, hence creating a change in its energetic content.

Residuals r_4, r_5 and r_6 confirm their insensitivity to the clogging and to the leak.

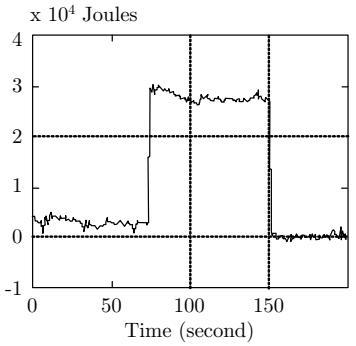


Fig. 10.62. Residual r_3 leak in the boiler

10.4.8 Evaluation of the experimental results

In spite of the simplifications, the model of the steam generator is quite good, as shown by the fact that in normal operation, the residuals are practically zero.

Structural analysis has been applied to design the fault diagnostic system, and has provided residuals which detect and isolate the faults given in the specifications. The consistency between the expected behaviour of the residuals and their actual behaviour has been experimentally shown.

The decision procedure which has been applied is rather simple: The residuals have been compared with a threshold equal to 3 times the standard deviation under the normal operation hypothesis. It has not been necessary to develop more sophisticated approaches, since the results obtained in the nominal operating regime (6-8 bars) are quite satisfactory: For all the experiments which have been performed, the false alarm and the missed-detection rates were zero.

10.5 Fault-tolerant electrical steering of warehouse trucks

This case study deals with electrical steering, a combined hardware-software-control problem that shows how the methods from the theory chapters are applied to this type of embedded system. Being critical to the safety of vehicles, the steering system of a vehicle is required to maintain the vehicles ability to steer until it is brought to halt, should a fault occur. With electrical steering becoming a cost-effective candidate for electrical powered vehicles, a fault-tolerant architecture is needed that meets this requirement. This case study treats the fault-tolerance properties of an electrical steering system. It presents a novel fault-tolerant architecture where a dedicated AC-motor design was used in conjunction with cheap voltage measurements to ensure detection of all relevant faults in the steering system. The study shows how active control reconfiguration can accommodate all critical faults. The fault-tolerant steering system was implemented on the hardware of a warehouse

truck and validations were made with hardware-in-the-loop tests, demonstrating diagnosis of all critical faults and ability to obtain the required fault-tolerant capabilities.

10.5.1 Introduction

A steering system for a vehicle on public road is required to maintain its ability to steer until it can be brought to halt, irrespective of any single fault in the steering system. With electrical steering becoming a cost-effective candidate for steering of electrical powered vehicles, system architecture and underlying interfaces, signal processing and control methods need be dedicated to meet this fundamental requirement.

Fail-operational systems that are able to continue operation with unchanged performance irrespective of any defect within the system itself are common in critical applications such as airplane control. Implemented with triple redundancy or more, these systems are, however, prohibitive for commercial markets. In order to achieve low-cost solutions, ideas from fault-tolerant control could be useful since we could accept degraded performance after a fault has occurred, if the vehicle is still able to be steered until it can be brought to a halt.

This case study suggests a fault-tolerant solution for an electrical steering system, discusses how hardware, software and system functionality should be addressed and presents a fault-tolerant architecture that enables a cheap solution that meets the requirement of authorities for driving on public roads. Analysing the architecture of a steering-by-wire system it is considered how duplicated actuator motors could be avoided. Using the AC motor star-point measurement, the study shows how diagnosis is obtained for all critical single-fault cases in the system. Finally, an extract of systematic tests of hardware and software faults are included as validation of the fault-tolerance abilities of electrical steering system. Hardware in the loop tests were made as validation using the hardware platform from a warehouse truck to document real performance.

10.5.2 Electrical Steering

The architecture of a basic electrical steering system for vehicles is shown in Fig. 10.63. The double-arrows indicate that connected sub-systems affect each other. The steering system contains a steering input system, a computer, a drive system, an induction motor, mechanical link to steering wheel(s), and a battery power supply. The user command comes from a steering-wheel or a joy-stick. The steering input system is a hardware component, which transforms the mechanical input to a reference steering signal for the computer. A control algorithm in the computer generates the actual control command to the drive system, and the drive system converts the voltage of the power supply into a three phase voltage signal for the induction motor. The motor is mechanically linked to the wheel(s) of the vehicle.

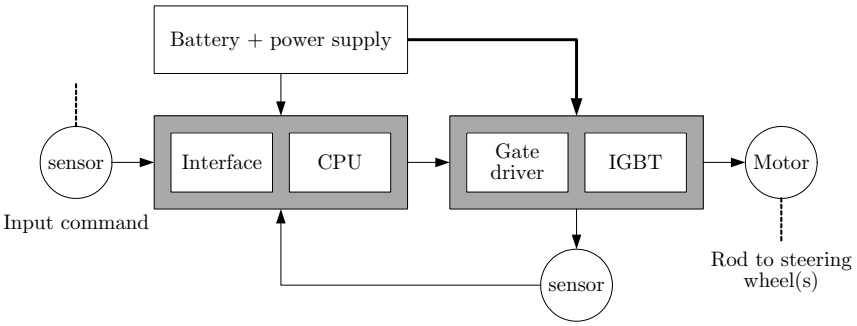


Fig. 10.63. A basic electrical steering system

For warehouse trucks and similar commercial applications, a steering system is needed that is low-cost and fault-tolerant. A basic electric steering system is shown in Fig. 10.63. It is potentially low-cost but is not fault-tolerant. To develop a low-cost and fault-tolerant steering system, all subsystems and their interactions need be carefully considered. With this purpose, it is useful to separate into three subsystems. One is the power supply, the second is the steering input system, the third is the wheel drive actuator, comprising computer, drive electronics, induction motor, wheel, and sensors. This case study considers the latter.

Actuator and sensor system. Fig. 10.64 shows wheel drive actuator system. The drive consists of an inverter and a gate-driver. PWM-signals are calculated by the computer and forwarded to the gate-driver that opens and closes the semiconductor switches (IGBT). The power electronics converts the battery supply voltage to a switched three phase voltage which is applied to the induction motor. Potentially, such system could contain a number of different sensors. However, for the warehouse truck and other low-cost applications, it is possible to use a control principle which uses only the average current i_{DC} for feedback. Figure 10.64 shows three sensors: the wheel demand V_w, current consumed by the power stage i_{DC} and the stator'a star point voltage of the motor, V_{st}.

Critical faults. It is possible to identify a large number of possible faults in the drive system and the induction motor, however, not all faults are relevant. For the warehouse truck, safety is the key issue, and faults that need to be considered are those that have implications on safety.

The steering system for a warehouse truck is as a relative low-torque/low-power application and in such applications, regulations by public authorities list mechanical components which are unlikely to fail if their design follow given standards. For example, breakage of the rotor shaft and breakage of the iron bars in the rotor belong to this category. In contrast, motor bearings become worn and will fail eventually.

All electronic components need to be considered prone to failure. This includes the DC link sensor (the i_{DC} signal), the computer with analog interface and the

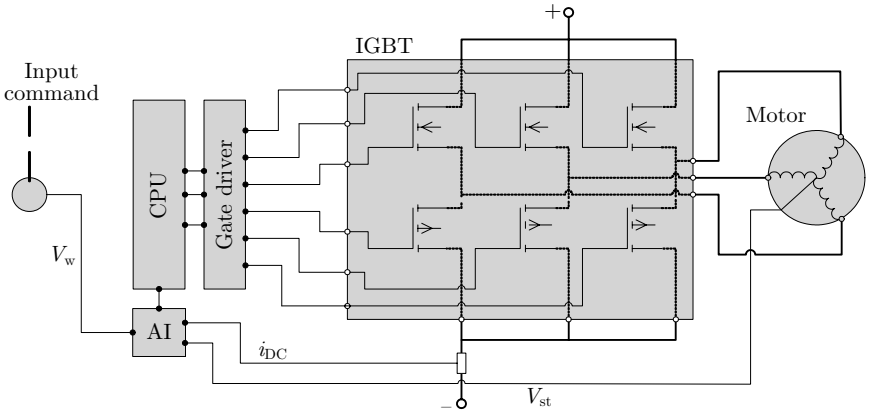


Fig. 10.64. Basic wheel drive system

inverter electronics. The inverter contains gate-driver and the power switches. Each power switch can either be short circuited or be open circuited (disconnected).

Electrical components include harness and induction motor. In the motor, stator windings are subject to be disconnected or short circuited, an internal turn winding short circuit could occur. In the harness or in the motor, a phase could be disconnected (open phase fault). A phase could be short-circuited to chassis but such failure is dealt with by intrinsic design using well established techniques with double-insulation.

Component failures that would cause loss of steering capability with the basic architecture of the electrical steering system are listed in Table 10.1.

Table 10.1. Critical component failures

Subsystem	Component	Failure	No.
Drive	Gate driver	Malfunction	f_1
	Power switch	Short circuit	f_2
	Power switch	Fail open	f_3
Motor	Winding	Open phase	f_4
	Winding	Turn short	f_5
	Harness	Open phase	f_6
	Harness	Phase to chassis	f_7
	Bearing	Stuck	f_8
Sensor	DC link	Malfunction	f_9
Power	Harness	Power loss	f_0

10.5.3 System architecture

The requirement of maintaining ability to steer the vehicle until it can be brought to a halt has some straightforward implications on the architecture of the electrical steering system,

- No single failure of a component may prevent adequate steering ability
- The system shall bypass a faulty component to continue operation or override the effects of a faulty component
- If an actuator fault is present, parallel action should have enough control authority to override the effects of the fault

While good case by case engineering designs could be achieved from such immediate observations, a systematic and rigorous analysis offer important benefits. One is to assure that any component discrepancy from normal is covered by an analysis of fault propagation and the consequences of component failure. Another is a provable correct deduction of fault propagation from basic assumptions is a valuable tool in the quality assurance and systematic validation of a design. The algebraic approach to fault propagation analysis, with the extension to generic component representations provide a useful method for a systematic design.

Component-based analysis. The analysis presented here is based on services offered by components in normal or faulty modes, and the impact the architecture has on the service available from the entire electrical steering system. A complete fault-propagation analysis was carried out in [254].

Subsystem behaviours. The system breakdown in Fig. 10.63 showed the system components and their interaction. With notation for signals shown in Table 10.3, behaviours in normal mode between input and output signals are listed in Table 10.2. The service $S^{(k)}$ offered by a component k is to deliver an output, according to the specified behaviour $S^{(k)}(c_k^{(v)})$ where $v \in \{n, d1, d2, .., o\}$ is a version of the service that follows from the condition of the component (normal, reduced1,..,none). If a component has an internal failure, a version of the service may be available with degraded performance $S^{(k)}(c_k^{(d)})$ or the service may not be available at all $S^{(k)}(c_k^{(d)})$.

System service. The steering service obtained for the system as an entirety is a function of the component architecture A and the version vector $\boldsymbol{v}$ for the present condition of components. With m components, the set of available behaviours will be $C_v = (c_1^{v(1)}, c_2^{v(2)}, ..., c_m^{v(m)})$, and the overall system service is $S^{(s)}(A, c_i^{v(i)}) = A(S^{(i)}), i = 1, ..., m$. With a single string architecture as Fig. 10.64, we obtain

Table 10.2. Component services and behaviours

Component	In	Out	Behaviour
Reference	u_{ref}	T_{dem}	$T_{\mathrm{dem}} = c_r(u_{\mathrm{ref}})$
Computer	T_{dem}	d_{com}	$d_{\mathrm{com}} = c_c(T_{\mathrm{dem}})$
AC drive	$\boldsymbol{u}_{\mathrm{com}}$	$\boldsymbol{u}_s$	$u_s = c_d(d_{\mathrm{com}})$
Motor	$\boldsymbol{u}_s$	T_m,$\mathbf{i}_s$	$T_m = c_{mT}(\boldsymbol{u}_s, \mathbf{i}_s, \omega_s)$
			$\mathbf{i}_s = c_{mi}(\boldsymbol{u}_s, T_m, \omega_s)$
i-sensor	i_m	i_{dc}	$i_m = i_{dc}$
Power	V_{bat}	V_{bus}	$V_{\mathrm{bus}} = const.$

Table 10.3. Notation

Variable	Explanation
u_{ref}	Driver's input command
Q_{dem}	Torque demand
$\boldsymbol{u}_{\mathrm{com}}$	Command to inverter
$\boldsymbol{u}_s$; $\boldsymbol{u}_r$	Stator (rotor) voltage
$\mathbf{i}_s$; $\mathbf{i}_r$	Stator (rotor) current
ψ_s; ψ_r	Stator (rotor) magnetic flux linkage
ϕ, θ	Angles stator and rotor fields
T_m	Motor torque
ω_m	Motor-shaft angular velocity
ϕ_m	Motor-shaft angle
i_{dc}	Average motor current
i_m	Measured average motor current
u_n	Starpoint voltage
V_{bus}	Battery voltage
V_{bus}	Bus voltage

$$S^{(s)}_{\mathrm{single}} = S^w \wedge S^p \wedge S^m \wedge S^d \wedge S^i \wedge S^c \wedge S^y, \tag{10.30}$$

where superscript w indicates wheel, p is power supply, m is motor, d is drive, i is current sensor, c is computer and u is voltage sensor.

An alternative could be a hardware configuration with two parallel totally redundant lines with only the wheel in common,

$$\begin{aligned} S^{(s)}_{\mathrm{rhw}} \quad = \quad & S^w \wedge \big(\big(S^{p1} \wedge S^{m1} \wedge S^{d1} \wedge S^{i1} \wedge S^{c1} \wedge S^{y1} \big) \\ & \vee \big(S^{p2} \wedge S^{m2} \wedge S^{d2} \wedge S^{i2} \wedge S^{c2} \wedge S^{y2} \big) \big) \end{aligned} \tag{10.31}$$

This solution is expensive as it requires two motors. Two motors would be allowed to drive a common shaft if it is proved that a healthy motor will be able to have control authority over a faulty one. A cost effective solution would be one single motor that could use dual windings on the stator and divide the power drive output stages between the windings. With such solution, both winding sets and inverter

stages would be used in normal operation while, in case of failure in one stator, the motor would be driven with up to half of maximal (nominal) torque. The fault-tolerant architecture shown in Fig. 10.65 is based on this idea. It was a prerequisite for this solution that the rotor bars of the AC motor are not prone to failure. Present (2006) standards allow a common rotor provided design rules are adhered to, just as is the case of the common rod from the motor to the steering wheel, which is assumed unlikely to fail provided standard design rules are followed.

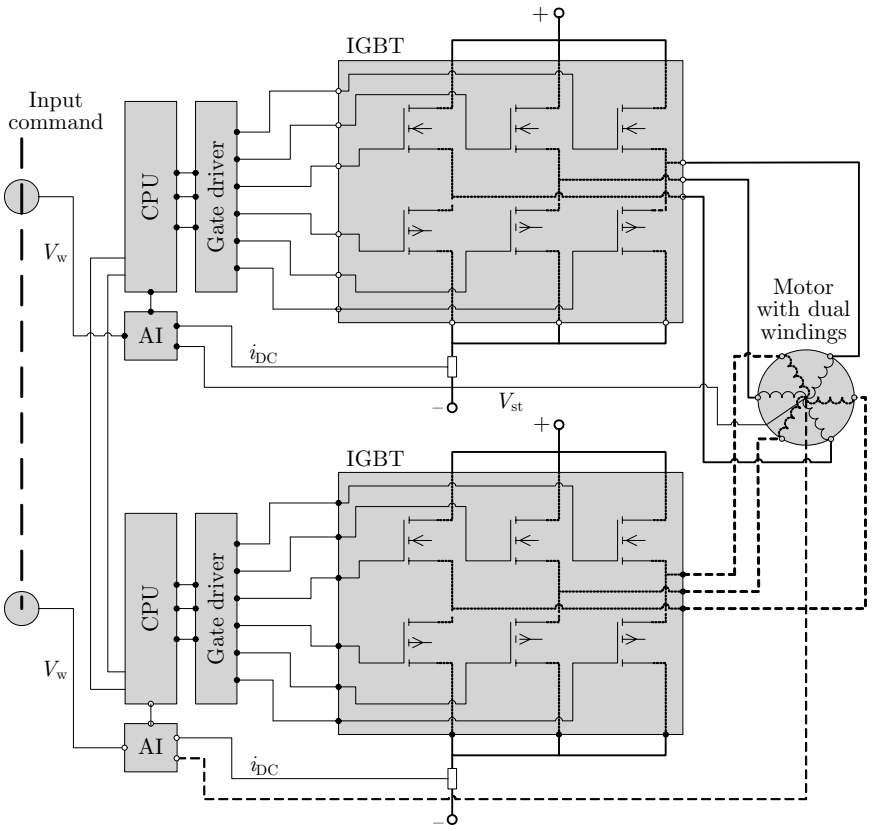


Fig. 10.65. An architecture for fault-tolerant electrical steering

The service at system level is

$$S_{\text{ftc}}^{(s)} = S^w \wedge S^{md} \wedge \left(S^{p1} \vee S^{p2}\right) \wedge \left(S^{i1} \vee S^{i2}\right) \wedge \left(\left(S^{c1} \wedge S^{d1} \wedge S^{y1}\right) \vee \left(S^{c2} \wedge S^{d2} \wedge S^{y2}\right)\right) \tag{10.32}$$

The paradigm in this architecture is that component failures should be detectable and faulty components be bypassed by controlling the signal flow in the software of the system. Using a motor with dual windings requires

- No critical failure in the motor must prevent the development of torque in the non-faulty part.
- A motor bearing fault should be detected before it can turn into a failure that makes the motor shaft unable to turn.

Dual stator AC motor. An AC motor with four windings was proposed earlier in the literature. A simpler and more cost effective solution is to remain with a three phase drive systems since mass produced power stages for inverters (IGBT devices) are available for the six transistor switch bridge sets used by 3-phase drives. Hence, it is worthwhile to investigate AC motor properties for motors with duplicated stator windings. A layout of a dual stator AC motor is shown in Fig. 10.66. A scrutiny of parameters in the dual winding motor showed that one physical motor with dual windings give fault-tolerance properties quite equivalent to two independent motors on a common shaft. The key issue is that the mutual inductance M_{ss} is not so large that a short circuit on one winding will prevent the motor from turning using the other winding for control.

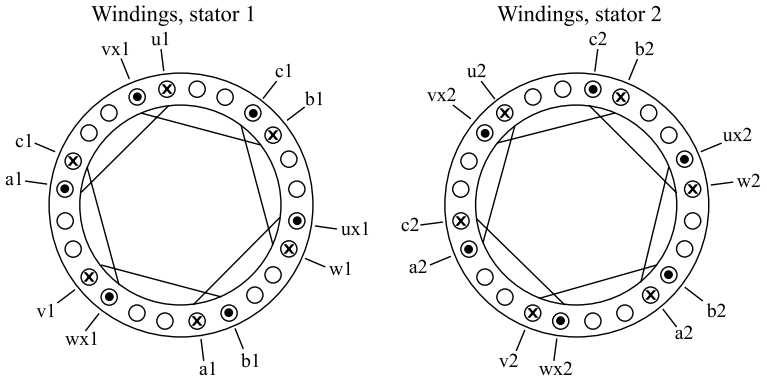


Fig. 10.66. AC motor with dual stator windings (by courtesy of J. S. Thomsen)

Requirements to fault detection and accommodation. The fault-tolerant architecture includes two actuators in parallel consisting of drive system and stator windings. In normal operation both actuators are active and co-operate to rotate the wheel according to control input. If a fault should occur in one of the two actuators it must be detected and accommodated to achieve fault tolerance. All critical faults which are not handled by design as described in Section 10.5.3 must be detected and accommodated. This includes faults f_1, f_2, f_3, f_4, f_5, and f_9, in Table 10.1.

It can be expected that faults in the drive system will propagate to have an effect in the stator windings. Several methods exists for detecting faults in stator windings.

Utilizing a star point sensor is advantageous for a low-cost solution as only two voltage measurements are needed.

Coverage. The paradigm in this architecture is that faults can be detected and fault handling be successfully achieved to bypass malfunctioning components. In theory, the probability of making a successful detection and system reconfiguration (coverage), is not 100%, but it will be shown in the experimental section that for all faults, which impair steering performance, faults can indeed be detected, isolated and the system be reconfigured.

In the following, we focus the discussion to power drive and actuator and limit the treatment to the critical faults, noting that diagnosis of motor bearing wear is not pursued in this context.

10.5.4 Structural analysis

The analysis of structure comprises the elements: formulating constraints, providing a matching on the unknown variables, determine measurements in which all critical faults are detectable and faulty parts are isolable such that correct reconfiguration can be made.

Constraints for the dual stator AC motor. The behaviours of the dual stator AC motor are available through a small extension to the theory for usual electrical motors, by taking account of the mutual inductions within the dual winding motor. Let the terminal voltage be $\boldsymbol{u}_t = (u_{t1}\ u_{t2}\ u_{t3})'$, the current in a stator winding $\mathbf{i}_s = (i_{s1}\ i_{s2}\ i_{s3})'$ and current in the rotor $\mathbf{i}_r = (i_{r1}\ i_{r2}\ i_{r3})'$. The flux in a stator is similarly the vector ψ_s and the flux through the rotor is ψ_r. Parameters are R for resistance, L for inductance, M for mutual inductance, θ for electrical angle between stator 1 and rotor, β the electrical offset angle between two stators and $N_{(3,3)}(\phi)$ a rotation matrix used to express the rotor-stator flux interaction,

$$\boldsymbol{N}(\phi) \quad = \begin{pmatrix} \cos(\phi) & \cos(\phi + \frac{2\pi}{3}) & \cos(\phi + \frac{4\pi}{3}) \\ \cos(\phi + \frac{4\pi}{3}) & \cos(\phi) & \cos(\phi + \frac{2\pi}{3}) \\ \cos(\phi + \frac{2\pi}{3}) & \cos(\phi + \frac{4\pi}{3}) & \cos(\phi) \end{pmatrix}$$

For later use, note that the sum of elements in a column of $\boldsymbol{N}(\phi)$ is zero,

$$\textstyle\sum_{k=1}^{3} \boldsymbol{N}_{kj}(\phi) \quad = \quad cos(\phi) + cos(\phi + \frac{2\pi}{3}) + cos(\phi + \frac{4\pi}{3}) = 0. \quad (10.33)$$

The basic electrical equations for the dual winding AC motor are

$$
\begin{array}{lll}
c_1: & \boldsymbol{u}_t^{(1)} & = u_n^{(1)} + R_s \mathbf{i}_s^{(1)} + \frac{d}{dt}\left(\psi_s^{(1)}\right); \\
c_2: & \boldsymbol{u}_t^{(2)} & = u_n^{(2)} + R_s \mathbf{i}_s^{(2)} + \frac{d}{dt}\left(\psi_s^{(2)}\right); \\
c_3: & 0 & = R_r \mathbf{i}_r + \frac{d}{dt}\left(\psi_r\right) \\
c_4: & \psi_s^{(1)} & = L_s \mathbf{i}_s^{(1)} + M_{sr} \boldsymbol{N}(\theta) \mathbf{i}_r + M_{ss} \boldsymbol{N}(-\beta) \mathbf{i}_s^{(2)} \\
c_5: & \psi_s^{(2)} & = L_s \mathbf{i}_s^{(2)} + M_{sr} \boldsymbol{N}(\theta+\beta) \mathbf{i}_r + M_{ss} \boldsymbol{N}(\beta) \mathbf{i}_s^{(1)} \\
c_6: & \psi_r & = L_r \mathbf{i}_r + M_{sr} \boldsymbol{N}'(\theta) \mathbf{i}_s^{(1)} + M_{sr} \boldsymbol{N}'(\theta-\beta) \mathbf{i}_s^{(2)} \\
c_7: & 0 & = \sum_{k=1}^{3} i_{sk}^{(1)} \\
c_8: & 0 & = \sum_{k=1}^{3} i_{sk}^{(2)} \\
c_9: & 0 & = \sum_{k=1}^{3} i_{rk}
\end{array}
\tag{10.34}
$$

where $\boldsymbol{u}_t^{(1)}$ and $\boldsymbol{u}_t^{(2)}$ are the terminal voltage vectors applied on the two stator windings.

The mechanical variables are the angle θ and the angular velocity ω, motor torque T_m and load torque T_l. Total inertia referred to the motor shaft is I_t. The torque balance of the motor then gives

$$
\begin{array}{lll}
c_{10}: & T_m & = \left(\mathbf{i}_s^{(1)}\right)' \frac{\partial}{\partial \theta}\left(M_{sr}(\theta)\right) \mathbf{i}_r \\
 & & \quad + \left(\mathbf{i}_s^{(2)}\right)' \frac{\partial}{\partial \theta}\left(M_{sr}(\theta-\beta)\right) \mathbf{i}_r \\
c_{11}: & \frac{d}{dt}\left(I_t \omega\right) & = T_m - T_l \\
d_1: & \omega & = \frac{d}{dt}\left(\theta\right)
\end{array}
\tag{10.35}
$$

Without loss of generality, several differential constraints have been written implicitly in Equations (10.34) and (10.35) to limit the size of the incidence matrix.

The sets of unknown variables X_m and known variables K_m in Equations (10.34) and (10.35) are

$$
\begin{array}{lll}
X_m & = & \{u_n^{(1)}, u_n^{(2)}, \mathbf{i}_s^{(1)}, \mathbf{i}_s^{(2)}, \mathbf{i}_r, \psi_s^{(1)}, \psi_s^{(2)}, \psi_r, \theta, \omega, T_m, T_l\} \\
K_m & = & \{\boldsymbol{u}_t^{(1)}, \boldsymbol{u}_t^{(2)}\}.
\end{array}
\tag{10.36}
$$

Viewing Eq. (10.36) as scalar variables, there are 24 unknown and 6 known scalar variables. There are 24 scalar constraints comprised in Equations (10.34) and (10.35). Hence, the system is under-constrained or just constrained on X_m.

The structure graph S_m is shown in the incidence matrix where each of the vector constraints have been split into their scalar parts a, b and c respectively, and the columns refer to the scalar variables in the set X_m.

	$u_n^{(1)}$	$u_n^{(2)}$	$i_{s1}^{(1)}$	$i_{s2}^{(1)}$	$i_{s3}^{(1)}$	$i_{s1}^{(2)}$	$i_{s2}^{(2)}$	$i_{s3}^{(2)}$	i_{r1}	i_{r2}	i_{r3}	$\psi_{s1}^{(1)}$	$\psi_{s2}^{(1)}$	$\psi_{s3}^{(1)}$	$\psi_{s1}^{(2)}$	$\psi_{s2}^{(2)}$	$\psi_{s3}^{(2)}$	ψ_{r1}	ψ_{r2}	ψ_{r3}	θ	ω	T_m	T_l	$u_{ta}^{(1)}$	$u_{tb}^{(1)}$	$u_{tc}^{(1)}$	$u_{ta}^{(2)}$	$u_{tb}^{(2)}$	$u_{tb}^{(3)}$
c_{1a}	1	0	①	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c_{1b}	1	0	0	①	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c_{1c}	①	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c_{2a}	0	1	0	0	0	①	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c_{2b}	0	1	0	0	0	0	①	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c_{2c}	0	①	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
c_{3a}	0	0	0	0	0	0	0	0	①	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
c_{3b}	0	0	0	0	0	0	0	0	0	①	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
c_{3c}	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	①	0	0	0	0	0	0	0	0	0	0
c_{4a}	0	0	0	1	0	0	1	1	1	1	1	①	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
c_{4b}	0	0	0	1	0	1	1	1	1	1	1	0	①	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
c_{4c}	0	0	0	1	0	1	1	1	1	1	1	0	0	①	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
c_{5a}	0	0	1	1	1	1	0	0	1	1	1	0	0	0	①	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
c_{5b}	0	0	1	1	1	0	1	0	1	1	1	0	0	0	0	①	0	0	0	0	1	0	0	0	0	0	0	0	0	0
c_{5c}	0	0	1	1	1	0	0	1	1	1	1	0	0	0	0	0	①	0	0	0	1	0	0	0	0	0	0	0	0	0
c_{6a}	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	①	0	0	1	0	0	0	0	0	0	0	0	0
c_{6b}	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	①	0	1	0	0	0	0	0	0	0	0	0
c_{6c}	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	①	1	0	0	0	0	0	0	0	0	0
c_7	0	0	1	1	①	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	①	0	0	0	0	0	0	0	0	0
c_8	0	0	0	0	0	1	1	①	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c_9	0	0	0	0	0	0	0	0	1	1	①	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c_{10}	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	①	0	0	0	0	0	0
c_{11}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	①	0	0	0	0	0
d_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	①	0	0	0	0	0	0	0	0

A complete matching on X_m is marked in the incidence matrix by ①. The matching is also complete on C_m so driven by the terminal voltages on the two windings, and with unknown load torque, this system is just constrained. The matching could not be found using the simple ranking algorithm due to the closed loops in this graph. The more general algorithms had to be employed.

The existence of a complete matching knowing the terminal voltages on the motor, but not the mechanical load torque, shows that the motor will enter an equilibrium state where motor torque outbalance the load torque. Currents and fluxes within the motor are determined by the solution to the set of nonlinear equations in Equations (10.34) and (10.35). The existence of a symbolic or numeric solution can not be determined from the structure graph alone since the loops in the structure graph comprise nonlinear elements. A scrutiny in electrical machines shows, however, that a solution does exist, which is indeed expected from similarity of this motor with single stator AC motors.

Assured that the solution does exist, it timely to consider which sensors should be made available to meet the fault-tolerance requirements.

Measurements on the motor. Several signals could be monitored in the AC motor and the connected components. From a cost perspective, high-frequency (i.e. 20-200 kHz) pulse width modulated (PWM) voltage signals are difficult and expensive to monitor. In contrast, voltages without high-frequency contents are easily and inexpensively converted to digital signals. Measurement of high currents is certainly possible but require expensive transducers when the currents are above the 5-10 A range. Since $\boldsymbol{u}_t$ is high frequency chopped (PWM) and the 6 components of the vector valued currents $\mathbf{i}_s^{(1)}$, $\mathbf{i}_s^{(2)}$ belong to the high current category, the preferred monitoring possibility for the AC motor are the two star point voltages $v_n^{(1)}$ and $v_n^{(2)}$. Choosing the star-point voltages as measurements means to add two constraints,

$$\begin{array}{lll} m_1: & u_{nm}^{(1)} & = u_n^{(1)} \\ m_2: & u_{nm}^{(2)} & = u_n^{(2)} \end{array} \tag{10.37}$$

Star point voltage. With $u_n^{(1)}$ and $u_n^{(1)}$ related to voltage and current in the two stators, constraints c_1 to c_2 in Eq. (10.34), consider summation of the a, b and c components for each stator. For brevity, we use the notation

$$\sum_{k=1}^{3} u_{tk}^{(1)} := \sum_{3} u_t^{(1)}.$$

Then

$$\sum_{3} u_t^{(1)} \quad = \quad 3u_n^{(1)} + L_s \sum_{3} i_s^{(1)} + \frac{d}{dt} \sum_{3} \psi_s^{(1)} \tag{10.38}$$

and

$$\begin{aligned}\sum_3 \psi_{tk}^{(1)} &= L_s \sum_3 i_s^{(1)} + M_{sr} \sum_3 (\boldsymbol{N}(\theta)\mathbf{i}_r) \\ &+ M_{ss} \sum_3 (\boldsymbol{N}(\theta-\beta)\mathbf{i}_s^{(2)})\end{aligned}$$

Since

$$\begin{aligned}\sum_3 \boldsymbol{N}(\theta)\mathbf{i}_r &= \sum_{k=1}^{3} \boldsymbol{N}_{k1}(\theta) i_{r1} + \sum_{k=1}^{3} \boldsymbol{N}_{k2}(\theta) i_{r2} + \sum_{k=1}^{3} \boldsymbol{N}_{k3}(\theta) i_{r3} \\ &= \mathbf{0} \cdot \mathbf{i}_r\end{aligned}$$

and

$$\sum_3 i_s^{(1)} = 0,$$

Eq. (10.38) is reduced to

$$0 = 3u_n^{(1)} - \sum_3 u_t^{(1)}, \tag{10.39}$$

and a similar result is obtained for stator 2.

This shows that with a symmetric voltage vector applied at the terminals , i.e. $\sum_3 u_t = 0$, the star point voltage is zero when the system acts according to its normal behaviour.

Structural detectability in star-point residuals. Adding the two measured star-point voltages as known variables and the associated constraints from Eq. (10.37) to the structure graph, these two constraints remain unmatched and are hence parity relations, that are used for residual generation. Backtracking through the matching disclose how violation of constraints are detectable in the two residuals,

$$\begin{aligned}r_1(t) &= 3u_n^{(1)}(t) - \sum_3 u_t^{(1)}(t) \\ r_2(t) &= 3u_n^{(2)}(t) - \sum_3 u_t^{(2)}(t),\end{aligned} \tag{10.40}$$

The result is that from the star-point measurements, violation of any constraint in Equations (10.34) and (10.35) are detectable in both of the residuals of Eq. (10.40).

Structural isolability is not achieved for any violation of the primary constraints because both residuals depend structurally on the entire set of the basic constraints. This does not necessary mean that (not structural) isolation is impossible. In order to scrutinize the properties of the residuals in Eq. (10.40).

Extension to other components. So far, only the dual stator AC motor was treated. It remains to formulate the constraints for the remaining components of the electrical steering system.

Each of the power stages (power drives) receive a voltage command $\boldsymbol{u}_{cmd}$ from the microprocessor and deliver the PWM signal $\boldsymbol{u}_t$ to the motor. The power consumption of the drive is P_d and power delivered to a stator is P_s with an efficiency η_d. Then, $(j = 1, 2)$

$$\begin{aligned} pd_{1j}: \quad & \boldsymbol{u}_t^{(j)} && = \boldsymbol{u}_{cmd}^{(j)} \\ pd_{2j}: \quad & P_d^{(j)} && = (\mathbf{i}_{dc}^{(j)}) V_{bus} \\ pd_{3j}: \quad & \eta_d P_d^j && = P_s^j \end{aligned} \tag{10.41}$$

where i_{dc} is the current drawn by the power stage from the DC voltage supply.

For each of the micro processors $(j = 1, 2)$,

$$\begin{aligned} mp_{1j}: \quad & \boldsymbol{u}_{cmd}^j && = c_c(u_{rm}^{(j)}) \\ mp_{2j}: \quad & u_{nm}^{(j1)} && = u_n^{(1)} \\ mp_{3j}: \quad & u_{nm}^{(j2)} && = u_n^{(2)} \\ mp_{4j}: \quad & i_m^{(j1)} && = i_{dc}^{(1)} \\ mp_{5j}: \quad & i_m^{(j2)} && = i_{dc}^{(2)} \\ mp_{6j}: \quad & u_{rm}^{(j)} && = u_{ref} \end{aligned} \tag{10.42}$$

where constraint mp_2 shows that the physical measurement of the star point voltage is done by the microprocessor unit. Similarly, it is the microprocessor that measures command u_{ref} and current consumption i_m for each of the power stages. The processor outputs the command voltage $\boldsymbol{u}_{cmd}$. The measurements of are conducted such that microprocessor associated with the stator (1) line has information about the measurements from the stator (2) line. The cross-measurements are not necessarily analog interface but could be implemented using data-bus communication between the microprocessors, however with a penalty in isolability of faults in interface or sensors.

Having introduced power consumption in the constraints, we revert to the motor equations and express the power balance of the motor using already available variables,

$$\begin{aligned} c_{12}: \quad & P_s^{(1)} && = (\mathbf{i}_s^{(1)})' \boldsymbol{u}_s^{(1)} \\ c_{13}: \quad & P_s^{(2)} && = (\mathbf{i}_s^{(2)})' \boldsymbol{u}_s^{(2)} \\ c_{14}: \quad & P_m && = \eta_m \left(P_s^{(1)} + P_s^{(1)} - P_0 \right) \\ c_{15}: \quad & P_m && = T_m \omega \end{aligned} \tag{10.43}$$

where η_m is a known motor efficiency and P_0 the magnetization loss, which is also known from motor data.

Matching the structure graph resulting from constraints in Equations (10.41), (10.42) and (10.43) leads to further parity relations that makes it possible to isolate faults in either of the computer units, and in each of the combination of drive and stator blocks. The structural analysis that gives these results is straight forward.

10.5.5 Analytical properties of residuals

Continuing with the next level of detail in the design, we now analyze the properties of the analytical parity relations found in the structural analysis. Isolability of faults to one or the other of the stator feed lines (computer - power stage - stator) is the key issue and weak and strong detectability are of faults are essential in this respect.

Star-point residuals. Reverting to $r_1(t)$ from Eq. (10.40),

$$\begin{aligned} r_1(t) &= 3u_n^{(1)}(t) - \sum_3 \boldsymbol{u}_t^{(1)}(t), \\ &= R_s \sum_3 \mathbf{i}_s^{(1)} + \sum_3 \frac{d}{dt}\left(\psi_s^{(1)}\right) \\ &= \sum_3 R_s \mathbf{i}_s^{(1)} + \omega \sum_3 M_{sr} \frac{\partial \boldsymbol{N}(\theta)}{\partial \theta} \mathbf{i}_r + \sum_3 L_s \frac{d}{dt}(\mathbf{i}_s^{(1)}) \\ &\quad + \sum_3 M_{sr} \boldsymbol{N}(\theta) \frac{d}{dt}(\mathbf{i}_r) + \sum_3 M_{ss} \boldsymbol{N}(-\beta) \frac{d}{dt}(\mathbf{i}_s^{(2)}) \end{aligned} \tag{10.44}$$

and a symmetrical result is obtained for $r_2(t)$.

Eq. (10.44) shows that there is strong detectability in r_1 of faults in stator 1 and in the rotor. Faults in faults in stator 2 will be weakly detectable if the resulting derivatives of currents are symmetrical, i.e. the sum of the components remain zero. The implication is that stator 2 faults will be weakly detectable or very small in r_1. With the symmetry of r_1 and r_2, stator 2 and rotor faults will be strongly detectable in r_2 and stator 1 faults will be weakly detectable.

It it noted that also faults in the stator 1 power electronics will be strongly detectable as imbalance in $\boldsymbol{u}_t$ will give rise to imbalance in $\mathbf{i}_s$.

This is a quite fortunate result as it is possible to obtain a unambiguous diagnostic result and the remedial reaction to diagnosed faults is clear,

$$\begin{aligned} H_1(r_1) \wedge H_0(r_2) &\Rightarrow \text{disable(1)} \\ H_0(r_1) \wedge H_1(r_2) &\Rightarrow \text{disable(2)} \\ H_1(r_1) \wedge H_1(r_2) &\Rightarrow \text{reduce power to motor} \end{aligned}$$

It is noted that, according to regulations, by regulatory design of the rotor and the shaft of an AC motor, rotor defects are considered impossible. If, nonetheless, a rotor defect should be developing, the total power is reduced to prevent rotor failure.

Change detection from star point residual. With an unbalance, the star point residual will differ from zero in amplitude. The star point has the fundamental frequency of $\boldsymbol{u}_t$ as a dominant component. Detection of changes in $r(t)$ therefore need be done at the fundamental frequency. Correlation at the fundamental frequency is

done through correlation over a window of N samples of the residual, and with sampling time T_s,

$$r_j(n) \quad = \quad \sum_{k=n-N}^{n} u_{nm}(nT_s)^{(j)} e^{-j\omega_s T_s k}. \tag{10.45}$$

Other residuals. With two current measurements representing electrical power to a drive, additional residuals exist but the load torque on the wheel is unknown and only one additional redundancy relation can be obtained as a reliable measure of discrepancy within the system. This residual is r_3 as measured by CPU 1 and r_4 as measured by CPU2,

$$\begin{aligned} r_3(t) &= i_{m1}^{(1)}(t) - i_{m1}^{(2)}(t) \\ r_4(t) &= i_{m2}^{(1)}(t) - i_{m2}^{(2)}(t) \end{aligned} \tag{10.46}$$

Isolation in case of a DC link sensor fault is possible from the "passive" residuals Eq. (10.46), but certain common mode faults would affect both residuals, which would prevent isolation. As the system is part of an fault-tolerant control system, active isolation can easily be performed. A perturbation signal is added to the command for each drive and the signature in the current signal $i_m^{(1)}$ and $i_m^{(2)}$ will determine which sensor could be defect.

10.5.6 Fault detection and isolation

The critical faults that need be detected and accommodated in the part of the system on which we focus, are f_1 - f_5 and f_9 of Table 10.1. The effects of each fault was investigated and the signature found in the residuals.

$$\begin{aligned} f_1 &\rightarrow \{u_n \text{ balance} \vee u_n \text{ unbalance }\} \\ f_2 &\rightarrow \{u_n \text{ unbalance}\} \\ f_3 &\rightarrow \{u_n \text{ unbalance}\} \\ f_4 &\rightarrow \{u_n \text{ unbalance}\} \\ f_5 &\rightarrow \{u_n \text{ unbalance}\} \\ f_9 &\rightarrow \{\text{DC link incorrect value}\} \end{aligned} \tag{10.47}$$

f_1 is a gate driver malfunction. If a critical fault occurs in the gate-driver it can easily be assumed that the output signals has no resemblance with valid PWM signals, except when all signals are logical zero. Some combinations of signals will enable a circuit from positive to negative supply, either in the inverter or through the stator. In both cases one or more power switches would be destroyed. This would result in stator unbalance similar to an open phase fault. Lack of unbalance only occur in the situation where all output signals are logical zero. This does not leave a signature

in the star point signal but the fault is easily identified in the DC link signal, which will be zero despite a control input is present. f_2 is a power switch short circuit. As destruction of one or more switches can be expected, the voltages applied to the stator will not be balanced. Likewise unbalanced signals are obtained if f_3 (power switch fail open) or f_4 (windings open phase) happen. f_5 is an internal turn fault in the stator. Such fault results in a reduced winding and, thus, an unbalanced stator is obtained. f_9 is a DC link sensor fault. This causes no unbalance in the stator windings but with two DC link sensors, fault isolation is possible.

Change detection and isolation. In conclusion, mean value or combined mean value and variance change detection in the two residuals $r_j(n)$, and in $\epsilon(n)$ was able to detect each of the critical faults. All critical faults were strongly detectable and application of standard CUSUM methods was straightforward. Isolation was also possible in all cases with appropriate means.

10.5.7 Experiments

The fault-tolerant architecture in Section 10.5.3 has been implemented as a laboratory test system using actual hardware. With the test system it is possible to generate selected non-destructive faults; a phase wire can be physically disconnected in system 1, each transistor in the inverter of system 2 can be disabled, and a part of a phase winding in a stator can be short circuited. Using the test system it is possible to experimentally generate faults f_1, f_2, f_3, f_4 and f_5, and detection and isolation was validated.

A number of tests were performed. Three test results (tests 2, 5, and 7) are shown below. Fig. 10.67 shows the case where a phase is physically disconnected in system 1, in Fig. 10.68 one of the inverter transistors is disabled (open) in system 2, and in Fig. 10.69 a part of a phase winding in system 1 is short circuited. The tests show the strong detectability of faults and a time to detect in the 0.1-0.3 s range, which is acceptable.

10.5.8 Evaluation of the results

This industrial case study showed how the systematic approach from component and system structure could be employed to find the properties of the overall fault-tolerant electrical steering system. It was shows how hardware, software and system functionality aspects could be combined to obtain system-wide fault-tolerance. Duplicated motors were avoided and replaced by one double stator induction motor to obtain a low-cost yet fault-tolerant solution. Using the AC motor star-point measurement and a simple measurement of total current to each drive section, the case showed how correct diagnosis was obtained for all single fault cases, both in the motor and in associated power electronics. The case considered how defects in power

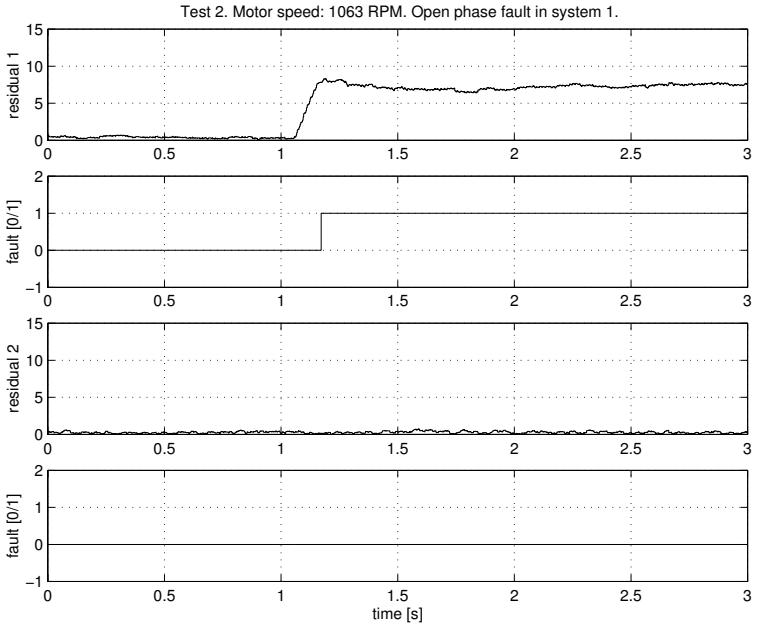


Fig. 10.67. A phase is physically disconnected in system 1 (by courtesy of J. S. Thomsen)

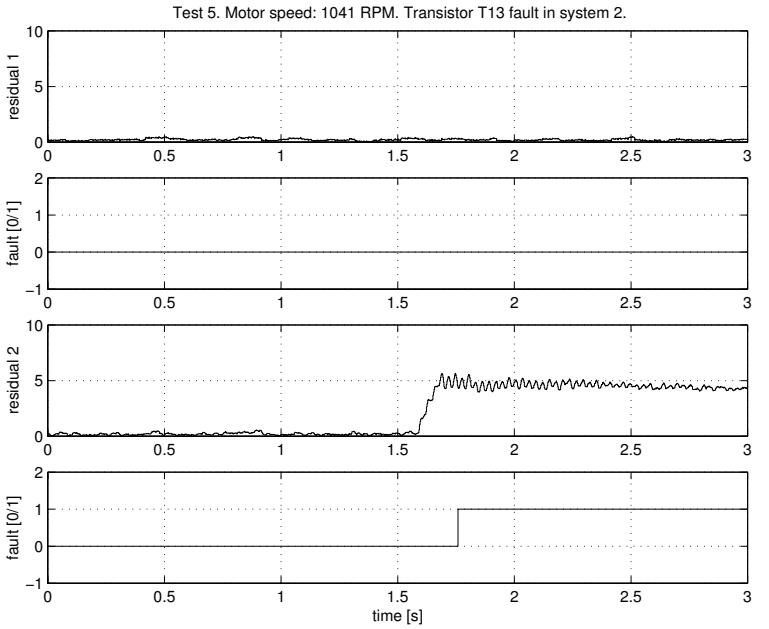


Fig. 10.68. An inverter transistor is disabled (open) in system 2 (by courtesy of J. S. Thomsen)

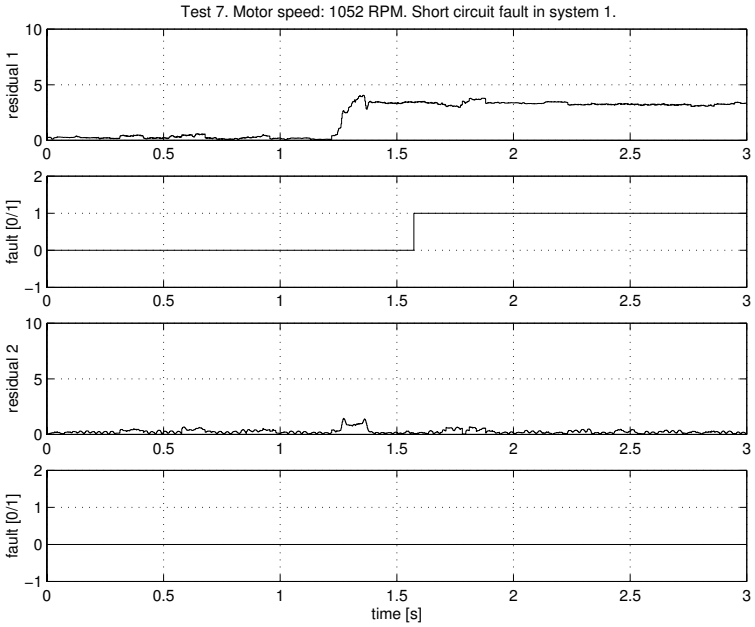


Fig. 10.69. A part of a phase winding in system 1 is short circuited (by courtesy of J. S. Thomsen)

electronics could be detected in time to allow faults to be isolated and how reconfiguration could be obtained. Finally, systematic tests of faults imposed on a warehouse truck platform demonstrated the fault-tolerant abilities of the new steering system.

10.6 Summary: Guidelines for the design of fault-tolerant control

As a summary for further applications, this section treats the architecture for implementation of autonomous supervision and describes the design process needed to achieve a fault-tolerant control algorithm in a documented and reliable way.

10.6.1 Architecture

Autonomous supervision requires development and implementation observing completeness and correctness qualities. It is important that the design of a supervised control system follows a modular approach, where each functionality can be designed, implemented, and tested independently of the remaining system. The algorithms that realise the supervisory functionality constitute themselves an increased

risk for failures in software, so the overall reliability can only be improved if the supervisory level is absolutely trustworthy.

The design of an autonomous supervisor relies heavily on having an appropriate architecture that supports clear allocation of methods to different software tasks. This is crucial for both development and verification. The latter is vital since test of the supervisor functions in an autonomous control system is a daunting task.

Supervisor. The implementation of a supervisory level onto a control system is not a trivial task. The architecture shall accommodate the implementation of diverse functions

- Support of overall coordinated plant control in different phases of the controlled process; start-up, normal operation, batch processing, event triggered operation with different control objectives, close-down.
- Support of all use-modes for normal operation and modes of operation in fault-tolerant control versions of services, for the foreseeable faults.
- Autonomous monitoring of operational status, control errors, process status and conditions.
- Fault diagnosis, accommodation and re-configuration as needed. This is done autonomously, with status information to plant-wide coordinated control.

These functions are adequately implemented in a supervisory structure with three levels in the autonomous controller, and communication to a plant-wide control as the fourth. The autonomous supervision is composed of levels 2 and 3, taking care of fault diagnosis, logic for state control and effectors for activation or calculation of appropriate remedial actions. This is illustrated in Fig. 10.70 that shows:

1. A lower level with input/output and the control loop.
2. A second level with algorithms for fault diagnosis and effectors to fault accommodation.
3. A third level with supervisor logic.
4. A fourth layer with plant-wide control and coordination.

The control level is designed and tested in each individual mode that is specified by different operational phases and different instrumentation configurations. The miscellaneous controller modes are considered separately and it is left to the supervisor design to guarantee selection of the correct mode in different situations.

The detectors are signal processing units that observe the system and compares with the expected system behaviour. An alarm is raised when an anomaly is detected. The effectors execute the remedial actions associated with fault accommodation or reconfiguration.

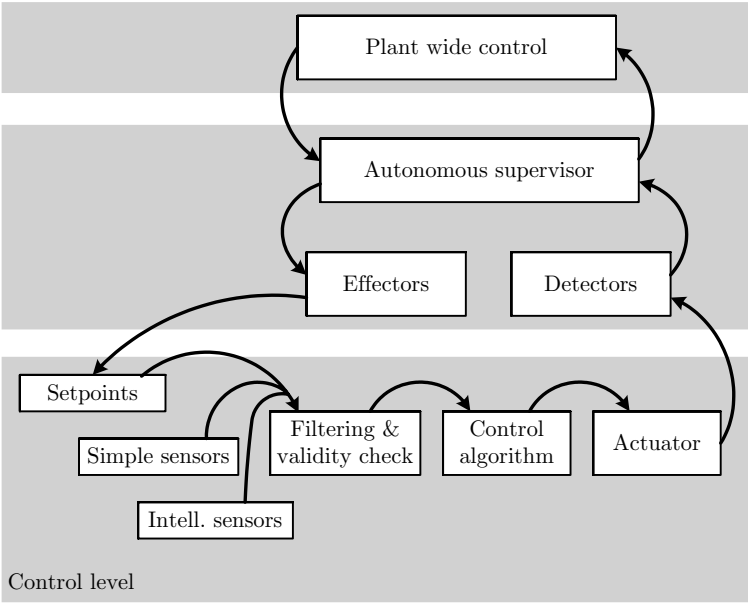


Fig. 10.70. Autonomous supervisor comprises fault diagnosis, supervisor logic and effectors, the latter to carry out the necessary remedial actions when faults are diagnosed. The upper level is plant-wide control and operator supervision.

10.6.2 Design procedure

When the level of autonomy becomes high and thereby demands a high level of reliable operation, it becomes inherently complex for the designer to cover all possible situations and guarantee correct and complete operation.

A systematic design strategy will use the analysis of fault propagation and structure as basic elements to obtain completeness of the analysis and correct operation of the system when implemented.

Component based analysis. Describe components and their interconnections using the generic model and fault propagation analysis introduced in Chapter 4. The result is a list of component related faults/failures that needs to be handled to avoid high severity end effects or critical events.

- **System breakdown:** Make, as the initial step, a top-down breakdown of the system into suitable subsystems. Make a further breakdown of subsystems into components, of types aggregated or simple.

- **Component models:** Describe the services of each component, the use-modes and services associated with each use-mode. List input-output variables associated with each version of a service. Provide fault description, effects on compo-

nent output from possible faults and failure and construct the propagation matrices associated with each version of a service.

- **Fault propagation:** Make a Fault Propagation Analysis of all relevant subsystems and combine into a complete analysis of the controlled system. The end-effects describe consequences at top level. Re-use any available fault propagation matrices for components and accumulating knowledge about component failures.

- **Severity assessment:** Judge top level end-effects for severity. The ones with significant influence on control performance, safety or availability are collected in a list for treatment by the autonomous supervisor.

- **Reverse deduction:** Make a reverse deduction of fault propagation to locate faults that would cause any of the severe end-effects, or combinations thereof, from the list.

- **Result:** The result is a short-list of faults that should be diagnosed and handled.

Structural analysis. Analyse system structure using the methods given in Chapter 5. The result gives a type of information whether sufficient redundancy is available in the system to detect and isolate each of the selected faults, and to handle the faults by a reconfiguration strategy.

- **Constraints:** Deduce an enumerated list of constraints from the set of models of the individual components.There may be different sets of constraints associated with different services.

- **Structure graph:** Use the set of constraints to formulate the system structure graph as explained in Chapter 5.

- **Matching:** Make a complete matching to find a set of unmatched constraints.

- **Constraints for residual generation:** Use unmatched constraints to provide parity equations for residual generation.

- **Ability to diagnose severe faults:** For each fault with severe end effects listed in the fault propagation analysis, verify that the system structure allows the particular fault to be diagnosed.

- **Ability to perform fault handling:** For each of the faults from the list, verify that sufficient redundancy is available in the faulty system to allow handling of the fault. For sensor faults investigate structural observability. For actuator faults, investigate structural controllability. For other faults validate the ability to control the faulty system.

Remedial actions. The possible remedial actions are designed in this step. For each of the short-listed faults, the designed must choose to utilise physical redundancy or analytical redundancy according to the results of the structural analysis. Whether the original control objectives can be met will not be known at this stage of design. The following issues need to be dealt with.

- **Fault-tolerant control version of service with full performance:** If redundant hardware is available, use this to operate with full performance.

- **Fault-tolerant control version of service with degraded performance:** Change to a control scheme that does not require the faulty component or can compensate the fault by estimating its magnitude. A fault-tolerant solution with some performance degradation is mostly an acceptable alternative to part of a plant becoming unavailable. A performance index should be available to guide controller re-design.

- **Predetermined controller re-design:** Predetermined reactions to faults can be obtained in many cases and should be preferred when possible, due to less complexity than the alternative. Predetermined solutions include estimator or controller re-design done at the design stage.

- **Autonomous controller re-design:** When remedial actions depend on the state of the system, online autonomous re-design can be necessary. Autonomous re-design is considered the most challenging of the fault-tolerant control possibilities, with a complexity equivalent to solutions in adaptive control.

- **Fail to safe state:** When appropriate fault handling cannot be achieved, the supervisor should make the system fail to a safe state. When autonomous re-design is relevant, the supervisor should comprise an independent diagnostic module for performance monitoring and must retain the ability to fail to a safe state, should proper performance of the re-design fail to be confirmed.

Fault diagnosis design. The structure information again provides a list of possibilities. The reconfigurability measure for the faulty system indicates how difficult reconstruction will be.

- **Residual generator:** Based on unmatched constraints, formulate the parity equations that can provide the basis for a residual generator.

- **Detailed design for detection:** Make a detailed design for diagnosis following the results in Section 6.2 using the parity equations approach or Section 6.4 using an optimisation-based approach.

- **Detailed design for isolation:** The selected remedial actions determine the requirements for fault isolation. It is not necessary to isolate faults below the level where the fault propagation can be stopped.

- **Detailed design for estimation:** If the remedial action requires fault estimation, design fault estimation if possible.

Control of the faulty system. Control of the faulty system is by nature of the problem, as difficult as a design of an original control system. However, if the fault is of one of the simpler types in a sensor, a remedial action is sometimes straightforward. This is also the case if the fault can be treated as an incremental change to the system, or the fault is purely additive, then results in Chapter 7 can be applied and show the family of controllers that stabilise the system. In the general case of re-design, the entire range of design methods in feedback control could be employed, however, following the discussion in Chapter 7, a methodology is recommended that is based on the formulation of a clearly defined performance specification.

- **Sensor faults:** Attempt to estimate the faulty measurement, design and employ an observer; design an output feedback without using the faulty sensor; if the type of fault is additive (bias or drift) consider a compensation through fault estimation.

- **Actuator faults:** Consider the controllability without using the faulty actuator. If possible, make a controller re-design for the faulty system using remaining actuators. If the actuator faults is physically additive, fault estimation may make it possible to make a simple compensator.

- **Plant faults:** Consider controllability (stabilisability) of the faulty system. Determine possible performance without re-tuning the controller. Investigate whether a simple re-tuning can be achieved of the faulty system using Youla-Kucera parameterisation.

- **Reconfiguration:** If other options fail, re-design the controller completely, to obtain required performance.

- **Time-to-reconfigure:** If reconfiguration is needed and complete isolation cannot be achieved within the required time to reconfigure, the set (Σ, τ) will need to be selected assuming a worst-case condition in the set of diagnostic results. The worst case fault is one that has the highest degree of severity.

- **Change objective:** When other possibilities are exhausted, relax the performance objectives for the faulty system and design an appropriate controller.

- **Fail to safe:** If the original control objectives cannot be met, handling of the problem by a the supervision function must be considered. The autonomous part

of supervision must always be to offer graceful degradation and close down when this is necessary as fall-back.

Supervisor logic. Supervisor inference rules are designed using the information about which faults/effects are detected and how they are treated. The autonomous supervisor determines the most appropriate action from the present condition and commands. The autonomous supervisor must be designed to treat mode changes of the controlled process and any overall/operator commands. Worst-case conditions and overall safety objectives should have priority when full isolation or controller-re-design cannot be accomplished within the required time to get within control specifications after a fault.

Test. Tests should be complete. The main obstacle is the complexity of the resulting hybrid system consisting of controller and plant. Transient conditions like switching between normal and not-normal controllers - and reverse - should in particular be carefully tested.

The above steps are intended to make the supervisor design cheaper, faster, and better. The fault coverage is then (hopefully) as complete as is possible, because the fault propagation analysis step in principle includes all possible faults. The analysis is modular, because small subsystems are treated individually. Furthermore, the strategy has the advantage that the system is analysed on a logical level as far as possible before the laborious job of mathematical modelling and design is initiated. This should ensure that superfluous analysis and design are avoided.

10.7 Bibliographical notes

The reconfiguration problem for the three-tank system described in Section 10.1 has been tackled as a benchmark problem in the COSY project [91]. Several solutions are described in [3], [151].

The chemical process described in Section 10.2 has been used to test and evaluate different diagnostic methods and fault-tolerant control principles. The results outlined here have been published in [156], [155] and [217]. The virtual sensor and virtual actuator example for reconfiguration is published in [162], [240]. A fault-tolerant control principle, which is based on an on-line optimisation, is described in [212]. The recent experimental results with the conductivity control problem are reported in [215].

The diagnosis and fault-tolerant control problem of the ship propulsion system described in Section 10.3 was presented as an international benchmark [105], [106] and was used as a platform for the comparison of different methods and the development of new ideas. The modelling of the ship propulsion system was described in [14], [18]. The quantised systems approach to the diagnosis of the ship system is presented in detail in [92]. The stability of an observer used in Section 10.3 has been proved in [71].

[19] presented an adaptive observer solution to estimate states and faults and used the same nonlinear observer for reconfiguration. [53] extended used a dedicated sliding mode observer for fault detection on the nonlinear propulsion plant. Two Ph.D. theses used the benchmark as a main example. Supervisor logic design was a main theme in [107]; the nonlinear problem

was the focus for [133]. [28] used a high gain observer for the nonlinear shaft speed dynamics, similar to that of [19], however not adaptive. They applied a static detector to diagnose pitch and engine gain faults. An estimate of the magnitude of sensor faults was used by the control scheme to accommodate faults by setpoint alteration. Reflecting on the nonlinear dynamics, uncertain parameters and complex designs of several earlier methods, [103] suggested a neuro-fuzzy output observer for diagnosis. [270] suggested a Kalman filter solution where non-switching fault accommodation was obtained using sensor fault estimation.

Describing the behaviour of steam generators considered in Section 10.4 results in highly nonlinear models due to the coupling of many physical phenomena of different natures. Taking into account the fact that steam generators are among the most widely spread processes, many works have been devoted to the subject, e.g. [3], [29], [189], [251].

The coefficients of the thermal model described in Section 10.4 are computed by empirical algorithms given in [251].

One of the first applications of fault-tolerant control in an industrial scale was described in [256] for mass-produced inverters for induction motors.

The process of arriving at development methods for fault-tolerant control is an evolution where some areas of application have been explored, but a final form cannot be said to be reached. Steps in the evolution include [22] presenting general ideas, [26] and [25] where experience from applying fault-tolerant methods for the the Ørsted satellite were incorporated, [107] who treat the supervisor-logic level, [257] and [256] aiming at implementation in a large volume industrial product and therefore also includes cost-benefit assessments.

Concerning implementation, a correct and consistent control system analysis should always be followed by equally correct software implementation. This is particularly relevant for the supervisory parts of an fault-tolerant control scheme [107]. Testing of the fault-tolerant control elements is difficult since it is difficult to replicate the real conditions under which faults occur. Well planned software architecture and implementation are thus crucial issues for fault-tolerant control implementation. A study of the use of object-oriented programming architectures was described by [137]. The fault-tolerant control area is hence very wide and involves several areas of system theory. One overview [193] emphasised many algorithmic essentials and the role of fault diagnosis. Another [22] presented an engineering view of the means to obtain fault-tolerant control. Formal definitions were introduced in [21]. Analysis of structure was covered in [76], [107] and [232]. Measures of recoverability were discussed in [67], [76] and [268]. Quantitative techniques to assess the reliability of fault-tolerant control implementations was the subject of [23], [266], [267].

Modeling for detection of faults in individual components have been widely studied: [38] filtered the star-point voltage of an AC motor around the fundamental frequency and used a level test to detect AC motor faults; a model for simulation of turn faults was published in [248]; detection of particular faults occurring in closed-loop AC motor actuators were studied in [249]; [111] used diagnostic methods for centrifugal pumps; [208] analysed ways to detect partial failure in power switch circuits. Performance of components was considered by [239] who analysed a multi-phase induction machine with faults. Analysis of an entire system was treated in [55] who needed a fully hardware redundant solution, with duplicated permanent motors, to cope with component faults. An AC motor with four windings was proposed for fault-tolerant systems in [239]. The case study of electrical steering originated in the methods to obtain system-wide fault-tolerance [22] and specific research results obtained in [254], in the patent [20] and in [255].

References

1. A. Aitouche, A. L. Gehin, N. Flix and G. Dumortier. Generic control/command distributed system. Application to the supervision of moving stage sets in theaters. *European Journal of Control*, 8: 64-75, 2002.
2. J. S. Albus and F. G. Proctor. A reference model architecture for intelligent hybrid control systems. *IFAC 13th World Congress*, pp. 473-488, San Francisco 1996.
3. K. J. Aström, P. Albertos, M. Blanke, A. Isidori, R. Sanz and W. Schaufelberger. *Control of Complex Systems*. Springer Verlag London, 2001.
4. K. J. Aström and B. Wittenmark. *Computer Controlled Systems: Theory and Design*. Prentice-Hall, Englewood Cliffs 1984.
5. M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes: Theory and Application*. Information and System Science, Prentice Hall, New York 1993.
6. A. Benveniste, A. Aghasaryan, E. Fabre, R. Boubour and C. Jard. Fault detection and diagnosis in distributed systems: An approach by partially stochastic petri nets. *Discrete Event Dynamic Systems: Theory and Applications*, 8: 203-231, 1998.
7. M. Bayart, A. L. Gehin and M. Staroswiecki. Fault detection and isolation and mode management in smart actuators. *IFAC SICICA'92*, Malaga 1992.
8. M. Bayart, E. Lemaire, M. A. Péraldi and C. André. External model and synccharts description of an automobile cruise control system. *Control Engineering Practice*, 7: 1259-1267, 1999.
9. M. Bayart and M. Staroswiecki. Smart actuators for distributed intelligent systems. *IFAC Symposium Distributed Intelligent Systems*, Arlington 1991.
10. M. Bayart, M. Staroswiecki and A. L. Gehin. Mode management in a distributed intelligent automated production system. *IFAC ARRTC*, Ostende 1995.
11. M. Bayart, M. Staroswiecki, F. Simonot-Lion and J. P. Thomesse. Analysis of distributed systems based on intelligent devices. *1st IEEE Workshop on Factory Communication Systems*, Leysin 1995.
12. C. Berge. *Two Theorems in Graph Theory*. Princeton University, 1957.
13. G. Biswas, M.-O. Cordier, J. Lunze, L. Travé-Massuyès and Staroswiecki, M.: Diagnosis of complex systems: Bridging the methodologies of the FDI and DX communities, Editorial, *IEEE Trans.*, SMC-34: 2159-2162, 2004.
14. M. Blanke. *Ship Propulsion Losses Related to Automatic Steering And Prime Mover Control*. PhD thesis, Technical University of Denmark, 1981.
15. M. Blanke. Consistent design of dependable control systems. *Control Engineering Practice*, 4: 1305-1312, 1996.
16. M. Blanke. Fault-tolerant sensor fusion with an application to ship navigation. *IEEE joint 2005 International Symposium on Intelligent Control and 13th Mediterranean Conference on Control and Automation*, pp. 1385-1390, 2005.

17. M. Blanke, O. Borch, F. Bagnoli and G. Allasia. Development of an automated technique for failure modes and effect analysis. *Proc. European Safety and Reliability Conf.*, pp. 839-844, Munich 1999.
18. M. Blanke and J. S. Andersen. On dynamics of large two stroke diesel engines: New results from identification. *Proceedings 9th IFAC World Conference*, Budapest 1984.
19. Blanke, M., Izadi-Zamanabadi, R., and Lootsma, T.F. (1998). Fault monitoring and reconfigurable control for a ship propulsion plant, *Journal of Adaptive Control and Signal Processing*, vol. 12, pp.671-688.
20. M. Blanke, T. Frederiksen, J. Kristensen und J. Sandberg Thomsen. *Electrical steering system.* United states patent, US 6,693,405 b2.
21. M. Blanke, C. W. Frei, F. Kraus, R. J. Patton and M. Staroswiecki. What is fault-tolerant control? *IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes*, 1: 40-51, Budapest 2000.
22. M. Blanke, R. Izadi-Zamanabadi, S. A. Bøgh and C. P. Lunau. Fault-tolerant control systems – a holistic view. *Control Engineering Practice*, 5: 693-702, 1997.
23. M. Blanke, M. Staroswiecki and N. E. Wu. Concepts and methods in fault-tolerant control. *Proc. American Control Conference*, Washington 2001.
24. M. Blanke and M. Staroswiecki: Fault-tolerant control with safe behaviour under multiple actuator or sensor faults - Theory and application. *14th IFAC Safeprocess Symposium*, Beijing 2006.
25. S. A. Bøgh. *Fault Tolerant Control Systems – A Development Method and Real-Life Case Study*. PhD thesis, Dept. of Control Eng., Aalborg University, Denmark 1997.
26. S. A. Bøgh, R. Izadi-Zamanabadi and M. Blanke. Onboard supervisor for the ørsted satellite attitude control system. *Artificial Intelligence and Knowledge Based Systems for Space, 5th Workshop*, pp. 137-152, Noordwijk 1995.
27. B. Buchberger. Gröbner bases: An algorithmic method in polynomial ideal theory. *Multidimensional Systems Theory*, pp. 184-232, Dordrecht Reidel 1985.
28. Bonivento C., Paoli A. and Marconi L. (2003). Fault-Tolerant Control for the Ship Propulsion System, *Control Engineering Practice*, vol. 11, pp. 483-492.
29. O. Boumaman, G. Dauphin-Tanguy. Bond graph model of a steam generator process and its environment. *10-th European Simulation Multiconference*, pp. 238-242, Budapest 1996.
30. A. Bouras, M. Bayart and M. Staroswiecki. Specification of smart instruments for distributed intelligent control. *IEEE International Conference on Systems*, Vancouver 1995.
31. A. Bouras and M. Staroswiecki. Building distributed architectures by the interconnection of intelligent instruments. *IFAC INCOM'98*, Nancy 1998.
32. A. Bouras and M. Staroswiecki. How can intelligent instruments interoperate in an application framework? A mechanism for taking into account operating constraints. *IFAC SICICA'97*, Annecy 1997.
33. R. Bukharaev. *Theorie der stochastischen Automaten*. B. G. Teubner, Stuttgart 1995.
34. S. L. Campbell and R. Nikoukhah. *Auxiliary signal design for failure detection*. Princeton University Press, Princeton, N.J. 2004.
35. C. Cao, F. Lin and Z. Lin. Why event observation: Observability revisited. *Discrete Event Dynamic Systems: Theory and Applications*, 7: 127-149, 1997.
36. J. Carlyle. State-calculable stochastic sequential machines, equivalences and events. Switching circuit theory and logic. *IEEE Conf. Rec. Switch. Circuit Th. and Logic Design*, pp. 865-870, 1965.
37. T. Carpentier, R. Litwak and J.-Ph. Cassar. Criteria for the evaluation of FDI systems – Application to sensors location. *IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes*, pp. 1083-1088, Hull 1997.
38. M. A. Cash, T. G. Habetler and G. B. Kliman. Insulation failure prediction in induction machines using line-neutral voltages. *IEEE Transactions on Industry Applications*, 54: 1234-1239, 1998.

39. C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.
40. J. P. Cassar, M. Bayart and M. Staroswiecki. Hierarchical data validation in control systems using smart actuators and sensors. *IFAC Symposium on Intelligent Components and Instruments for Control Application (SICICA'92)*, Malaga 1992.
41. G. Chartrand and O. R. Oellermann. *Applied and algorithmic graph theory. Pure and applied mathematics*. McGraw-Hill Inc., 1993.
42. C. T. Chen. *Linear System Theory and Design*. Holt, Rinehart and Winston, 1984.
43. J. Chen and R. J. Patton. *Robust Model-Based Fault Diagnosis for Dynamic Systems*. Kluwer Academic, 1999.
44. E. Y. Chow and A. S. Willsky. Analytical redundancy and the design of robust failure detection filters. *IEEE Trans.*, AC-29: 603-614, 1984.
45. M. -O. Cordier, P. Dague, F. Lévy, M. Dumas, J. Montmain, M. Staroswiecki, and L. Travé-Massuyès. AI and automatic control approaches of model-based diagnosis: links and underlying hypothesis. *IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes*, pp. 274 - 279, Budapest 2000.
46. D. Cox, J. Little, J. and D. O'Shea. *Varieties and Algorithms*, Springer, New York 1992.
47. P. Declerck and M. Staroswiecki. Characterisation of the canonical components of a structural graph for fault detection in large scale industrial plants. *Proc. European Control Conference*, Grenoble 1991.
48. X. Ding and P. M. Frank. Frequency domain approach and threshold selector for robust model-based fault detection and isolation. *IFAC Symp. Safeprocess*, 1: 307-312, Baden-Baden 1991.
49. S. Diop. Elimination in control theory. *Math. Control Signals Systems*, 4: 17-32, 1991.
50. A. L. Dulmage and N. S. Mendelsohn. Covering of bi-partite graphs. *Canada J. Math.*, 10: 517-534, 1958.
51. A. L. Dulmage and N. S. Mendelsohn. A structure theory of bi-partite graphs of finite exterior dimension. *Trans. of Royal Soc. Canada, Section III*, 53: 1-13, 1959.
52. J. Edmonds. Paths, trees and flowers. *Canad. J. of Math.*, 17: 449-467, 1965.
53. Edwards, C. and S. K. Spurgeon (2000). A sliding mode observer based FDI scheme for the ship benchmark. *European Journal of Control*, vol 6(4) pp 341-356.
54. A. Emami-Naeini and M. M. Akhter and S. M. Rock. Effect of model uncertainty on failure detection – The threshold selector. *IEEE Trans.*, AC-33: 1106-1115, 1988.
55. N. Ertugrul, W. Soong, G. Dostal and D. Saxon. Fault tolerant motor drive system with redundancy for critical applications. *Proc. IEEE 33rd Annual Power Electronics Specialists Conference*, pp. 1457-1462, 2002.
56. P. M. Frank. Analytical and qualitative model-based fault diagnosis – A survey and some new results. *European Journal of Control*, 2: 6-28, 1996.
57. P. M. Frank and X. Ding. Frequency domain approach to optimally robust residual generation and evaluation using model-based fault diagnosis. *Automatica*, 30: 789-804, 1994.
58. L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canad. J. Math.*, 8: 399-404, 1956.
59. L. R. Ford and D. R. Fulkerson. A simple algorithm for finding maximal network flows and an application to the hitchcock problem. *Canad. J. Math.*, 9: 210-218, 1957.
60. D. Förstner, M. Jung and J. Lunze. A discrete-event model of asynchronous quantised systems. *Automatica*, 38: 1277-1286, 2002.
61. D. Förstner and J. Lunze. Qualitative modeling of a power stage for diagnosis. *13th International Workshop on Qualitative Reasoning*, Loch Awe 1999.
62. D. Förstner and J. Lunze. Qualitative model-based fault detection of a fuel injection system. *IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes*, pp. 88-93, Budapest 2002.
63. D. Förstner and J. Lunze. Discrete-event models of quantised systems for diagnosis. *Intern. J. Control*, 74: 690-700, 2001.

64. D. Förstner, R. Weber and J. Lunze. Diagnose eines Diesel-Einspritzsystems mit ereignisdiskreten Modellen, *Automatisierungstechnische Praxis* 45: 73-79, 2003.
65. P. M. Frank. *Diagnosis in Dynamical Systems via State Estimation – A Survey*. 1: 35-98, D. Reidel Publishing Company, 1987.
66. C. W. Frei. *Fault-Tolerant Methods in Anesthesia*, PhD Thesis, EHT Zürich, 2000.
67. C. W. Frei, F. J. Kraus and M. Blanke. Recoverability viewed as a system property. *Proc. European Control Conference*, Karlsruhe 1999.
68. C. Fritsch, J. Lunze, Complexity reduction of remote diagnosis of discrete-event systems, *Proc. MTNS*, Kyoto 2006.
69. C. Fritsch, J. Lunze, M. Schwaiger and V. Krebs, Remote diagnosis of discrete-event systems with on-board and off-board components, *IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS)*, Beijing 2006.
70. Z. Gao and P. J. Antsaklis. Stability of the pseudo-inverse method for reconfigurable control systems. *International Journal of Control*, 53: 717-729, 1991.
71. J. P. Gauthier, H. Hammouri and S. Othman. A simple observer for nonlinear systems applications to bioreactors. *IEEE Trans.*, AC-37: 875-880, 1992.
72. A. L. Gehin, M. Assas and M. Staroswiecki. Structural analysis of system reconfigurability. *IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes*, pp. 292 - 297, Budapest 2000.
73. A. L. Gehin, M. Bayart and M. Staroswiecki. Faulty resources management in automation systems. *IEEE SMC'97*, Orlando 1997.
74. A. L. Gehin and M. Staroswiecki. A formal approach to reconfigurability analysis. Application to the three tank benchmark. *Proc. European Control Conference*, Karlsruhe 1999.
75. A. L. Gehin, M. Staroswiecki and M. L. Assas. A bottom-up approach to analyse reconfiguration possibilities. *10th International Workshop in Principles of Diagnosis*, Loch Awe 1999.
76. A. L. Gehin and M. Staroswiecki. A formal approach to reconfigurability analysis – Application to the three tank benchmark. *Proc. European Control Conference*, Karlsruhe 1999.
77. A. Georgiou and C. A. Floudas. Structural analysis and synthesis of feasible control systems: Theory and applications. *Chemical Engineering Research and Design*, 67: 600-618, 1989.
78. J. Gertler. *Fault Detection and Diagnosis in Engineering Systems*. Marcel Dekker, New York 1998.
79. J. Gertler. Analytical redundancy methods in failure detection and isolation. *Control Theory and Advanced Technology*, 1(9): 259-285, 1993.
80. J. Gertler and D. Singer. A new structural framework for parity space equation based failure detection and isolation. *Automatica*, 26: 381-388, 1990.
81. S. T. Glad. Non-linear state space and input-output descriptions using differential polynomials. *New Trends in Nonlinear Control Theory*. Control and Information Science, 122: 182-189, Springer, Berlin 1989.
82. K. Golver and L. M. Silverman. Characterisation of structural controllability. *IEEE Trans.*, AC-21, 4: 534-537, 1976.
83. M. Gondran and M. Minoux. Graphes et algorithmes. *Coll. Direction des Etudes et Recherches EDF, Eyrolles*, (3d edition), 1995.
84. S. F. Graebe, A.L.B. Ahlén. Dynamic transfer among alternative controllers and its relation to anti-windup controller design. *IEEE Transactions on Control System Technology*, 4: 92-99, 1996.
85. M. Green and D.J.N. Limebeer. *Linear Robust Control*. Prentice-Hall, Englewood Cliffs, New Jersey, 1995.
86. W. Hamscher, L. Console and J. de Kleer. *Readings in Model-Based Diagnosis*. Morgan Kaufmann, 1992.
87. T. Hanus, M. Kinnaert and J.L. Henrotte. Conditioning Technique, a General Anti-Windup and Bumpless Transfer Method. *Automatica*, 3: 729-739, 1987.

88. Hanus, R. and Y. Peng. Conditioning Technique for Controllers with Time Delays. *IEEE Transactions on Automatic Control*, 37: 689-692, 1992.
89. F. Harary. A graph theoretic approach to matrix inversion by partitioning. *Numer. Math.*, 4: 128-135, 1962.
90. D. Harel. Statecharts, a visual formalism for complex systems. *Science of Computer Programms*, 89:231-274, 1987.
91. B. Heiming and J. Lunze. Definition of the three-tank benchmark problem for controller reconfiguration. *European Control Conference*, Karlsruhe 1999. http://www.ruhr-uni-bochum. de/atp.
92. D. Herrmann. *Qualitative Fehlerdiagnose im Automatennetz am COSY Ship Propulsion Benchmark.* Diplomarbeit, TU Hamburg-Harburg, 2000.
93. S. A. Herrin. Maintainability applications using the matrix FMEA technique. *IEEE Trans.*, R-30: 212-217, 1981.
94. D. M. Himmelblau. *Fault Detection and Diagnosis in Chemical and Petrochemical Processes*. Elsevier, Amsterdam 1978.
95. J. E. Hopcroft and R. M. Karp. An algorithm for maximum matchings in bipartite graphs. *SIAM J. Comp.*, 2: 225-231, 1973.
96. R. F. Huang, C. Y. Stangel. Restructurable control using proportional-integral implicit model following. *J. Guidance, Control and Dynamics*, 13: 303-309, 1990.
97. R. Isermann. Process fault detection based on modelling and estimation methods: A survey. *Automatica*, 20(4): 387-404, 1984.
98. R. Isermann. *Fault-Diagnosis Systems*. Springer-Verlag, Berlin 2006.
99. R. Isermann. Fault diagnosis of machines via parameter estimation and knowledge processing. *Automatica*, 29(4): 815-836, 1993.
100. R. Isermann and P. Ballé. Trends in the application of model-based fault detection and diagnosis of technical processes. *Control Engineering Practice*, 5(5): 709-719, 1997.
101. Y. Iwasaki and H. A. Simon. Causality in device behaviour. *Artificial Intelligence*, 29: 3-32, 1986.
102. R. Izadi-Zamanabadi, P. Amann, M. Blanke, V. Cocquempot, G. L. Gissinger, E. C. Kerrigan, T. F. Lootsma, J. M. Perronne and G. Schreier. *Ship propulsion control and reconfiguration*. in [3], pp. 285-315.
103. R. Izadi-Zamanabadi, M. Blanke and S. Katebi (2003). Cheap diagnosis using structural modelling and fuzzy-logic based detection. *Control Engineering Practice* vol. 11(4) pp 415-421.
104. J. Isaksson. An on-line threshold selector for for failure detection. *Proc. Int. Conf. on Fault Diagnosis: TOOLDIAG*, pp. 628-634, Toulouse 1993.
105. R. Izadi-Zamanabadi and M. Blanke. Ship propulsion system as a benchmark for fault-tolerant control. Technical report, Control Engineering Dept., Aalborg University, Denmark 1998.
106. R. Izadi-Zamanabadi and M. Blanke. A ship propulsion system as a benchmark for fault-tolerant control. *Control Engineering Practice*, 7: 227-239, 1999.
107. R. Izadi-Zamanabadi. *Fault-Tolerant Supervisory Control – System Analysis and Logic Design*. PhD thesis, Dept. of Control Eng., Aalborg University, Denmark 1999.
108. P. Jalote. *Fault Tolerance in Distributed Systems*. Prentice-Hall, 1994.
109. A. H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, 1970.
110. N. R. Jennings, J. A. Pople and E. H. Mamdani. Designing a reusable coordination module for cooperative industrial control application. *IEE Proceedings on Control Theory and Applications*, pp. 91-102, 1996.
111. K. S. Kallesøe. Fault Detection and Isolation in Centrifugal Pumps. PhD thesis, *Department of Control Engineering and Grundfos A/S*, 2005.
112. D. Karnopp and R. C. Rosenberg. *Systems Dynamics. A Unified Approach*. Wiley Intersciences, New York 1975.

113. M. Kinnaert, Y. Peng. Residual generation for sensor and actuator fault detection and isolation, a frequency domain approach. *Intern. J. Control* 61: 1423-1435, 1995.
114. M. Kinnaert, D. Vrancic, E. Denolin, D. Juricic and J. Petrovcic. Model-based fault detection and isolation for agas-liquid separation unit. *Control Engineering Practice*, 8: 1273-1283, 2000.
115. M. Kokar. On consistent symbolic representations of general dynamic systems. *IEEE Trans.*, AC-40: 1231-1242, 1995.
116. J. Korbicz, J. Koscielny, Z. Kowalczuk and W. Cholewa. *Fault Diagnosis*. Springer-Verlag, Berlin 2004.
117. S. Kowalewski, S. Engell, J. Preißig and O. Stursberg. Verification of logic controllers for continuous plants using timed condition/event-system models. *Automatica*, 35: 505-518, 1999.
118. B. H. Krogh and A. Chutinan. Computing polyhedral approximations to flow pipes for dynamic systems. *IEEE Conf. on Decision and Control*, paper TM-01, 1998.
119. V. Kucera. *Analysis and Design of Discrete Linear Control Systems*. Prentice-Hall 1991.
120. H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2: 83-97, 1956.
121. G. Lamperti and M. Zanella, *Diagnosis of Active Systems*, Kluwer Academic Publishers, Dordrecht 2003.
122. J. M. Legg. Computerized approach for matrix-form FMEA. *IEEE Trans.*, R-27: 254-257, 1978.
123. A. Leitold and K. M. Hangos. Structural solvability analysis of dynamic process models. *Computers and Chemical Engineering*, 25: 1633-1646, 2001.
124. G. Lichtenberg, J. Lunze, R. Scheuring and J. Schröder. Prozeßdiagnose mittels qualitativer Modelle am Beispiel eines Wasserstoffverdichters. *Automatisierungstechnik*, 47: 101-109, 1999.
125. G. Lichtenberg and A. Steele. An approach to fault diagnosis using parallel qualitative observers. *Workshop on Discrete Event Systems*, pp. 290-295, 1996.
126. C. T. Lin. Structural controllability. *IEEE Trans.*, AC-19: 201-208, 1974.
127. C. T. Lin. System structure and minimal structure controllability. *IEEE Trans.*, AC-22: 855-862, 1977.
128. F. Lin. Diagnosability of discrete event systems and its applications. *Discrete Event Dynamic Systems: Theory and Applications*, 4: 197-212, 1994.
129. F. Lin and W. Wonham. On observability of discrete-event systems. *Information Sciences*, 44: 173-198, 1988.
130. X. Lin, M. O. Tade and R. B. Newell. Output structural controllability condition for the synthesis of control systems for chemical processes. *Int. J. Systems Sci.*, 22: 107-132, 1991.
131. M. Lind. Modeling goals and functions of complex industrial plants. *Applied Artificial Intelligence*, 8: 259-283, 1994.
132. L. Ljung. *System Identification: Theory for the User*. Prentice-Hall, 1999.
133. T. F. Lootsma. Observer-based fault detection and isolation for nonlinear systems. Ph.D. thesis, Department of Control Engineering, Aalborg University, Denmark 2001.
134. T. Lorentzen and M. Blanke. Industrial use of structural analysis - a rapid prototyping tool in the public domain. *Proc. ACD Workshop*, pp. 166-171, Karlsruhe 2004.
135. X. C. Lou, A. S. Willsky and G. C. Verghese. Optimally robust redundancy relations for failure detection in uncertain systems. *Automatica*, 22: 333-344, 1986.
136. C. Luh and B. Zeigler. Abstracting event-based control models for high autonomy systems. *IEEE Trans.*, SMC-23: 42-54, 1993.
137. C. P. Lunau. A reflective architecture for process control applications. In M. Aksit and S. Matsuoka, editors, *ECOP'97 Object Oriented Programming*, pp. 170-189. Springer Verlag, 1997.
138. J. Lunze. Qualitative modelling of linear dynamical systems with quantized state measurements. *Automatica*, 30: 417-431, 1994.

139. J. Lunze. *Künstliche Intelligenz für Ingenieure, Band 2*. Oldenbourg-Verlag, München 1995.
140. J. Lunze. Stabilisation of nonlinear systems by qualitative feedback controllers. *Intern. J. Control*, 62: 109-128, 1995.
141. J. Lunze. On the Markov property of quantized state measurement sequences. *Automatica*, 34: 1439-1444, 1998.
142. J. Lunze. A timed discrete-event abstraction of continuous-variable systems. *Intern. J. Control*, 72: 1147-1164, 1999.
143. J. Lunze. Diagnosis of quantised systems based on a timed discrete-event model. *IEEE Trans.*, SMC-30: 322-335, 2000.
144. J. Lunze, *Automatisierungstechnik*. Oldenbourg-Verlag, München 2003.
145. J. Lunze. Complexity reduction in state observation of stochastic automata. *Workshop on Discrete Event Systems*, pp. 349-354, Reims 2004.
146. J. Lunze. Control reconfiguration, in Topic 6.43 Control Systems, Robotics and Automation, edited by H. Unbehauen. in Encyclopedia of Life Support Systems (EOLSS), Eolss Publishers, Oxford 2004.
147. J. Lunze. The state partitioning problem of quantized systems, in Blondel, V. D.; Megretski, A.: *Unsolved Problems in Mathematical Systems and Control Theory*, Princeton University Press, pp. 134-139, Princeton 2004.
148. J. Lunze, *Ereignisdiskrete Systeme*. Oldenbourg-Verlag, München 2006.
149. J. Lunze, Control reconfiguration after actuator failures: The generalised virtual actuator. *IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS)*, Beijing 2006.
150. J. Lunze, J. Askari-Marnani, A. Cela, P. M. Frank, A. L. Gehin, B. Heiming, M. Lemos, T. Marcu, L. Rato and M. Staroswiecki. Three tank control reconfiguration. In [3].
151. J. Lunze, J. Askari-Marnani and B. Heiming. Controller reconfiguration based on qualitative model: A solution of three-tanks benchmark problem. *European Control Conference*, Karlsruhe 1999.
152. J. Lunze, J. Neidig: Decentralised diagnosis of automata networks. *16-th IFAC World Congress*, paper Th-A05-TO/4, Prague 2005.
153. J. Lunze, B. Nixdorf and J. Schröder. Deterministic discrete-event representations of continuous-variable systems. *Automatica*, 35: 101-109, 1999.
154. J. Lunze and F. Schiller. Logic-based process diagnosis utilising the causal structure of dynamical systems. *Preprints of IFAC/IFIP/IMACS Int. Symp. on Artificial Intelligence in Real-time Control*, pp. 649-654, Delft 1992.
155. J. Lunze and J. Schröder. Application of qualitative observation and prediction to a neutralisation process. *Proceedings of 14th IFAC Congress*, 1: 49-54, Beijing 1999.
156. J. Lunze and J. Schröder. Process diagnosis based on a discrete-event description. *Automatisierungstechnik*, 47: 358-365, 1999.
157. J. Lunze and J. Schröder. State observation and diagnosis of discrete-event systems described by stochastic automata. *Discrete Event Dynamic Systems: Theory and Applications*, 11, 2001.
158. J. Lunze and J. Schröder. Connections between qualitative dynamical models and the Frobenius-Perron operator, *Automatisierungstechnik*, 51: 471-479, 2003.
159. J. Lunze and J. Schröder. Sensor and actuator fault diagnosis of systems with discrete inputs and outputs, *IEEE Trans.*, SMC-34: 1096-1107, 2004.
160. J. Lunze and T. Steffen. Control reconfiguration by means of a virtual actuator, *IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS)*, Washington 2003.
161. J. Lunze and T. Steffen. Control reconfiguration demonstrated at a two-degrees-of-freedom helicopter model, *European Control Conference*, Cambridge 2003.
162. J. Lunze and T. Steffen. Rekonfiguration linearer Systeme bei Aktor- und Sensorfehlern. *Automatisierungstechnik*, 43, 2003.

163. J. Lunze and T. Steffen. Control reconfiguration after actuator failures using disturbance decoupling methods. *IEEE Trans.* AC-51 (2006) No. 9.
164. J. M. Maciejowski. *Predictive Control with Constraints*. Prentice-Hall, 2002.
165. C. Maffezzoni, L. Ferrarini and E. Carpenzano. Object oriented models for advanced automation engineering. *9th Symposium on Information Control in Manufacturing*, pp. 21-31, 1998.
166. J. F. Magni and P. Mouyon. On residual generation by observer and parity space approaches. *IEEE Trans.*, AC-39: 441-447, 1994.
167. M. Mahmoud, J. Jiang and Y. Zhang. *Active Fault Tolerant Control Systems, Stochastic Analysis and Synthesis*. Springer-Verlag, London 2003.
168. R. S. Mangoubi. *Robust Estimation and Failure Detection*. Springer-Verlag, 1998.
169. M. Meyer, J. -M. Le Lann, B. Koehret and M. Enjalbert. Optimal selection of sensor location on a complex plant using a graph oriented approach. *Computer Chemical Eng.*, 18: 535-540, 1994.
170. B. C. Moore. Principal component analysis in linear systems: controllability, observability and model reduction. *IEEE Trans.* AC-26: 17-32, 1981.
171. M. Morari and G. Stephanopoulos. Studies in the synthesis of control structures for chemical processes. Part ii: Structural aspects and the synthesis of alternative feasible control. *AIChE Journal*, 40: 232-246, 1980.
172. K. Murota. *Systems analysis by graphs and matroïds. Structural solvability and controllability*. Springer Verlag, 1987.
173. J. M. Nahman. *Dependability of Engineering Systems*. Springer-Verlag, Berlin 2002.
174. J. Neidig, C. Falkenberg, J. Lunze and M. Fritz. Qualitative diagnosis of an automative air path. *atp international*, 3: 37-42, 2005.
175. R. Nikoukhah and S.L. Campbell. A multi-model approach to failure detection in uncertain sampled-data systems. *European Journal of Control*,11:1-11, 2005.
176. H. Niemann, and J. Stoustrup. An architecture for fault tolerant controllers. *International Journal of Control*, 78: 1091-1110, 2005.
177. H. Niemann and N. K. Poulsen. Active fault diagnosis in closed-loop systems. *IFAC World Congress*, Prague 2005.
178. H. Niemann. Dual Youla parameterization. *IEE Proceedings - Control Theory and Applications*, 150: 493-497, 2003.
179. I. Nikiforov. A simple recursive algorithm for diagnosis of abrupt changes in signals and systems. *American Control Conference*, pp. 1938-1942, 1998.
180. I. V. Nikiforov. A simple change detection scheme. *Signal Processing*, 81: 149-172, 2001.
181. R. Nikoukhah. Innovation generation in the presence of unknown inputs: application to robust failure detection. *Automatica*, 30: 1851-1868, 1994.
182. H.H. Niemann. *Robust Control, Fault Diagnosis and Fault-Tolerant Control - a Standard Setup Approach*. Lecture Notes. Automation at Ørsted·DTU, 2002.
183. H. H. Niemann and J. Stoustrup. Gain scheduling using the Youla parameterization. *IEEE Conference on Decision and Control*, pp. 2306-2311, Phoenix 1999.
184. H. H. Niemann and J. Stoustrup. Design of fault detectors using $\mathcal{H}_\infty$ optimisation. *Proceedings of the 39th IEEE Conference on Decision and Control*, 4237-4238, 2000.
185. H.H. Niemann and J. Stoustrup. Reliable control using the primary and dual Youla parameterization. *41st IEEE Conf. Decision and Control*, pp. 4353-4358, Las Vegas 2002.
186. H. Niemann and J. Stoustrup. An Architecture for Fault Tolerant Controllers. *Int. J. Control*, 78: 1091-1110, 2005.
187. M. Nyberg. *Model Based Fault Diagnosis: Methods, Theory and Automotive Engine Applications*. PhD thesis, Linköping University, Department of Electrical Engineering, 1999.
188. M. Nyberg. Criterions for detectability and strong detectability of faults in linear systems. *International Journal of Control*, 7: 490-501, 2002.
189. A. W. Ordys. *Modeling and Simulation of Power Generation Plants.* Springer-Verlag 1994.

190. B. Ould Bouamama, A. L. Gehin and M. Staroswiecki. Alarm filtering by component modelling and bond graph approach. *4th IFAC Workshop on On-line Fault Detection and Supervision in the Chemical Process Industries (CHEMFAS-4)*, Seoul, 2001.
191. A. S. Özveren and A. Willsky. Observability of discrete event dynamic systems. *IEEE Trans.*, AC-35: 797-806, 1990.
192. A. Papoulis. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, 1965.
193. R. J. Patton. Fault-tolerant control: The 1997 situation. *IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes*, pp. 1033-1055, Hull 1997.
194. R. J. Patton, P. M. Frank and R. N. Clark (Eds). *Fault Diagnosis in Dynamic Systems Theory and Application*. Prentice Hall, New York 1989.
195. R. J. Patton; P. M. Frank and R. Clark (Eds.). *Issues of Fault Diagnosis for Dynamical Systems*. Springer-Verlag, London 1999.
196. L. Pau. *Failure Diagnosis and Performance Monitoring*. Marcel Dekker, New York 1981.
197. X. Paynter. *Analysis and Design of Engineering Systems*. MIT Press, 1961.
198. Y. Peng, A. Youssouf, Ph. Arte and M. Kinnaert. A complete procedure for residual generation and evaluation with application to heat exchanger. *IEEE Trans.*, CST-5: 542-555, 1997.
199. Y. Peng, Vrancic, D. and R. Hanus. Anti-windup, bumpless, and conditioned transfer techniques for PID controllers. *IEEE Control System Magazine*, 10: 48-57, 1996.
200. L. Portinale. Behavioural petri nets: A model for diagnostic knowledge representation and reasoning. *IEEE Trans.*, SMC-27: 184-195, 1997.
201. M. J. H. Pijpers, H. Preisig and M. Weiss. A discrete modelling procedure for continuous processes based on state discretization. *Proc. 2nd MATHMOD Conf.* pp. 189-194, Vienna 1997.
202. K. R. Popper. *Conjectures and Refutations.* Routledge and Kegan Paul, London 1963.
203. M.-I. Brudny. *Karl Popper: Un Philosophe Heureux.* Grasset, Paris 2002.
204. J. Raisch. Nondeterministic automata as approximations for continuous systems – An approach with an adjustable degree of accuracy. *IMACS Symposium on Mathematical Modelling*, pp. 195-202, 1997.
205. P. Ramadge and W. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, 25: 206-230, 1987.
206. H. E. Rauch. Autonomous control reconfiguration. *IEEE Control Systems Magazine*, pp. 37-48, December 1995.
207. K. J. Reinschke. *Multivariable Control: A Graph Theoretic Approach*. Springer-Verlag, 1988.
208. R. L. A. Ribeiro, C. B. Jacobinna, E. R. C. da Silva and A. M. N. Lima. Fault-tolerant voltagefed pwm inverter ac motor drive systems. *IEEE Transactions on Industrial Electronics*, 51: 439-446, 2004.
209. A. Saberi, A. A. Stoorvogel and P. Sannuti. Exact, almost and optimal input decoupled (delayed) observers. *Int. J. Control*, 73: 552-582, 2000.
210. M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen and D. Teneketzis. Diagnosability of discrete event systems. *IEEE Trans.*, AC-40: 1555-1575, 1995.
211. M. Sampath, R. Sengupta, S. Lafurtune, K. Sinnamohideen and D. Teneketzis. Failure diagnosis using discrete event models. *IEEE Trans.*, CST-4, 1996.
212. F. Schiller and J. Schröder. Combining qualitative model-based diagnosis and observation with fault-tolerant systems. *AI Communications*, 12: 79-98, 1999.
213. F. Schiller, J. Schröder and J. Lunze. Diagnosis of transient faults in quantised systems, *Engineering Applications of Artificial Intelligence*, 14: 519-536, 2001.
214. C. Schizas and F. J. Evans. A graph theoretic approach to multivariables control system design. *Automatica*, 17: 371-377, 1981.
215. T. Schlage. *Rekonfiguration einer Prozessregelung mittels virtuellem Aktor*. Diplomarbeit, Ruhr-Universität Bochum, Lehrstuhl für Automatisierungstechnik und Prozessinformatik, 2006.

216. J. Schröder, F. Schiller and J. Lunze. Diagnosis of transient faults in quantised systems. *IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes*, Budapest, 2000.
217. J. Schröder. *Modelling, State Observation and Diagnosis of Quantised Systems*. Springer-Verlag, Berlin, 2002.
218. A. Seidenberg. An elimination theory for differential algebra. *Univ. California Pul. Math.*, 3: 31-65, 1956.
219. S. Simani, C. Fantuzzi and R. R. Patton. *Model-Based Fault Diagnosis in Dynamic Systems Using Identification Techniques*. Springer-Verlag, 2002.
220. S. Skogestad and I. Postlethwaite. *Multivariable Feedback Control: Analysis and Design*. 3rd edition, Wiley, 2005.
221. T. Söderström and P. Stoica. *System Identification*. Prentice-Hall, 1989.
222. V. Srinivasan and M. Jafari. Fault detection/monitoring using timed petri nets. *IEEE Trans.*, SMC-23, 1993.
223. H. Starke. *Abstrakte Automaten*. Deutscher Verlag der Wissenschaften, Berlin, 1969.
224. M. Staroswiecki, G. Hoblos, A. Aitouche. Fault tolerance analysis of sensor systems. *IEEE Conf. on Decision and Control*, Phoenix 1999.
225. M. Staroswiecki. Les actionneurs intelligents. *Revue Générale de l'Electricité*, 7: 30-34, 1990.
226. M. Staroswiecki. Fault tolerant control: the pseudo-inverse method revisited. *16-th IFAC World Congress*, paper Th-E05-TO/2, Prague 2005.
227. M. Staroswiecki. fault tolerant control using an admissible model matching approach. *CDC-ECC*, paper TuA10.6, Sevilla 2005.
228. M. Staroswiecki. Distributed system supervision based on intelligent instruments. *International Conference on Fault Diagnosis, TOOLDIAG'93*, Toulouse 1993.
229. M. Staroswiecki and A. L. Gehin. Analysis of system reconfigurability using generic component models. *UKACC Control'98*, Swansea 1998.
230. M. Staroswiecki and M. Bayart. *Les actionneurs intelligents*. Hermès, Paris 1994.
231. M. Staroswiecki and M. Bayart. Models and languages for the interoperability of smart instruments. *Automatica*, 32: 859-873, 1996.
232. M. Staroswiecki, S. Attouche and M. L. Assas. A graphic approach for reconfigurability analysis. *10th Int. Workshop on Principles of Diagnosis*, Loch Awe 1999.
233. M. Staroswiecki, J. P. Cassar and P. Declerck. A structural framework for the design of FDI in large scale industrial plants. In [195].
234. M. Staroswiecki and P. Declerck. Analytical redundancy in non-linear interconnected systems by means of structural analysis. *IFAC/IMACS/IFORS Conf. AIPAC' 89*, Nancy, France, 1989.
235. M. Staroswiecki and A. L. Gehin. Analysis of system reconfigurability using generic component models. *CONTROL'98* pp. 1157-1162, Swansea 1998.
236. M. Staroswiecki and A. L. Gehin. Control, fault tolerant control and supervision problems. *IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes*, Budapest 2000.
237. M. Staroswiecki, G. Hoblos and A. Aitouche. Fault tolerance analysis of sensor systems. *IEEE Conf. on Decision and Control*, Phoenix 1999.
238. M. Staroswiecki and G. Comtet Varga. Analytical redundancy relations for fault detection and isolation in algebraic dynamical systems, *Automatica*, 37: 687-699, 2001.
239. G. K. Singh, V. Pant. Analysis of a multiphase induction machine under fault condition in a phase-redundant a.c. drive system. *Electric machines and Power Systems*, 28: 577-590, 2000.
240. T. Steffen. *Control Reconfiguration of Dynamical System: Linear Approaches and Structural Tests*. Springer-Verlag, Heidelberg 2005.
241. D. V. Steward. On an approach to techniques for the analysis of the structure of large systems of equations. *SIAM Review*, 4: 321-342, 1962.

242. A. A. Stoorvogel, H. H. Niemann, A. Saberi and P. Sannuti. Optimal fault signal estimation. *International Journal of Robust and Nonlinear Control*, 12: 697-727, 2002.
243. J. Stoustrup and M. J. Grimble. Integrating control and fault diagnosis: A separation result. *IFAC Sym. on Fault Detection, Supervision and Safety for Technical Processes*, pp. 323-328, Hull 1997.
244. J. Stoustrup, M.J. Grimble and H.H. Niemann. Design of integrated systems for control and detection of actuator/sensor faults. *Sensor Review*, 17: 157-168, 1997.
245. J. Stoustrup and H.H. Niemann. Fault tolerant feedback control using the Youla parameterization. *European Control Conference*, Porto 2001.
246. O. Stursberg, S. Kowalewski and S. Engell. Generating timed discrete models of continuous systems. *Proc. 2nd MATHMOD Conf.*, pp. 203-209, Vienna 1997.
247. M. Tagina, J. -P. Cassar, G. Dauphin-Tanguy and M. Staroswiecki. Bond-graph models for direct generation of formal fault detection systems. *Systems Analysis Modelling and Simulation J.*, 23: 1-17, 1996.
248. R. M. Tallam, T. G. Habetler and R. G. Harley. Transient model for induction machines with stator winding turn faults. *IEEE Transactions on Industry Applications*, 38: 632-637, 2003.
249. R. M. Tallam, T. G. Habetler and R. G. Harley. Stator winding turn-fault detection for closed-loop induction motor drives. *IEEE Transactions on Industry Applications*, 39: 720-724, 2003.
250. T. T. Tay, I. M. Y. Mareels and J. B. Moore. *High Performance Control*. Birkhäuser, 1997.
251. J. Thoma and B. Ould Bouamama. *Modelling and simulation in thermal and chemical engineering: A Bond graph approach*. Springer-Verlag Berlin, 2000.
252. J. U. Thoma. *Modern Oilhydraulic Engineering*. Trade and technical press Ltd, Morden, Surrey 1971.
253. L. Thomas, L. Lambolais, R. Lesiour, C. André, M. Bayart and C. Choukair. Architectural techniques for the description and validation of distributed real-time systems. *2nd IEEE International Symposium on Object oriented Real-Time Distributed Computing (ISORC'99)*, pp. 323-331, 1999.
254. J. S. Thomsen. *A Fault Tolerant Electronic Steering System for a Fork Lift Truck.* Internal report, Aalborg University and Danfoss A/S, 2000.
255. J. S. Thomsen and M. Blanke. Fault-tolerant Electrical Steering for Warehouse Trucks. *IEEE IECON'06: 32nd Annual Conference of the IEEE Industrial Electronics Society*, November 2006 (submitted).
256. C. Thybo. *Fault-Tolerant Control of Inverter Controlled Induction Motors*. PhD Thesis, Aalborg University 2000.
257. C. Thybo and M. Blanke. Industrial cost-benefit assessment for fault-tolerant control systems. *Proc. IEE Conference Control*, Swansea 1998, pp. 1151-1156.
258. J. Unger, A. Kröner and W. Marquardt. Structural analysis of differential-algebraic equation systems – theory and applications. *Computers and Chemical Engineering*, 19: 867-882, 1995.
259. R. J. Veillette, J. V. Medani and W. R. Perkins. Design of reliable control systems. *IEEE Trans.*, AC-37: 290-304, 1992.
260. N. Viswanadham, J. H. Taylor and E. C. Luce. A frequency-domain approach to failure detection and isolation with application to GE-21 turbine engine control system. *Control-Theory and Advanced Technology*, 3: 45-72, 1987.
261. E. Walter and L. Pronzato. *Identification of Parametric Models from Experimental Data*. Springer, 1997.
262. Q. Wang and F. E. Cellier. Time windows: Automated abstraction of continuous-time models into discrete-event models in high autonomy systems. *Int. J. General Systems*, 19: 241-262, 1991.
263. J. Willems. Paradigms and puzzles in the theory of dynamic systems. *IEEE Trans.*, AC-36: 259-294, 1991.

264. W. M. Wonham. A control theory for discrete-event system. In M. J. Denham and A. J. Laub, editors, *Advanced Computing Concepts and Techniques in Control Engineering*, pp. 129-169, Springer-Verlag, 1988.
265. N. E. Wu. Coverage in fault-tolerant control. *Automatica*, 40: 537-548, 2004.
266. N. E. Wu and T. J. Chen. Reliability prediction for self-repairing flight control systems. *Proc. 35th IEEE Conference on Desicion and Control*, Kobe 1996.
267. N. E. Wu and G. J. Klir. Optimal redundancy management in reconfigurable control systems based on normalised nonspecificity. *Int. Journal of Systems Science*, 31: 797-808, 2000.
268. N. E. Wu, K. Zhou and G. Salomon. Reconfigurability in linear time-invariant systems. *Automatica* 36: 1767-1771, 2000.
269. N. E. Wu, Zhang and Zhou. Detection, estimation and accommodation on loss of control effectiveness. *International Journal of Adaptive Control and Signal Processing*, 2000.
270. N. E. Wu, S. Thavamani, Y. M. Zhang and M. Blanke. Sensor fault masking of a ship propulsion system, *Control Engineering Practice*, 2006 (in print).
271. X. Zhang, T. Parisini and M. M. Polycarpou. Adaptive fault-tolerant control of nonlinear uncertain systems: An information-based diagnostic approach. *IEEE Transactions on Automatic Control*, AC-49: 1259-1274, 2004.
272. J. Yamé and M. Kinnaert. Parameterization of linear controllers for bumpless switching in multi-controller schemes. *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Providence, Rhode Island 2004.
273. Z. Yang and J. Stoustrup. Design of robust reconfigurable control for parametric and additive faults. *39th IEEE Conference on Decision and Control*, pp. 4132-4137, Sydney 2000.
274. Z. Yang and M. Blanke. The robust control mixer module method for control reconfiguration. *American Control Conference*, 2000.
275. Z. Yang, R. Izadi-Zamanabadi and M. Blanke. On-line multiple-model based adaptive control reconfiguration for a class of non-linear control systems. *Safeprocess*, Budapest 2000.
276. L. Zaccarian and A.R. Teel. A common framework for anti-windup, bumpless transfer and reliable designs, *Automatica*, 38: 1735-1744, 2002.
277. Q. Zhang, M. Basseville and A. Benveniste. Early warning of slight changes in systems. *Automatica*, 30: 95-113, 1994.
278. Zhang, Q., M. Basseville and A. Benveniste. Fault Detection and Isolation in Nonlinear Dynamic Systems: A Combined Input-Output and Local Approach. *Automatica*, 34: 1359-1373, Benveniste 1998.
279. K. Zhou, J. C. Doyle and K. Glover. *Robust and Optimal Control*. Prentice Hall, 1995.

Appendix 1

Some prerequisites on vectors and matrices

Homogeneous system. Let $\boldsymbol{A}$ be an $m \times n$ matrix, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{x}$ an n vector $\boldsymbol{x} \in \mathbb{R}^n$. The system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{O}$$

is called a *homogeneous system*.

- Every homogeneous system has the solution $\boldsymbol{x} = \mathbf{0}$. This solution is called the trivial solution.
- If the homogeneous system has fewer equations than unknowns, it has an infinite number of solutions; in particular, it has a nontrivial solution.

Vector space. A collection V of n-vectors is a *vector space* if V satisfies the following properties:

1. (*Closure under vector addition*) The sum of two vectors u and v in V is a vector $u + v$ that belongs to V.
2. (*Closure under scalar multiplication*) The product of a vector v in V with any scalar c is a vector cv that belongs to V.

If V satisfies these properties, it is said to be *closed* under vector addition and scalar multiplication.

Rank of a matrix. Let $\boldsymbol{A}$ be an $m \times n$ matrix and let $\tilde{\boldsymbol{A}}$ be its Gauss-reduced form. The rank of $\boldsymbol{A}$ is the number of nonzero rows of $\tilde{\boldsymbol{A}}$. It is equal to the number of linearly independent rows or columns in matrix $\boldsymbol{A}$.

Subspace. A vector space V is a *subspace* of a vector space W if every vector in V also belongs to W.

Let $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ be an $m \times n$ matrix.

Definition. *The subspace of $\mathbb{R}^n$ consisting of all solutions to the homogeneous linear system $\boldsymbol{Ax} = \boldsymbol{0}$ is called the null space of the matrix $\boldsymbol{A}$, and is denoted by $N(\boldsymbol{A})$.*

Definition. *The subspace of $\mathbb{R}^m$ consisting of all vectors $\boldsymbol{b}$ for which the system $\boldsymbol{Ax} = \boldsymbol{b}$ is consistent is called the column space of the matrix $\boldsymbol{A}$, and is denoted by $C(\boldsymbol{A})$.*

Example *Column space*

Let

$$\boldsymbol{A} = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 6 & 9 \end{pmatrix}.$$

The null space of $\boldsymbol{A}$ is the subspace of $\mathbb{R}^3$ consisting of all vectors of the form

$$\boldsymbol{x} = t \begin{pmatrix} 2 \\ -1 \\ 0 \end{pmatrix}' + u \begin{pmatrix} 3 \\ 0 \\ -1 \end{pmatrix}'$$

for real numbers t and u. The column space of $\boldsymbol{A}$ is the subspace of $\mathbb{R}^2$ consisting of all vectors of the form

$$\boldsymbol{b} = v \begin{pmatrix} 1 \\ 3 \end{pmatrix}'$$

for real numbers v. □

Four fundamental subspaces. Let $\boldsymbol{A}$ be an $m \times n$ matrix. The four fundamental subspaces of $\boldsymbol{A}$ are

- the *column space* of $\boldsymbol{A}$, denoted by $C(\boldsymbol{A})$.
- the *nullspace* of $\boldsymbol{A}$, denoted by $N(\boldsymbol{A})$.
- the *row space* of $\boldsymbol{A}$, which is the column space of $\boldsymbol{A}'$ and denoted by $R(\boldsymbol{A})$.
- the *left nullspace* of $\boldsymbol{A}$, which is the nullspace of $\boldsymbol{A}'$. It contains all vectors $\boldsymbol{y}$ such that $\boldsymbol{A}'\boldsymbol{y} = 0$, and is denoted by $N(\boldsymbol{A}')$.

Fundamental theorem of linear algebra. Let $\boldsymbol{A}$ be an $m \times n$ matrix of rank r. Then

- The column space of $\boldsymbol{A}$ has dimension r.
- The nullspace of $\boldsymbol{A}$ has dimension $n - r$.
- The rowspace of $\boldsymbol{A}$ has dimension r.
- The left nullspace of $\boldsymbol{A}$ has dimension $m - r$.

Example *Four fundamental subspaces*

Let

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 6 & 9 \end{bmatrix}.$$

- The (right) null space of $\boldsymbol{A}$ has a basis $\left\{ \begin{pmatrix} 2 \\ -1 \\ 0 \end{pmatrix}', \begin{pmatrix} 3 \\ 0 \\ -1 \end{pmatrix}' \right\}$ It is a subspace of $\mathbb{R}^3$ of dimension 2.
- The column space of $\boldsymbol{A}$ has a basis $\left\{ \begin{pmatrix} 1 \\ 3 \end{pmatrix}' \right\}$. It is a subspace of $\mathbb{R}^2$ of dimension 1.
- The row space of $\boldsymbol{A}$ has a basis $\left\{ \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}' \right\}$. It is a subspace of $\mathbb{R}^3$ of dimension 1.
- The (left) null space of $\boldsymbol{A}$ is the set of solutions to the matrix equation $\boldsymbol{x}'\boldsymbol{A} = \boldsymbol{0}'$. It has a basis $\left\{ \begin{pmatrix} -3 \\ 1 \end{pmatrix}' \right\}$ and is a subspace of $\mathbb{R}^2$ of dimension 1. □

Orthogonal subspaces. Two subspaces V and W of the vector space $\mathbb{R}^n$ are *orthogonal* if every vector $\boldsymbol{v} \in V$ is orthogonal to every vector $\boldsymbol{w} \in W$; that is, $\boldsymbol{v}'\boldsymbol{w} = 0$ for all $\boldsymbol{v}$ and $\boldsymbol{w}$.

Orthogonal complement. Given a subspace V of $\mathbb{R}^n$, the space of all vectors orthogonal to V is called the *orthogonal complement* of V, and is denoted $V^{\perp}$.

Jacobian and other derivatives. Derivatives of functions with respect to vectors are employed when nonlinear systems ae linearised. In the dynamic vector equation

$$\dot{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}), \quad \boldsymbol{x} \in R^n, \boldsymbol{u} \in R^m \tag{A1.1}$$

we introduce $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \bar{\boldsymbol{x}}$ and $\tilde{\boldsymbol{u}} = \boldsymbol{u} - \bar{\boldsymbol{u}}$ where the overbar indicates the value (functional) about which the system is to be to linearised. Taylor expansion of Eq. (A1.1) yields

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

and the first order expansion defines the linearised system,

$$\frac{d\tilde{\boldsymbol{x}}}{dt} = \frac{\partial \boldsymbol{g}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})}{\partial \boldsymbol{x}}\tilde{\boldsymbol{x}} + \frac{\partial \boldsymbol{g}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}})}{\partial \boldsymbol{u}}\tilde{\boldsymbol{u}} = \boldsymbol{A}\tilde{\boldsymbol{x}} + \boldsymbol{B}\tilde{\boldsymbol{u}}.$$

The *Jacobian* $\frac{\partial \boldsymbol{g}(\bar{\boldsymbol{x}},\bar{\boldsymbol{u}})}{\partial \boldsymbol{u}}$ is the $n \times m$ matrix

$$\frac{\partial \boldsymbol{g}(\bar{\boldsymbol{x}},\bar{\boldsymbol{u}})}{\partial \boldsymbol{u}} = \begin{pmatrix} \frac{\partial \boldsymbol{g}_1}{\partial x_1} & \frac{\partial \boldsymbol{g}_1}{\partial x_2} & \cdots & \frac{\partial \boldsymbol{g}_1}{\partial u_m} \\ \frac{\partial \boldsymbol{g}_2}{\partial x_1} & \frac{\partial \boldsymbol{g}_2}{\partial x_2} & \cdots & \frac{\partial \boldsymbol{g}_2}{\partial u_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{g}_n}{\partial x_1} & \frac{\partial \boldsymbol{g}_n}{\partial x_2} & \cdots & \frac{\partial \boldsymbol{g}_n}{\partial u_m} \end{pmatrix} \tag{A1.2}$$

of partial derivatives of the entries of g.

Other useful derivatives with respect to a vector v are

$$\begin{aligned} \frac{d}{dv}(\boldsymbol{A}\boldsymbol{v}) &= \boldsymbol{A} \\ \frac{d}{dv}(\boldsymbol{v}'\boldsymbol{A}) &= \boldsymbol{A}' \\ \frac{d}{dv}(\boldsymbol{v}'\boldsymbol{A}\boldsymbol{v}) &= \boldsymbol{v}'\left(\boldsymbol{A}+\boldsymbol{A}'\right). \end{aligned}$$

Appendix 2

Notions of probability theory

Introduction. In this appendix, different notions of probability theory that are used or mentioned in chapter 6 are briefly reviewed. Gaussian and χ^2-distributed random variables are first presented. Next a simple hypothesis testing problem is stated and solved by the so-called χ^2 test. The statistics of the empirical mean is analysed in the subsequent section. Finally, the last part of the appendix, which is also the main one, is presenting a review of notions on continuous and discrete-time random processes.

Gaussian and χ^2-distributed random variables. A normally distributed random variable x with mean μ and variance σ^2 is characterised by its probability density function, $p(x)$ which has the form

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{A2.1}$$

To indicate that the probability law of x, $\mathcal{L}(x)$, is the Gaussian law with mean μ and variance σ^2, one uses the notation $\mathcal{L}(x) = \mathcal{N}(\mu, \sigma^2)$.

The probability density function of a n-dimensional Gaussian random vector $\boldsymbol{x}$ with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{Q}$ has the form

$$p(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^n \det \boldsymbol{Q}}} \exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{\mu})' \boldsymbol{Q}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})}{2}\right) \tag{A2.2}$$

Remark. *Comment on notations*
The symbol x ($\boldsymbol{x}$) has two different meanings above. It denotes either a random variable (vector) or the argument of the probability density function associated to the considered random variable (vector). The distinction should be clear from the context. □

Consider now a set of independent normally distributed random variables x_i, $i = 1, ..., d$, each of them with zero mean and unit variance ($\mathcal{L}(x_i) = \mathcal{N}(0,1), i = 1, \ldots, d$). Then the distribution of the random variable $\xi = \sum_{i=1}^{d} x_i^2$ is denoted by $\chi^2(d)$, and it is called a chi-square distribution with d degrees of freedom. Its mean is equal to d, and its variance is $2d$.

If $\mathcal{L}(x_i) = \mathcal{N}(\mu_i, 1), i = 1, \ldots, d$, the distribution of $\xi = \sum_{i=1}^{d} x_i^2$ is denoted $\chi^2(d, \lambda)$. It is now a chi-square distribution with d degrees of freedom and with non-centrality parameter $\lambda = \sum_{i=1}^{d} \mu_i^2$. Its mean is equal to $d + \lambda$, and its variance is $2d + 4\lambda$.

With these definitions in mind, the basic hypothesis testing problem can now be stated and deliverd to the χ^2-test.

Basic hypothesis testing problem and χ^2-test. Consider the following problem:

Problem. Basic hypothesis testing *Given $\boldsymbol{x}_m$, a realisation of a normally distributed random vector $\boldsymbol{x}$ of dimension n, with variance $\boldsymbol{Q}$,*
determine whether $\boldsymbol{x}_m$ is a realisation of a random vector $\boldsymbol{x}$ with zero mean or non-zero mean, namely choose between the following two hypotheses:

$\mathcal{H}_0$*:* $\mathcal{L}(\boldsymbol{x}) = \mathcal{N}(0, \boldsymbol{Q})$
$\mathcal{H}_1$*:* $\mathcal{L}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{Q})$ *where $\boldsymbol{\mu}$ is an unknown non-zero vector*

The index m used above stands for "measured".

To solve the problem, it suffices to realise that the random variable $\xi = \boldsymbol{x}'\boldsymbol{Q}^{-1}\boldsymbol{x}$ has a χ^2 distribution under hypothesis $\mathcal{H}_0$. Indeed, since $\boldsymbol{Q}$ is positive definite, it can be factorised as $\boldsymbol{Q} = \tilde{\boldsymbol{Q}}\tilde{\boldsymbol{Q}}'$ by Choleski factorisation. Set $\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{Q}}^{-1}\boldsymbol{x}$, and consider the variance of $\tilde{\boldsymbol{x}}$ under hypothesis $\mathcal{H}_0$:

$$E(\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}') = \tilde{\boldsymbol{Q}}^{-1}E(\boldsymbol{x}\boldsymbol{x}')\tilde{\boldsymbol{Q}}^{-T} = \tilde{\boldsymbol{Q}}^{-1}\tilde{\boldsymbol{Q}}\tilde{\boldsymbol{Q}}'\tilde{\boldsymbol{Q}}^{-T} = \boldsymbol{I}_n,$$

where the superscript $-T$ stands for inverse transpose. As the variance of $\tilde{x}$ is the identity matrix, its components, $\tilde{\boldsymbol{x}}_i, i = 1, \ldots, n$ are independent and $\mathcal{L}(\tilde{\boldsymbol{x}}_i) = \mathcal{N}(0,1)$ under hypothesis $\mathcal{H}_0$. Finally, noticing that

$$\xi = \boldsymbol{x}'\boldsymbol{Q}^{-1}\boldsymbol{x} = \tilde{\boldsymbol{x}}'\tilde{\boldsymbol{x}} = \sum_{i=1}^{n_z} \tilde{x}_i^2,$$

one deduces that ξ indeed has a $\chi^2(n)$ distribution under hypothesis $\mathcal{H}_0$.

Thus a standard χ^2-test can be used to check whether $\xi_m = \boldsymbol{x}'_m\boldsymbol{Q}^{-1}\boldsymbol{x}_m$ is a realisation of a $\chi^2(n)$ random variable. This test relies on the χ^2 statistical table

that provides, for a given probability of false alarm, α (namely the probability of choosing $\mathcal{H}_1$ while $\mathcal{H}_0$ is true), a threshold h such that:

$$\text{Prob}(\xi \leq h) = 1 - \alpha, \tag{A2.3}$$

where $\text{Prob}(\xi \leq h)$ denotes the probability of the event $\xi \leq h$.

The so-called χ^2-test then simply amounts to the following operations:

Algorithm A2.2 *χ^2-Test*

Given:

1. $\boldsymbol{x}_m$, a realisation of the n-dimensional normally distributed random vector $\boldsymbol{x}$ with variance $\boldsymbol{Q}$,
2. α, a probability of false alarm.

Determine:

1. The threshold h that fulfils (A2.3) from the χ^2 statistical table.
2. $\xi_m = \boldsymbol{x}'_m \boldsymbol{Q}^{-1} \boldsymbol{x}_m$.

Output: Accept $\mathcal{H}_0$ if $\xi_m \leq h$.
Accept $\mathcal{H}_1$ otherwise.

Statistics of the empirical mean. Consider a data sample $\{x_{m,1}, \ldots, x_{m,n}\}$ where the $x_{m,i}, i = 1, \ldots, n$ are realisations of the random variables x_i. The latter are assumed to be independent and identically distributed. Let $m = E(x_i)$ and $\sigma^2 = E((x_i - m)^2)$ be respectively the mean and the variance of x_i ($E(.)$ denotes the expectation of the considered random variable). The empirical mean of the data sample is defined as

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i.$$

The mean and the variance of $\bar{x}$ are respectively given by

$$\begin{aligned} E(\bar{x}) &= m \\ E((\bar{x} - m)^2) &= \frac{\sigma^2}{n}. \end{aligned}$$

In the sequel, the particular case where the distribution of the x_i's is Gaussian is considered, namely $\mathcal{L}(x_i) = \mathcal{N}(m, \sigma^2)$. This implies $\mathcal{L}(\bar{x}) = \mathcal{N}(m, \frac{\sigma^2}{n})$.

To be able to characterise the quality of the estimate of the mean provided by the empirical mean one resorts to the following result. Define $T_{n-1} = \frac{\bar{x} - m}{q} \sqrt{n-1}$,

where $q^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$ is the empirical variance of the sample. Then it can be proven that T_{n-1} is a Student random variable with $n-1$ degrees of freedom. From this fact, the confidence region for the mean can be characterised by

$$\bar{x}_m - t_{\alpha/2}\frac{q_m}{\sqrt{n-1}} < m < \bar{x}_m + t_{\alpha/2}\frac{q_m}{\sqrt{n-1}}, \tag{A2.4}$$

where

- $\bar{x}_m$ and q_m are realisations of the random variables $\bar{x}$ and q respectively, namely $\bar{x}_m = \frac{1}{n}\sum_{i=1}^{n} x_{m,i}$ and $q_m^2 = \frac{1}{n}\sum_{i=1}^{n}(x_{m,i} - \bar{x}_m)^2$
- $t_{\alpha/2}$ is such that $\int_{-t_{\alpha/2}}^{t_{\alpha/2}} f_T(x)dx = \alpha$ with
  - $f_T(x)$: the probability density function associated to a Student random variable with $n-1$ degrees of freedom
  - α: the probability that m lies in the considered interval (A2.4)

Remark. *Notation conventions*
A different notation has been used above to denote a random variable or vector and a specific realisation of this variable or vector. In most of the other sections, this distinction is not indicated explicitly; the considered object should be clear from the context. □

Stochastic processes. A stochastic (or random) process is defined as a mapping that associates to each time instant t in a set $\mathcal{T}$ a random variable $x(t)$ (or a random vector in the case of a vector random process). A random process thus appears a an infinite set of random variables. One can distinguish continuous and discrete-time stochastic processes according as the variable t belongs to a continuum of values or the set $\mathcal{T}$ is made of the sampling instants. In the latter case, $\mathcal{T} = \{..., -T, 0, T, 2T, ...\}$ or $\mathcal{T} = \{..., -1, 0, 1, 2, ...\}$ when the sampling period T is chosen as the time unit. Discrete-time stochastic processes are also called random on stochastic sequences.

The notion of random process is used to model quantities for which there is no way to predict an exact value at a future instant of time. This is typically the case for measurement noise for instance. The outcome of an experiment generates a specific outcome or realisation of a random process, which is a function of time.

Distribution function and probability density function. A random process is completely characterised by its n^{th}-order distribution function (also called cumulative distribution function), for an arbitrary n, defined as

$$F(x_1, t_1; \cdots; x_n, t_n) = \mathrm{Prob}(x(t_1) \leq x_1, \cdots, x(t_n) \leq x_n),$$
$$t_i \neq t_j,\ x_1, \cdots, x_n \in \mathbb{R}, \quad t_1, \cdots, t_n \in \mathbb{R} \text{ or } Z,$$

where

$$\mathrm{Prob}(x(t_1) \leq x_1, \cdots, x(t_n) \leq x_n)$$

is the probability that the events $(x(t_1) \leq x_1), \cdots, (x(t_n) \leq x_n)$ are observed in a realisation of $x(t)$.

In particular, the first-order (cumulative) distribution function is defined as $F(x,t) = \mathrm{Prob}(x(t) \leq x)$. It is thus the probability of the event $(x(t) \leq x)$.

The probability density function of the random process $x(t)$ is defined as the derivative of the first-order cumulative distribution function with respect to x: $p(x,t) = \frac{\partial F(x,t)}{\partial x}$ and the n^{th} order probability density function is:

$$p(x_1, t_1; \cdots; x_n, t_n) = \frac{\partial^n F(x_1, t_1; \cdots; x_n, t_n)}{\partial x_1 \cdots \partial x_n}$$

Moments of stochastic processes. The properties of a random process $x(t)$ can be fully characterised by its moments defined, for the n^{th}-order moment as

$$M_x^n(t) = \int_{-\infty}^{\infty} x^n p(x,t) dx.$$

The first-order moment of a random process $x(t)$ is called its mean or its expected value,

$$\mu_x(t) = \int_{-\infty}^{\infty} x p(x,t) dx.$$

This operation is often denoted $\mu_x(t) = E(x(t))$ where "E" stands for the expectation operator. In the vector case, $\boldsymbol{\mu}_x(t)$ is a vector with the same dimension as $\boldsymbol{x}(t)$ of which the i^{th} entry is obtained by computing the above integral for the i^{th} component of $\boldsymbol{x}(t)$. In that case, $p(\boldsymbol{x}, t)$ is a scalar function of vector $\boldsymbol{x}$ and time t (see for instance the probability density function of a Gaussian random process in (A2.6) below).

Similarly, the mean square is defined as the second order moment

$$ms_x(t) = \int_{-\infty}^{\infty} x^2 p(x,t) dx = E(x^2(t)).$$

The variance of the stochastic process $x(t)$ is the centered second order moment of $x(t)$ ("centered" indicates that the mean is subtracted from $x(t)$),

$$\sigma_x^2(t) = \int_{-\infty}^{\infty} (x - \mu_x(t))^2 p(x,t) dx = E((x(t) - \mu_x(t))^2).$$

For a vector stochastic process, this expression takes the form

$$\begin{aligned} \boldsymbol{Q}_x(t) &= \int_{-\infty}^{\infty} (\boldsymbol{x}(t) - \boldsymbol{\mu}_x(t))(\boldsymbol{x} - \boldsymbol{\mu}_x(t))' p(\boldsymbol{x}, t) d\boldsymbol{x} \\ &= E((\boldsymbol{x}(t) - \boldsymbol{\mu}_x(t))(\boldsymbol{x}(t) - \boldsymbol{\mu}_x(t))'). \end{aligned}$$

$\boldsymbol{Q}_x(t)$ is thus a matrix of which entry (i,j) is given by

$$\int_{-\infty}^{\infty} (x_i - \mu_{x,i})(x_j - \mu_{x,j}) p(\boldsymbol{x}, t) d\boldsymbol{x}.$$

The integral is actually a multiple integral; integration is performed over each component of $\boldsymbol{x}$.

The autocorrelation function of a random process $\boldsymbol{x}(t)$ is the joint moment of the random vectors $\boldsymbol{x}(t_1)$ and $\boldsymbol{x}(t_2)$,

$$\boldsymbol{R}_{xx}(t_1,t_2) = E(\boldsymbol{x}(t_1)\boldsymbol{x}(t_2)') = \int_{-\infty}^{\infty} \boldsymbol{x}_1\boldsymbol{x}_2' p(\boldsymbol{x}_1,t_1;\boldsymbol{x}_2,t_2)d\boldsymbol{x}_1 d\boldsymbol{x}_2$$

and the (auto)covariance of $\boldsymbol{x}(t)$ is given as

$$\boldsymbol{C}_{xx}(t_1,t_2) = E((\boldsymbol{x}(t_1)-\boldsymbol{\mu}(t_1))(\boldsymbol{x}(t_2)-\boldsymbol{\mu}(t_2))').$$

This generalises to the cross-covariance of two random processes $\boldsymbol{x}(t)$ and $\boldsymbol{y}(t)$ defined as

$$\boldsymbol{C}_{xy}(t_1,t_2) = E((\boldsymbol{x}(t_1)-\boldsymbol{\mu}_x(t_1))(\boldsymbol{y}(t_2)-\boldsymbol{\mu}_y(t_2))).$$

Most often in engineering applications, one resorts to the first and second order moments of the stochastic processes; however these are not sufficient to fully characterise a random process, except if this process is normally distributed. In the latter case, $p(x,t)$ corresponds to the well known Gaussian distribution with mean μ_x and variance σ_x^2 (or $\boldsymbol{Q}_x$ in the vector case),

$$p(x,t) = \frac{1}{\sigma_x(t)\sqrt{2\pi}} \exp\left(-\frac{(x-\mu_x(t))^2}{2\sigma_x(t)^2}\right), \tag{A2.5}$$

and for an n-dimensional Gaussian random process,

$$p(\boldsymbol{x},t) = \frac{1}{\sqrt{(2\pi)^n \det \boldsymbol{Q}_x(t)}} \exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{\mu}_x(t))'\boldsymbol{Q}_x(t)^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_x(t))}{2}\right) \tag{A2.6}$$

Remark. *Discrete-time random process*
The above definitions are valid for both continuous-time and discrete-time stochastic processes. Below, when specific expressions must be considered for discrete-time processes, the superscript $''d''$ will be used to indicate explicitly that moments or other quantities are associated to discrete-time random processes. □

Stationary random process. A stochastic process is said to be stationary (in the strict sense) if its probability density functions of any order are not affected by a shift in the time origin, namely

$$p(x_1,t_1;x_2,t_2;\cdots;x_n,t_n) = p(x_1,t_1+\epsilon;\cdots;x_n,t_n+\epsilon)$$

for any real ϵ and any integer n.

On the other hand, a weakly stationary random process has the following properties:

- its mean does not change with time.
- its covariance $C(t_1,t_2)$ depends on t_1 and t_2 only through the difference t_2-t_1. Thus $C(t_1,t_2) = C(t_2-t_1) = E((x(t+t_2-t_1)-\mu_x)(x(t)-\mu_x))$.

A random process with the above two properties is also sometimes said to be stationary in the wide sense.

Stationarity in the strict sense implies weak stationarity, but the converse is not true, except if $x(t)$ has a Gaussian distribution.

Clearly the probability density function of a Gaussian weakly stationary random process is given by (A2.5) or (A2.6) in which the argument t is cancelled.

Empirical mean, variance and covariance of a stochastic process. Often in practice, only one or a few outcomes of an experiment are available, and one has to infer from such data a model of the observed phenomenon. In particular, one has to deduce the properties of measurement noise from a realisation of this noise. A classical approach to do this is to estimate the moments of the corresponding random process by time averages. This amounts to assuming that time averages are equal to ensemble averages (i.e. expected values); it is the so-called ergodic hypothesis. Roughly speaking, this hypothesis holds true for a stochastic process $x(t)$ if, as τ increases, the random variables (or vectors) $x(t)$ and $x(t+\tau)$ become uncorrelated. This situation is quite general in practice.

For a weakly stationary ergodic continuous-time stochastic process, the following equalities hold

$$\begin{aligned}
\boldsymbol{\mu}_x &= E(\boldsymbol{x}(t)) = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{'} \boldsymbol{x}(t)dt \\
\boldsymbol{R}_{xx}(\tau) &= E(\boldsymbol{x}(t+\tau)\boldsymbol{x}(t)') = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{'} \boldsymbol{x}(t+\tau)\boldsymbol{x}(t)'dt \\
\boldsymbol{C}_{xx}(\tau) &= E((\boldsymbol{x}(t+\tau)-\boldsymbol{\mu}_x)(\boldsymbol{x}(t)-\boldsymbol{\mu}_x)') \\
&= \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{'} (\boldsymbol{x}(t+\tau)-\boldsymbol{\mu}_x)(\boldsymbol{x}(t)-\boldsymbol{\mu}_x)'dt.
\end{aligned}$$

In practice, only a finite number, say N, of samples of a realisation of a stochastic process is available. The integral in the above expressions can be estimated from the data set $\{\boldsymbol{x}(1), \cdots, \boldsymbol{x}(N)\}$ as

$$\hat{\boldsymbol{\mu}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}(i)$$

$$\hat{Q} = \hat{C}(0) = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{x}(i)-\hat{\boldsymbol{\mu}})(\boldsymbol{x}(i)-\hat{\boldsymbol{\mu}})'.$$

Spectral density or power spectrum. In classical control engineering, transfer functions are extremely useful computational and analysis tools. To exploit them in stochastic control theory, the notion of power spectrum (or spectral density) is introduced. By definition, the power spectrum of the continuous-time (discrete-time)

stochastic process $x(t)$ is the Fourier transform (discrete Fourier transform) of its autocovariance function,

$$S_{xx}(\omega) = \int_{-\infty}^{\infty} C_{xx}(\tau) \exp(-j\omega\tau) d\tau \tag{A2.7}$$

($S^d_{xx}(\omega) = \sum_{n=-\infty}^{\infty} C^d_{xx}(n) \exp(-j\omega n)$).
For a random vector, $\boldsymbol{S}_{xx}(\omega)$ is a matrix of which each entry is the Fourier transform of the corresponding entry in $\boldsymbol{C}_{xx}$.

The name "spectral density" used for $S_{xx}(\omega)$ is due to the fact that the variance of the stochastic process can be recovered through

$$\sigma_x^2 = \frac{1}{2\pi} \int_{-\infty}^{\infty} S_{xx}(\omega) d\omega \tag{A2.8}$$

($\sigma_x^{d2} = \frac{1}{2\pi} \int_{-\pi}^{\pi} S^d_{xx}(\omega) d\omega$).
(A2.8) is obtained by taking the inverse Fourier transform of (A2.7) and setting the time shift equal to zero in the result.

Example *Computation of spectral density function*

The autocovariance function of a random process $v(t)$ is given by

$$C_{vv}(\tau) = \sigma^2 e^{(-\beta|\tau|)},$$

where σ^2 and β are known variables. The spectral density function for the $v(t)$ process is

$$S_{vv}(\omega) = \frac{2\sigma^2\beta}{\omega^2 + \beta^2}$$

Evaluation of (A2.8) yields

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{2\sigma^2\beta}{\omega^2 + \beta^2} d\omega = \frac{\sigma^2\beta}{\pi} [\frac{1}{\beta} \tan^{-1}(\frac{\omega}{\beta})]_{-\infty}^{\infty} = \sigma^2, \tag{A2.9}$$

as expected since $C_{vv}(0) = \sigma^2$. □

Example *Band-limited noise through low-pass filter*

Given a stable low-pass filter with the transfer function

$$H(s) = \frac{\alpha}{s + \alpha} \tag{A2.10}$$

with input $w(t)$, which is a band-limited random signal with autocorrelation function

$$R_{ww}(\tau) = \sigma_w^2 e^{-\beta|\tau|}.$$

The output of the filter is $y(t)$ whose covariance σ_y^2 should be determined. Stability implies $\alpha > 0$ and $\beta > 0$

Applying Eq. (A2.8)

$$\begin{aligned}
\sigma_y^2 &= \frac{1}{2\pi}\int_{-\infty}^{\infty}\sigma_w^2\frac{2\beta}{\omega^2+\beta^2}\frac{\alpha^2}{\omega^2+\alpha^2}d\omega \;\Leftrightarrow \\
\sigma_y^2 &= \frac{\sigma_w^2 2\alpha^2}{(\beta^2-\alpha^2)}\frac{1}{2\pi}\left(\int_{-\infty}^{\infty}\frac{\beta}{(\omega^2+\alpha^2)}d\omega-\int_{-\infty}^{\infty}\frac{\beta}{(\omega^2+\beta^2)}d\omega\right) \;\Leftrightarrow \\
\sigma_y^2 &= \frac{2}{\pi}\alpha^2\frac{\sigma_w^2}{\beta^2-\alpha^2}\left(\frac{1}{2}\pi\frac{\beta}{\alpha}-\frac{1}{2}\pi\right) \;\Leftrightarrow \\
\sigma_y^2 &= \sigma_w^2\frac{(\beta-\alpha)\,\alpha}{\beta^2-\alpha^2} \;\Rightarrow\; \sigma_y^2 = \frac{\alpha}{\alpha+\beta}\,\sigma_w^2 \quad \square
\end{aligned} \tag{A2.11}$$

Continuous and discrete-time white noise processes. A scalar continuous-time white noise process $x(t)$ is a weakly stationary continuous-time random process with autocovariance function

$$C_{xx}(\tau) = E((x(t+\tau)-\mu)(x(t)-\mu)) = s\delta(\tau), \tag{A2.12}$$

where "s" is a real constant called intensity and $\delta(\tau)$ is the Dirac impulse. For a vector white noise process, (A2.12) takes the form

$$\boldsymbol{C}_{xx}(\tau) = E((\boldsymbol{x}(t+\tau)-\boldsymbol{\mu})(\boldsymbol{x}(t)-\boldsymbol{\mu})') = \boldsymbol{S}\delta(\tau), \tag{A2.13}$$

where the intensity $\boldsymbol{S}$ is a semi-positive definite matrix.

For a white noise process, the spectral density function is constant:

$$S_{xx}(\omega) = s \quad \text{or} \quad \boldsymbol{S}_{xx}(\omega) = \boldsymbol{S}.$$

A white noise process is clearly an abstraction, as it corresponds to a process with infinite variance. It can be seen as the limiting case when β tends to infinity in the given example, which means that no correlation exists between successive time instants. The term "white noise" comes from the analogy with the spectral properties of white light; all frequencies are present with the same intensity in the signal.

A white noise sequence is defined in a similar way as its continuous-time counterpart. It is a weakly stationary discrete-time process with autocovariance given by

$$C_{xx}^d(\tau) = \begin{cases} \sigma^2 & \tau = 0 \\ 0 & \tau = \pm 1, \pm 2, \cdots \end{cases} \tag{A2.14}$$

in the scalar case. The associated spectral density writes $S_{xx}^d(\omega) = \sigma^2$.

A band-limited white noise process is a random process whose spectral density is constant over a finite range of frequencies, and zero outside this range. A noise source that has a constant spectral density down to zero frequency, and a bandwidth of B [rad/s] is defined by

$$S_{vv}(\omega) = \begin{cases} \alpha & |\omega| \le B \\ 0 & |\omega| > B \end{cases} \tag{A2.15}$$

Such a stochastic process can be approximated as the output of a linear dynamical system with a white noise input and with cut-off frequency $B/2\pi$ $[s^{-1}]$. This is the subject of the next section.

Filtered weakly stationary process and filtered white noise process. Consider a linear time-invariant system described by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{w}(t) \tag{A2.16}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{v}(t), \tag{A2.17}$$

where $\boldsymbol{w}(t)$ and $\boldsymbol{v}(t)$ are mutually uncorrelated weakly stationary random processes with mean and power spectral density $\boldsymbol{m}_w, \boldsymbol{S}_w(\omega)$ and $\boldsymbol{m}_v, \boldsymbol{S}_v(\omega)$ respectively.

Our aim is first to compute the power spectral density of $\boldsymbol{x}(t)$ and $\boldsymbol{y}(t)$. To this end, let us introduce the following transfer functions: $\boldsymbol{H}_{yw}(s) = \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B}$ and $\boldsymbol{H}_{xw}(s) = (s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B}$. When system (A2.16), (A2.17) is asymptotically stable, $\boldsymbol{x}(t)$ and $\boldsymbol{y}(t)$ are stationary stochastic processes and their spectral density is respectively

$$\begin{aligned} \boldsymbol{S}_x(\omega) &= \boldsymbol{H}_{xw}(j\omega)S_w(\omega)\boldsymbol{H}_{xw}(-j\omega)' \\ \boldsymbol{S}_y(\omega) &= \boldsymbol{H}_{yw}(j\omega)S_w(\omega)\boldsymbol{H}_{yw}(-j\omega)' + \boldsymbol{S}_v(\omega). \end{aligned} \tag{A2.18}$$

The mean of $\boldsymbol{y}(t)$, $\boldsymbol{m}_y$ is computed from $\boldsymbol{m}_y = \boldsymbol{H}_{yw}(0)\boldsymbol{m}_w + \boldsymbol{m}_v$.

When $\boldsymbol{w}(t)$ and $\boldsymbol{v}(t)$ are zero-mean white noise processes with intensities $\boldsymbol{S}_w$ and $\boldsymbol{S}_v$, one can compute the mean and variance of $\boldsymbol{x}(t)$ ($\boldsymbol{m}_x(t)$ and $\boldsymbol{Q}_x(t)$) directly from the state-space model. Let $\boldsymbol{m}_0$ and $\boldsymbol{Q}_0$ denote the mean and variance of the initial state, then $\boldsymbol{m}_x(t)$ and $\boldsymbol{Q}_x(t)$ are governed by the following differential equations:

$$\frac{d\boldsymbol{m}_x(t)}{dt} = \boldsymbol{A}\boldsymbol{m}_x(t), \quad \boldsymbol{m}_x(0) = \boldsymbol{m}_0 \tag{A2.19}$$

$$\frac{d\boldsymbol{Q}_x(t)}{dt} = \boldsymbol{A}\boldsymbol{Q}_x(t) + \boldsymbol{Q}_x(t)\boldsymbol{A}' + \boldsymbol{B}\boldsymbol{S}_w\boldsymbol{B}', \quad \boldsymbol{Q}_x(0) = \boldsymbol{Q}_0 \tag{A2.20}$$

The variance of $\boldsymbol{y}(t)$ is deduced from (A2.17), namely

$$\boldsymbol{Q}_y(t) = \boldsymbol{C}\boldsymbol{Q}_x(t)\boldsymbol{C}' + \boldsymbol{S}_v. \tag{A2.21}$$

If moreover the linear time-invariant system (A2.16), (A2.17) is asymptotically stable, $\boldsymbol{x}(t)$ tends to a weakly stationary stochastic process with zero mean and variance $\bar{\boldsymbol{Q}}_x$ given as the solution of the following algebraic Lyapunov equation

$$\boldsymbol{A}\bar{\boldsymbol{Q}}_x + \bar{\boldsymbol{Q}}_x\boldsymbol{A}' + \boldsymbol{B}\boldsymbol{S}_w\boldsymbol{B}' = \boldsymbol{O}. \tag{A2.22}$$

Example *Covariance calculation - continuous time case*

A stationary random signal $w(t)$ has the autocovariance function $R_{ww}(\tau) = \sigma_w^2 e^{-\beta|\tau|}$ We wish to calculate the variance σ_w^2.

The signal $w(t)$ is generated by a dynamic (low pass) filter,

$$\dot{w}(t) = -\beta w(t) + \sqrt{2\beta}\sigma_w v(t),$$

where $v(t)$ is a white noise with intensity 1. The variance of $w(t)$ is obtained from the Lyapunov equation,

$$\begin{aligned} 0 &= -\beta Q - Q\beta + \sqrt{2\beta}\sigma_w\sigma_w\sqrt{2\beta} \\ &\Rightarrow \quad -2\beta Q + 2\beta\sigma_w^2 = 0 \\ &\Rightarrow \quad Q = \sigma_w^2 \end{aligned}$$

which is the same result as was obtained by the frequency domain calculation (A2.9). □

Example *Covariance calculation - high-pass filter*

A stationary random signal $w(t)$ with the autocorrelation function $R_{ww}(\tau) = \sigma_w^2 e^{-\beta|\tau|}$ is passed through a stable high-pass filter

$$H(s) = \frac{s}{s+\alpha}$$

with the equivalent state-space representation

$$\begin{aligned} \dot{x}(t) &= -\alpha x(t) + \alpha w(t) \\ y(t) &= w(t) - x(t). \end{aligned}$$

Hence, the filter acting on v(t) is

$$\begin{aligned} \frac{d}{dt}\begin{pmatrix} w(t) \\ x(t) \end{pmatrix} &= \begin{pmatrix} -\beta & 0 \\ \alpha & -\alpha \end{pmatrix}\begin{pmatrix} w(t) \\ x(t) \end{pmatrix} + \begin{pmatrix} \sigma_w\sqrt{2\beta} \\ 0 \end{pmatrix} v(t) \\ y(t) &= \begin{pmatrix} 1 & -1 \end{pmatrix}\begin{pmatrix} w(t) \\ x(t) \end{pmatrix} \end{aligned}$$

The covariance is symmetric,

$$\boldsymbol{Q} = \begin{pmatrix} \sigma_w^2 & \sigma_{wx}^2 \\ \sigma_{xw}^2 & \sigma_x^2 \end{pmatrix} = \begin{pmatrix} a & c \\ c & b \end{pmatrix}$$

and with $\mathbf{S}_v = 1$, the Lyapunov equation gives

$$\begin{pmatrix} -\beta & 0 \\ \alpha & -\alpha \end{pmatrix}\begin{pmatrix} a & c \\ c & b \end{pmatrix} + \begin{pmatrix} a & c \\ c & b \end{pmatrix}\begin{pmatrix} -\beta & \alpha \\ 0 & -\alpha \end{pmatrix} + \\ + \begin{pmatrix} \sigma_w\sqrt{2\beta} \\ 0 \end{pmatrix}\begin{pmatrix} \sqrt{\beta}\sigma_w\sqrt{2} & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

equivalent to the three conditions

$$\begin{aligned} -2a\beta + 2\sigma_w^2\beta &= 0 \\ -2b\alpha + 2c\alpha &= 0 \\ a\alpha - c(\alpha+\beta) &= 0 \end{aligned}$$

which gives

$$\begin{aligned} -2a\beta + 2\sigma_w^2\beta = 0 \quad &\Rightarrow \quad a = \sigma_w^2 \\ a\alpha - c(\alpha+\beta) = 0 \quad &\Rightarrow \quad c = b\frac{\alpha}{\alpha+\beta} \\ -2b\alpha + 2c\alpha = 0 \quad &\Rightarrow \quad b = c. \end{aligned}$$

Hence

$$Q = \begin{pmatrix} 1 & \frac{\alpha}{\alpha+\beta} \\ \frac{\alpha}{\alpha+\beta} & \frac{\alpha}{\alpha+\beta} \end{pmatrix} \sigma_w^2$$

and

$$\sigma_y^2 = CQC' = \frac{\alpha}{\alpha+\beta}\sigma_w^2 \tag{A2.23}$$

which is the desired result. □

Similar results can be obtained for the analysis of a discrete-time system subject to a white noise input sequence. More specifically, consider a discrete-time random process defined as the output of the following discrete-time linear time-invariant system

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{w}(k) \tag{A2.24}$$

$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{v}(k), \tag{A2.25}$$

where $\boldsymbol{w}(k)$ and $\boldsymbol{v}(k)$ are mutually uncorrelated white noise sequences with variance $\boldsymbol{Q}_w$ and $\boldsymbol{Q}_v$.

Let the mean and variance of the initial state of system (A2.24), (A2.25) be $\boldsymbol{m}_0$ and $\boldsymbol{Q}_0$ respectively. The mean value function of the random sequence $\boldsymbol{x}(k)$ is then given by

$$\boldsymbol{m}_x^d(k+1) = \boldsymbol{A}\boldsymbol{m}_x^d(k), \quad \boldsymbol{m}_x^d(0) = \boldsymbol{m}_0$$

and its variance is the solution of the following discrete Lyapunov equation

$$\boldsymbol{Q}_x^d(k+1) = \boldsymbol{A}\boldsymbol{Q}_x^d(k)\boldsymbol{A}' + \boldsymbol{B}\boldsymbol{Q}_w\boldsymbol{B}', \quad \boldsymbol{Q}_x^d(0) = \boldsymbol{Q}_0$$

If the linear time-invariant system is asymptotically stable, the random sequence $\boldsymbol{x}(k)$ tends to a stationary process with zero mean and with variance $\bar{\boldsymbol{Q}}_x^d$ given as the solution of the following discrete algebraic Lyapunov equation $\bar{\boldsymbol{Q}}_x^d = \boldsymbol{A}\bar{\boldsymbol{Q}}_x^d\boldsymbol{A}' + \boldsymbol{Q}_w$. The variance of the corresponding output is computed from

$$\bar{\boldsymbol{Q}}_y^d = \boldsymbol{C}\bar{\boldsymbol{Q}}_x^d\boldsymbol{C}' + \boldsymbol{Q}_v.$$

Generation of coloured noise. In the previous section, one was given a system with white noise inputs, and one had to compute the spectral density of its output. Quite often, the reverse problem is encountered; one has to generate a coloured noise with a given rational spectral density. For the sake of simplicity, only the case of a scalar stochastic process is considered. Such a coloured noise can be obtained by entering a white noise process into an appropriate stable filter as indicated in Fig. A2.1.

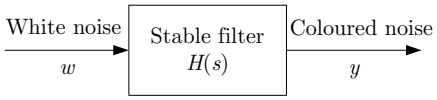


Fig. A2.1. Coloured noise generated by a filtered white noise

The computation of the filter is based on relation (A2.18) with $S_v(\omega) = 0$ and $S_w(\omega) = 1$,

$$S_y(\omega) = H_{yw}(j\omega)H_{yw}(-j\omega). \tag{A2.26}$$

Introducing $s = j\omega$, the right hand side of (A2.26) can be written:

$$F_{yw}(s) = H_{yw}(s)H_{yw}(-s) \tag{A2.27}$$

When z_i (p_i) is a zero (pole) of $H_{yw}(s)$, $-z_i$ ($-p_i$) is a zero (pole) of $H_{yw}(-s)$. Besides, every complex pole appears with its complex conjugate in $F_{yw}(s)$. The poles and zeros of $F_{yw}(s)$ are thus symmetric with respect to the imaginary axis. Letting $z_i^-, i = 1, \ldots, m$ and $p_i^-, i = 1, \ldots, n$ denote the zeros and poles of $F_{yw}(s)$ with negative real part, one can obtain the filter transfer function $H(s)$ as

$$H(s) = k\frac{\prod_{i=1}^{m}(s - z_i^-)}{\prod_{i=1}^{n}(s - p_i^-)},$$

where k is a gain chosen so that $S_y(0)$ has the desired value.

Sampling a linear stochastic differential equation. Consider the differential equation

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{w}(t), \tag{A2.28}$$

where $\boldsymbol{w}(t)$ is a zero mean white noise process with intensity $\boldsymbol{S}$. As seen above, the covariance of $\boldsymbol{w}(t)$ is $E(\boldsymbol{w}(t_1)\boldsymbol{w}'(t_2)) = \boldsymbol{S}\delta(t_1 - t_2)$, and the variance is thus infinite. To avoid mathematical difficulties, a stochastic process that is formally the integral of a white noise process is introduced $\boldsymbol{z}(t) = \int_0 \boldsymbol{w}(q)dq$. $\boldsymbol{z}(t)$ is called a Wiener process, and the infinitesimal increment $d\boldsymbol{z}(t) = \boldsymbol{z}(t + dt) - \boldsymbol{z}(t)$ has variance $\boldsymbol{S}dt$

With this notion, the differential equation (A2.28) can be written

$$d\boldsymbol{x} = \boldsymbol{A}\boldsymbol{x}dt + \boldsymbol{B}d\boldsymbol{z}. \tag{A2.29}$$

Integration of (A2.29) over one sampling period, T, yields

$$\boldsymbol{x}((k+1)T) = e^{\boldsymbol{A}T}\boldsymbol{x}(kT) + \int_{kT}^{(k+1)T} e^{\boldsymbol{A}((k+1)T-q)}\boldsymbol{B}d\boldsymbol{z}(q). \tag{A2.30}$$

Define the random variable $\boldsymbol{e}(kT) = \int_{kT}^{(k+1)T} e^{\boldsymbol{A}((k+1)T-q)}\boldsymbol{B}d\boldsymbol{z}(q)$. It has zero mean because $\boldsymbol{z}(q)$ has zero mean. The random vectors $\boldsymbol{e}(kT)$ and $\boldsymbol{e}(iT)$ are also uncorrelated for $k \neq i$, because the increments of $\boldsymbol{z}$ over disjoint intervals are uncorrelated. The variance of $\boldsymbol{e}(kT)$ is given by

$$E(\boldsymbol{e}(kT)\boldsymbol{e}(kT)')$$

$$= E(\int_{kT}^{(k+1)T} \int_{kT}^{(k+1)T} e^{\boldsymbol{A}((k+1)T-q)}\boldsymbol{B}d\boldsymbol{z}(q)d\boldsymbol{z}(t)'\boldsymbol{B}'e^{\boldsymbol{A}'((k+1)T-t)}$$

$$= \int_{kT}^{(k+1)T} e^{\boldsymbol{A}((k+1)T-q)}\boldsymbol{B}\boldsymbol{S}\boldsymbol{B}'e^{\boldsymbol{A}'((k+1)T-q)}dq.$$

Thus the sampled data model associated to (A2.28) can be written

$$\boldsymbol{x}((k+1)T) = e^{\boldsymbol{A}T}\boldsymbol{x}(kT) + \boldsymbol{e}(kT),$$

where $\boldsymbol{e}(kT)$ is a sample of a discrete time white noise sequence with variance given by (A2.31)

Calculation of the variance $Q_e = E(\boldsymbol{e}(kT)\boldsymbol{e}(kT)')$ (unit is $[amplitude^2]$) when sampling the continuous differential equation with white noise input (input intensity $\boldsymbol{S}$ $[amplitude^2/s]$) follows from Eq. (A2.31). With the usual approximation of the matrix exponential

$$e^{\boldsymbol{A}t} = \boldsymbol{I} + \boldsymbol{A}\tau + \frac{1}{2!}\boldsymbol{A}^2\tau^2 - ... \tag{A2.31}$$

A rapid approximation is to include only one term in the series expansion. We then get the approximation

$$\boldsymbol{Q}_e \simeq \boldsymbol{B}\boldsymbol{S}\boldsymbol{B}'T. \tag{A2.32}$$

Appendix 3

$\mathcal{H}_2$ and $\mathcal{H}_\infty$ controller design

This appendix reviews the $\mathcal{H}_2$ and $\mathcal{H}_\infty$ controller design for standard systems. First, the central controller solution is discussed and then all suboptimal $\mathcal{H}_2$ or $\mathcal{H}_\infty$ controllers are derived by using $\boldsymbol{Q}$-parametrisation.

$\mathcal{H}_2$ Controller Design

Consider a standard $\mathcal{H}_2$ design problem where the system $\boldsymbol{G}$ is given by the following state-space realisation:

$$\boldsymbol{G} = \left(\begin{array}{c|cc} \boldsymbol{A} & \boldsymbol{B}_1 & \boldsymbol{B}_2 \\ \hline \boldsymbol{C}_1 & \boldsymbol{O} & \boldsymbol{D}_{12} \\ \boldsymbol{C}_2 & \boldsymbol{D}_{21} & \boldsymbol{O} \end{array}\right). \tag{A3.1}$$

Notice that the matrix $\boldsymbol{D}$ has a special structure. If $\boldsymbol{D}$ does not have this structure, it is then impossible to design an $\mathcal{H}_2$ controller for the design problem.

For the design of an optimal $\mathcal{H}_2$ controller for the system given in (A3.1), some assumptions on $\boldsymbol{G}$ are the following:

- $(\boldsymbol{A}, \boldsymbol{B}_2)$ is stabilisable and $(\boldsymbol{C}_2, \boldsymbol{A})$ is detectable.
- $\boldsymbol{D}_{12}$ has full column rank, i.e. $\boldsymbol{D}_{12}$ is injective.
- $\boldsymbol{D}_{21}$ has full row rank, i.e. $\boldsymbol{D}_{21}$ is surjective.
- $\begin{pmatrix} \boldsymbol{A} - j\omega\boldsymbol{I} & \boldsymbol{B}_2 \\ \boldsymbol{C}_1 & \boldsymbol{D}_{12} \end{pmatrix}$ has full column rank for all ω.

- $\begin{pmatrix} \boldsymbol{A} - j\omega \boldsymbol{I} & \boldsymbol{B}_1 \\ \boldsymbol{C}_2 & \boldsymbol{D}_{21} \end{pmatrix}$ has full row rank for all ω.

The first assumption is for the stability of $\boldsymbol{G}$ by output feedback and the fourth and fifth assumption together with the first guarantee that the two Hamiltonian matrices associated with the $\mathcal{H}_2$ problem below belong to *dom(Ric)*. The rank assumption guarantee that the $\mathcal{H}_2$ optimal control problem is nonsingular. However, these assumptions can without problems be relaxed and the problem can still be solved, but the controller will not necessary be unique.

The $\mathcal{H}_2$ design problem is as follows:

Problem. *The $\mathcal{H}_2$ control problem is to find a proper, real-rational controller $\boldsymbol{K}$ which stabilise $\boldsymbol{G}$ internally and minimises the $\mathcal{H}_2$ norm of the transfer matrix $\boldsymbol{T}_{zw}$ from w to z.*

The optimal solution to this $\mathcal{H}_2$ design problem is given by the following theorem.

Theorem. *Let the system $\boldsymbol{G}$ given by (A3.1) which satisfies the five conditions given just after (A3.1). Then the optimal controller is given by*

$$\boldsymbol{K}_2 = \left(\begin{array}{c|c} \boldsymbol{A} + \boldsymbol{B}_2\boldsymbol{F}_2 + \boldsymbol{L}_2\boldsymbol{C}_2 & -\boldsymbol{L}_2 \\ \hline \boldsymbol{F}_2 & \boldsymbol{O} \end{array} \right), \tag{A3.2}$$

where $\boldsymbol{F}_2$ and L_2 are the state feedback gain and the observer gain or respectively. The two gains are given by

$$\begin{aligned} \boldsymbol{F}_2 &= -(\boldsymbol{D}_{12}'\boldsymbol{D}_{12})^{-1}(\boldsymbol{B}_2'\boldsymbol{X}_2 + \boldsymbol{D}_{12}'\boldsymbol{C}_1) \\ \boldsymbol{L}_2 &= -(\boldsymbol{Y}_2\boldsymbol{C}_2' + \boldsymbol{B}_1\boldsymbol{D}_{21}')(\boldsymbol{D}_{21}\boldsymbol{D}_{21}')^{-1}, \end{aligned}$$

where $\boldsymbol{X}_2 \geq 0$ and $\boldsymbol{Y}_2 \geq 0$ are the stabilising solutions of the following two Riccati equations:

$$\begin{aligned} \boldsymbol{O} = {} & \boldsymbol{A}'\boldsymbol{X}_2 + \boldsymbol{X}_2 A + \boldsymbol{C}_1'\boldsymbol{C}_1 - (\boldsymbol{X}_2\boldsymbol{B}_2 + \boldsymbol{C}_1'\boldsymbol{D}_{12})(\boldsymbol{D}_{12}'\boldsymbol{D}_{12})^{-1} \cdot \\ & \cdot(\boldsymbol{B}_2'\boldsymbol{X}_2 + \boldsymbol{D}_{12}'\boldsymbol{C}_1) \end{aligned}$$

$$\begin{aligned} \boldsymbol{O} = {} & \boldsymbol{A}\boldsymbol{Y}_2 + \boldsymbol{Y}_2\boldsymbol{A}' + \boldsymbol{B}_1\boldsymbol{B}_1' - (\boldsymbol{Y}_2\boldsymbol{C}_2' + \boldsymbol{B}_1\boldsymbol{D}_{21}')(\boldsymbol{D}_{21}\boldsymbol{D}_{21}')^{-1} \cdot \\ & \cdot(\boldsymbol{C}_2\boldsymbol{Y}_2 + \boldsymbol{D}_{21}\boldsymbol{B}_1') \end{aligned}$$

i.e.

$$\boldsymbol{A} - \boldsymbol{B}_2(\boldsymbol{D}_{12}'\boldsymbol{D}_{12})^{-1}(\boldsymbol{B}_2'\boldsymbol{X}_2 + \boldsymbol{D}_{12}'\boldsymbol{C}_1)$$

and

$$\boldsymbol{A} - (\boldsymbol{Y}_2\boldsymbol{C}_2' + \boldsymbol{B}_1\boldsymbol{D}_{21}')(\boldsymbol{D}_{21}\boldsymbol{D}_{21}')^{-1}\boldsymbol{C}_2$$

are stable. Moreover, the $\mathcal{H}_2$ norm of the closed-loop transfer matrix $\boldsymbol{T}_{zw}$ is given by

$$\|\boldsymbol{T}_{zw}\|_2^2 = \|\boldsymbol{G}_c\boldsymbol{B}_1\|_2^2 + \|\boldsymbol{F}_2\boldsymbol{G}_f\|_2^2 = \|\boldsymbol{G}_c\boldsymbol{L}_2\|_2^2 + \|\boldsymbol{C}_1\boldsymbol{G}_f\|_2^2,$$

where

$$\boldsymbol{G}_c = \left(\begin{array}{c|c} \boldsymbol{A} + B_2\boldsymbol{F}_2 & I \\ \hline \boldsymbol{C}_1 + \boldsymbol{D}_{12}\boldsymbol{F}_2 & \boldsymbol{O} \end{array}\right), \; \boldsymbol{G}_f = \left(\begin{array}{c|c} \boldsymbol{A} + \boldsymbol{L}_2\boldsymbol{C}_2 & \boldsymbol{B}_1 + \boldsymbol{L}_2\boldsymbol{D}_{21} \\ \hline \boldsymbol{I} & \boldsymbol{O} \end{array}\right).$$

The above controller is also named as the central $\mathcal{H}_2$ controller. Further, it is possible to derive a parametrisation of all $\mathcal{H}_2$ controllers which make the $\mathcal{H}_2$ norm of the closed loop smaller than a specified level γ by using the $\boldsymbol{Q}$-parametrisation. We have the following result.

Theorem. *The family of all admissible controllers such the* $\|\boldsymbol{T}_{zw}\|_2 < \gamma$ *equals the set of all transfer matrices from* y *to* u *in* $\mathcal{F}_l(M_2, Q)$ *where*

$$\boldsymbol{M}_2 = \left(\begin{array}{c|cc} \boldsymbol{A} + B_2\boldsymbol{F}_2 + \boldsymbol{L}_2\boldsymbol{C}_2 & -\boldsymbol{L}_2 & \boldsymbol{B}_2 \\ \hline \boldsymbol{F}_2 & \boldsymbol{O} & \boldsymbol{I} \\ -\boldsymbol{C}_2 & \boldsymbol{I} & \boldsymbol{O} \end{array}\right)$$

and where $\boldsymbol{Q} \in \mathcal{RH}_2$, $\|\boldsymbol{Q}\|_2^2 < \gamma^2 - (\|\boldsymbol{G}_c B_1\|_2^2 + \|\boldsymbol{F}_2\boldsymbol{G}_f\|_2^2)$.

$\mathcal{H}_\infty$ Controller Design. In a standard $\mathcal{H}_\infty$ design problem where the system $\boldsymbol{G}$ is given by the state-space realisation in (A3.1). The special structure for the $\boldsymbol{D}$ matrix is retained. If $\boldsymbol{D}$ does not have this structure, loop shifting technique can be used such that the $\mathcal{H}_\infty$ design problem is transformed to have the system's $\boldsymbol{D}$ matrix in the required off-diagonal form.

Again, it is assumed that the system $\boldsymbol{G}$ satisfies the following conditions, hence the suboptimal $\mathcal{H}_\infty$ controller exists:

- $(\boldsymbol{A}, \boldsymbol{B}_2)$ is stabilisable and $(\boldsymbol{C}_2, \boldsymbol{A})$ is detectable.
- $\boldsymbol{D}_{12}$ has full column rank, i.e. $\boldsymbol{D}_{12}$ is injective.
- $\boldsymbol{D}_{21}$ has full row rank, i.e. $\boldsymbol{D}_{21}$ is surjective.
- $\begin{pmatrix} \boldsymbol{A} - j\omega\boldsymbol{I} & \boldsymbol{B}_2 \\ \boldsymbol{C}_1 & \boldsymbol{D}_{12} \end{pmatrix}$ has full column rank for all ω.
- $\begin{pmatrix} \boldsymbol{A} - j\omega\boldsymbol{I} & \boldsymbol{B}_1 \\ \boldsymbol{C}_2 & \boldsymbol{D}_{21} \end{pmatrix}$ has full row rank for all ω.

Based on the description of the system $\boldsymbol{G}$, the following formulation of the suboptimal $\mathcal{H}_\infty$ design problem is obtained:

Problem. *Let* $\gamma > 0$ *be given. Design a proper, real-rational controller* $\boldsymbol{K}$*, if such one exist, which stabilises* $\boldsymbol{G}$ *internally and make the* $\mathcal{H}_\infty$ *norm of the transfer matrix* $\boldsymbol{T}_{zw}$ *from* $\boldsymbol{w}$ *to* $\boldsymbol{z}$ *smaller than* γ.

It is important to note that we are not looking after an optimal $\mathcal{H}_\infty$ controller, but instead after a controller which satisfies a pre-specified $\mathcal{H}_\infty$ norm bound. As a

consequently of this, not only one controller will satisfy the $\mathcal{H}_\infty$ norm constrain, a lot of controllers will do it. It is in general not possible or quite difficult to design an optimal $\mathcal{H}_\infty$ controller when we are working with state-space description. If we use a polynomial approach, it is much more easy to design optimal $\mathcal{H}_\infty$ controllers. However, on the other hand, it is much more difficult to use the polynomial methods in numerical optimisation.

The suboptimal $\mathcal{H}_\infty$ controller is given by the following theorem.

Theorem. *Let the system $\boldsymbol{G}$ given by (A3.1) which satisfies the five conditions given above. Then a suboptimal $\mathcal{H}_\infty$ controller which will make the $\mathcal{H}_\infty$ norm of the closed-loop transfer matrix from w to z smaller than γ is given by*

$$K_\infty = \left(\begin{array}{c|c} \boldsymbol{A}+\gamma^{-2}B_1B_1'X_\infty + B_2\boldsymbol{F}_\infty + Z_\infty L_\infty(\boldsymbol{C}_2+\gamma^{-2}\boldsymbol{D}_{21}B_1'X_\infty) & -Z_\infty L_\infty \\ \hline \boldsymbol{F}_\infty & \boldsymbol{O} \end{array}\right),$$

where

$$\begin{aligned} \boldsymbol{F}_\infty &= -(\boldsymbol{D}_{12}'\boldsymbol{D}_{12})^{-1}(\boldsymbol{B}_2'\boldsymbol{X}_\infty + \boldsymbol{D}_{12}'\boldsymbol{C}_1) \\ \boldsymbol{L}_\infty &= -(\boldsymbol{Y}_\infty\boldsymbol{C}_2' + \boldsymbol{B}_1\boldsymbol{D}_{21}')(\boldsymbol{D}_{21}\boldsymbol{D}_{21}')^{-1} \\ \boldsymbol{Z}_\infty &= (\boldsymbol{I}-\gamma^{-2}\boldsymbol{Y}_\infty\boldsymbol{X}_\infty)^{-1}. \end{aligned}$$

$\boldsymbol{X}_\infty \geq 0$ and $\boldsymbol{Y}_\infty \geq 0$ are the stabilising solutions of the following two Riccati equations

$$\begin{aligned} \boldsymbol{O} &= \boldsymbol{A}'\boldsymbol{X}_\infty + \boldsymbol{X}_\infty\boldsymbol{A} + \boldsymbol{C}_1'\boldsymbol{C}_1 + \gamma^{-2}\boldsymbol{X}_\infty\boldsymbol{B}_1\boldsymbol{B}_1'\boldsymbol{X}_\infty \\ &\quad -(\boldsymbol{X}_\infty\boldsymbol{B}_2 + \boldsymbol{C}_1'\boldsymbol{D}_{12})(\boldsymbol{D}_{12}'\boldsymbol{D}_{12})^{-1}(\boldsymbol{B}_2'\boldsymbol{X}_\infty + \boldsymbol{D}_{12}'\boldsymbol{C}_1) \\ \boldsymbol{O} &= \boldsymbol{A}\boldsymbol{Y}_\infty + \boldsymbol{Y}_\infty\boldsymbol{A}' + \boldsymbol{B}_1\boldsymbol{B}_1' + \gamma^{-2}\boldsymbol{Y}_\infty\boldsymbol{C}_1'\boldsymbol{C}_1\boldsymbol{Y}_\infty \\ &\quad -(\boldsymbol{Y}_\infty\boldsymbol{C}_2' + \boldsymbol{B}_1\boldsymbol{D}_{21}')(\boldsymbol{D}_{21}\boldsymbol{D}_{21}')^{-1}(\boldsymbol{C}_2\boldsymbol{Y}_\infty + \boldsymbol{D}_{21}\boldsymbol{B}_1') \end{aligned} \tag{A3.3}$$

and satisfy the coupling condition

$$\rho(X_\infty Y_\infty) < \gamma^2,$$

i.e.

$$\begin{aligned} &\boldsymbol{A}+\gamma^{-2}B_1B_1'X_\infty - B_2(\boldsymbol{D}_{12}'\boldsymbol{D}_{12})^{-1}(B_2'X_\infty + \boldsymbol{D}_{12}'\boldsymbol{C}_1) \\ &\boldsymbol{A}+\gamma^{-2}Y_\infty\boldsymbol{C}_1'\boldsymbol{C}_1 - (Y_\infty\boldsymbol{C}_2' + B_1\boldsymbol{D}_{21}')(\boldsymbol{D}_{21}\boldsymbol{D}_{21}')^{-1}\boldsymbol{C}_2 \end{aligned}$$

are stable.

Note that the two $\mathcal{H}_\infty$ Riccati equations in the theorem have an additional term compared to the Riccati equations for the $\mathcal{H}_2$ design problem. Further, a coupling condition has also been included in the $\mathcal{H}_\infty$ case compared with the $\mathcal{H}_2$ case. The controller in the theorem is also named as the central $\mathcal{H}_\infty$ controller.

It is also here required that the direct term, $\boldsymbol{D}_{12}$ and $\boldsymbol{D}_{21}$ have full rank. If this is not the case, the two Riccati equations are replaced by two quadratic matrix inequal-

ities and the equations for the two gains are replaced by two almost disturbances decoupling problems (ADDP).

As in the $\mathcal{H}_2$ case, it is also possible to derive a parametrisation of all $\mathcal{H}_\infty$ controllers which make the $\mathcal{H}_\infty$ norm of the closed loop smaller than a specified level 1 (a scaling has been used so $\gamma = 1$, this is without loss of generality) by using the $\boldsymbol{Q}$-parametrisation. We have the following result.

Theorem. *The family of all admissible controllers such the* $\|\boldsymbol{T}_{zw}\|_\infty < 1$ *equals the set of all transfer matrices from* $\boldsymbol{y}$ *to* $\boldsymbol{u}$ *in* $\mathcal{F}_l(\boldsymbol{M}_\infty, \boldsymbol{Q})$ *where*

$$\boldsymbol{M}_\infty = \left(\begin{array}{c|cc} \boldsymbol{A} + \boldsymbol{B}_1\boldsymbol{B}_1'\boldsymbol{X}_\infty + \boldsymbol{B}_2\boldsymbol{F}_\infty + \boldsymbol{Z}_\infty\boldsymbol{L}_\infty\tilde{C}_2 & -\boldsymbol{Z}_\infty\boldsymbol{L}_\infty & Z_\infty\tilde{\boldsymbol{B}}_2 \\ \hline \boldsymbol{F}_\infty & \boldsymbol{O} & \boldsymbol{I} \\ -\tilde{\boldsymbol{C}}_2 & \boldsymbol{I} & \boldsymbol{O} \end{array}\right),$$

where

$$\begin{aligned} \tilde{\boldsymbol{B}}_2 &= \boldsymbol{B}_2 + \boldsymbol{Y}_\infty\boldsymbol{C}_1'\boldsymbol{D}_{12} \\ \tilde{\boldsymbol{C}}_2 &= \boldsymbol{C}_2 + \boldsymbol{D}_{21}\boldsymbol{B}_1^{\boldsymbol{T}}\boldsymbol{X}_\infty \end{aligned}$$

$\boldsymbol{Q} \in \mathcal{RH}_\infty,\ \|\boldsymbol{Q}\|_\infty < 1.$

The non-regular case

The $\mathcal{H}_2$ and $\mathcal{H}_\infty$ design methods considered in this section, as well as the methods in [220], are based on the condition that the two direct matrices $\boldsymbol{D}_{12}$ and $\boldsymbol{D}_{21}$ in (7.105) have full rank, also know as the regular case. However, this condition might not always be satisfied. The case where the two direct matrices $\boldsymbol{D}_{12}$ and $\boldsymbol{D}_{21}$ does not have full rank is normal called for the singular case or the non-regular case.

It is possible to handle this singular case in a number of ways. As an ad-hoc method, the two direct matrices can be regular be adding a small perturbation and then use the standard regular $\mathcal{H}_2$ or $\mathcal{H}_\infty$ design method. The solution need to be investigated with respect to the sensitivity of the perturbation. This can be done by change the perturbation and check that the solution is "almost" independent of the perturbation.

The theoretical correct way is to use a design method that can handle the design problem directly without using any perturbation. One method is the singular $\mathcal{H}_2$ and the singular $\mathcal{H}_\infty$ design method. The singular methods are using Riccati inequalities instead of Riccati equations in the regular case. It is quite more complicated to solve these Riccati inequalities than Riccati equations. First, the Riccati inequality is transformed into a reduced order Riccati equation, that can be solved as a standard Riccati equation. There does not exist any standard toolbox for solving Riccati inequalities.

Another method is the use linear matrix inequality-based $\mathcal{H}_2$ and $\mathcal{H}_\infty$ design methods. Again, there is no condition on the two direct matrices. Further, the linear matrix inequality-based method can also handle other cases that cannot be handled

by the regular Riccati based method. A MATLAB toolbox exist for analysis and design using linear matrix inequalities.

Appendix 4

Nomenclature

The symbols are used according to the following conventions. Scalars are represented by italics like i, j, a, s, vectors by bold lower-case letters like $\boldsymbol{x}$, $\boldsymbol{u}$ and matrices by bold upper-case letters like $\boldsymbol{A}$, $\boldsymbol{K}$. Sets are denoted by calligraphic letters like $\mathcal{Y}$.

Abbreviation	Meaning	introduced on p.
ARR	Analytical Redundancy Relation	156
CUSUM	CUmulative SUM	239
FMEA	Failure-Modes and Effects Analysis	85
GLR	Generalised Likelihood Ratio	239
LFT	Linear Fractional Transformation	356
LQ	Linear Quadratic (regulator)	309
LTI	Linear time-invariant	233
UM	Use-Mode	70

Appendix 5

Terminology

The terminology used in diagnosis and fault-tolerant control literature has only during the recent years approached a coherency in the published material. The Safeprocess Technical Committee of IFAC, the International Federation of Automatic Control, has compiled a list of suggested definitions ([100]), which is generally in accordance with the terminology used throughout this book.

An exception is the use of the word estimation where the Safeprocess terminology is identification. The word identification is widely used as a synonym for system identification or system parameter identification. Estimation is commonly used when the magnitude of a signal is reconstructed, which is what is done in this book.

Active fault-tolerant control system	A fault-tolerant system where faults are explicitly detected and accommodated. Opposite of a passive fault-tolerant system.
Analytical redundancy	Use of two or more, but not necessarily identical ways to determine a variable where one way uses a mathematical process model in analytical form.
Availability	Likelihood that a system or equipment will operate satisfactorily and effectively at any given point in time.
Constraint	The limitation imposed by nature (physical laws) or man. It permits the variables to take only certain values in the variable space.
Decision logic	The functionality that determines which remedial action(s) to execute in case of a reported fault and which alarm(s) shall be generated.
Discrepancy	An abnormal behaviour of a physical value or inconsistency between more physical values and the relationship between them.

Error	Deviation between a measured or computed value (of an output variable) and the true, specified, or theoretically correct value.
Fail-safe	The ability to sustain a failure and retain the capability to make a safe close-down. A system where the occurrence of a single fault can be determined but not isolated and where the fault cannot be accommodated to continue operation.
Fail-operational	The ability to sustain any single point failure.
Failure	Permanent interruption of a systems ability to perform a required function under specified operating conditions.
Failure effect	The consequence of a failure mode on the operation, function, or status of an item.
Failure mode	Particular way in which a failure can occur.
Fault	Unpermitted deviation of at least one characteristic property or parameter of a system from its acceptable/usual/standard condition. A fault is the occurrence of a failure mode.
Fault accommodation	The action of changing the control law in response to fault, without switching off any system component. In fault accommodation, faulty components are still kept in operation thanks to an adapted control law.
Fault detection	Determination of faults present in a system and time of detection.
Fault detector	An algorithm that performs fault detection and isolation.
Fault diagnosis	Determination of kind, size, location, and time of occurrence of a fault. Fault diagnosis includes fault detection, isolation and estimation.
Fault estimation	Determination of a model of the faulty system.
Fault identification	Determination of the sihze and time-varying behaviour of a fault. Follows fault isolation. Used as a synonym for fault estimation.
Fault isolation	Determination of the location of a fault. Follows fault detection.
Fault modelling	Determination of a mathematical model to describe a specific fault effect.
Fault propagation analysis	Analysis to determine how certain fault effects propagate through the considered system.
Fault recovery	The result of a successful fault accommodation or system reconfiguration.
Fault-tolerant system	A system where a fault is recovered with or without performance degradation, but a single fault does not develop into a failure on subsystem or system level.
Hardware redundancy	Use of more than one independent instrument to accomplish a given function.
Incipient fault	A fault where the effect develops slowly e.g. clogging of a valve. In opposite to an abrupt fault.
Objective	The control specification, the aim of the control system.

Objective reconfiguration	The action of changing the objective of the control system. Objective reconfiguration is mandatory when unrecoverable faults for the current objective occur.
Passive fault-tolerant control system	A fault-tolerant system where faults are not explicitly detected and accommodated, but the controller is designed to be insensitive to a certain restricted set of faults. Contrary to an active fault-tolerant system.
Qualitative model	A system model describing the behaviour with relations among system variables and parameters in heuristic terms such as causalities or if-then rules.
Quantitative model	A system model describing the behaviour with relations among system variables and parameters in analytical terms such as differential or difference equations.
System reconfiguration	The action of switching off the faulty components and accordingly changing the control law, in response to a fault. In system reconfiguration, faulty components are no longer employed.
Recoverability	Possibility ot accommodate the fault or to reconfigure the system if fault occurs.
Reliability	Probability of a system to perform a required function under normal conditions and during a given period of time.
Remedial action	A correcting action (reconfiguration or a change in the operation of a system) that prevents a certain fault to propagate into an undesired end-effect.
Residual	Fault information carrying signals, based on deviation between measurements and model based computations.
Safety system	Electronic system that protects local subsystems from permanent damage or damage to environment when potential dangerous events occur.
Sensor fusion	Integration of information from different sensors taking into account the quality of the measurements provided by each of them.
Severity	A measure on the seriousness of fault effects using verbal characterisation. Severity considers the worst-case damage to equipment, damage to environment, or degradation of a system's operation.
Structural analysis	Analysis of the structural properties of the models, i.e. properties that are independent on the actual values of the parameter.
Supervision	Monitoring of a physical system and taking appropriate actions to maintain the operation in the case of faults.
Supervisor	A function that performs supervision using results of fault diagnosis, determines remedial actions when needed, and execute corrective actions to handle faults.
Threshold	Limit value of a residual's deviation from zero, so if exceeded, a fault is declared as detected.

Appendix 6

Dictionary

This appendix shows the main notions in English and the native languages of the four authors.

English	German	French	Danish
Basic notions in systems and control theory			
actuator	Aktor, Stellglied	actionneur	aktuator
analysis	Analyse	analyse	analyse
architecture	Aufbau, Architektur	architecture	arkitektur
automaton	Automat	automate	automat
block diagram	Blockschaltbild	schéma fonctionnel	blokdiagram
behaviour	Verhalten	comportement	opførsel
closed-loop system	Regelkreis	système en boucle fermée	lukketsløjfe system
continuous-variable system	wertkontinuierliches System	système à variables continues	kontinuert system
control	Steuerung, Regelung	commande, régulation	styring, regulering
controllability	Steuerbarkeit	gouvernabilité, commandabilité	styrbarhed

English	German	French	Danish
controller	Regler	régulateur, contrôleur	regulator
control design	Reglerentwurf	conception de régulateur	regulator design
control input	Stellgröße	commande	styresignal (procespåvirkning)
control loop	Regelkreis	boucle de régulation	reguleringssløjfe
control objective	Regelungsziel	consigne, objectif à réaliser	regulerings formål
control output	Regelgröße	signal de commande	styret variabel
discrete-event system	ereignisdiskretes System	système à événements discrets	diskret-hændelses system
disturbance	Störung	perturbation	forstyrrelse
event	Ereignis	événement	hændelse
feedback	Rückführung	boucle de retour	modkobling
feedback control	Regelung	régulation par rétroaction	lukketsløjfe styring eller - regulering
feedforward control	Steuerung (in der offenen Wirkungskette)	régulation avec action anticipative	fremkobling
input	Eingang	entrée	indgang
measurement	Messung, Messwert	signal de mesure	måling
model	Modell	modèle	model
modelling	Modellbildung	modélisation	modellering
model uncertainties	Modellunsicherheiten	erreur de modélisation	model usikkerhed
objective function	Gütefunktion	fonction de coût	kostfunktion
observability	Beobachtbarkeit	observabilité	observerbarhed
observation	Beobachtung	observation	observation
observer	Beobachter	observateur	observer
open-loop system	offene Kette	système en boucle ouverte	åbensløjfe system

English	German	French	Danish
optimal control	optimale Steuerung	régulation optimale, commande optimale	optimal regulering
output	Ausgang	sortie	udgang
Petri net	Petrinetz	réseau de Petri	Petri net
performance	Regelgüte	performance	performance
prediction	Vorhersage, Prädiktion	prédiction	prediktion
qualitative model	qualitatives Modell	modèle qualitatif	qualitativ model
quantised system	quantisiertes System	système quantifié	kvantiseret system
requirement	Forderung	cahier des charges	krav
specification	Güteforderung	spécification	specifikation
sensor	Sensor	capteur	giver, sensor
set-point	Sollwert	point de fonctionnement, grandeur de référence	setpunkt
signal	Signal	signal	signal
stability	Stabilität	stabilité	stabilitet
state	Zustand	état	tilstand
state space	Zustandsraum	espace d'état	tilstandsrum
structure	Struktur	structure	struktur
structure graph	Strukturgraf	graphe structurel	struktur graf
system	System	système	system
system identification	Systemidentifikation	identification des systèmes	system identifikation
Notions describing the system subject to faults			
actuator fault	Fehler im Stellglied	défaut d'actionneur	aktuator fejl
abrupt fault	plötzlich eintretender Fehler	défaut instantané	pludselig fejl

English	**German**	**French**	**Danish**
availability	Verfügbarkeit	disponibilité	tilgængelighed
consistency	Widerspruchsfreiheit, Konsistenz	cohérence, compatibilité	konsistens
consistency-based diagnosis	konsistenzbasierte Diagnose	diagnostic par invalidation (réfutation)	konsistensbaseret diagnose
decision logic	Entscheidungslogik	logique de décision	beslutningslogik
dependability	Verfügbarkeit	sûreté de fonctionnement	pålidelighed
deviation	Abweichung	déviation	afvigelse
diagnosis	Diagnose	diagnostic	diagnose
diagnostic algorithm	Diagnosealgorithmus	algorithme de diagnostic	diagnose algorithme
evaluation	Bewertung	évaluation	evaluering
failure	Versagen	panne	nedbrud
fault	Fehler	défault, défaillance	fejl
faultless system	fehlerfreies System	système sain	fejlfrit system
faulty system	fehlerbehaftetes System	système en défaut	fejlbehæftet system
fault-tolerant control	fehlertolerante Steuerung	commande tolérante aux fautes	fejltolerant regulering
fault detection	Fehlerdetektion	détection de défaillances	fejldetektion
fault diagnosis	Fehlerdiagnose	diagnostic de défaillances	fejldiganose
fault effects	Fehlerwirkung	effets d'une défaillance	fejleffekt
fault estimation	Schätzung der Fehlergröße	estimation de défaillance	fejlestimation
fault identification	Fehleridentifikation	identification de défaillance	fejlidentifikation
fault isolation	Fehlerisolation	localisation des défaillances	fejlisolation
fault probability	Fehlerwahrscheinlichkeit	probabilité d'une défaillance	fejlsandsynlighed
fault propagation	Fehlerfortpflanzung	propagation d'une défaillance	fejlpropagering

English	German	French	Danish
incipient fault	Fehler, der sich langsam entwickelt	défaillance naissante	fejl der udvikles langsomt
maintenance	Wartung	entretien, maintenance	vedligehold
model-based diagnosis	modellbasierte Diagnose	diagnostic fondé sur un modéle	modelbaseret diagnose
reconfiguration	Rekonfiguration	reconfiguration	rekonfiguration
redundancy	Redundanz	redondance	redundans
reliability	Zuverlässigkeit	fiabilité	pålidelighed
remedial action	Korrektur	action corrective	handling der skal genoprette normal funktion
repair	Reparatur	réparation	reparation
residual	Residuum	résidu	residual
robustness	Robustheit	robustesse	robusthed
safety	Sicherheit	sécurité, sûreté	sikkerhed
sensor fault	Sensorfehler	défaut de capteur	sensor fejl
sensor fusion	Sensorfusion	fusion de multi-capteurs	sensorfusion
shut-off	Abschaltung	arrêt	nedlukning
symptom	Symptom	symptôme	symptom
supervision	Überwachung	supervision	overvågning
threshold	Schranke	seuil	grænseværdi
tolerance	Toleranz	tolérance	tolerance

Subject index