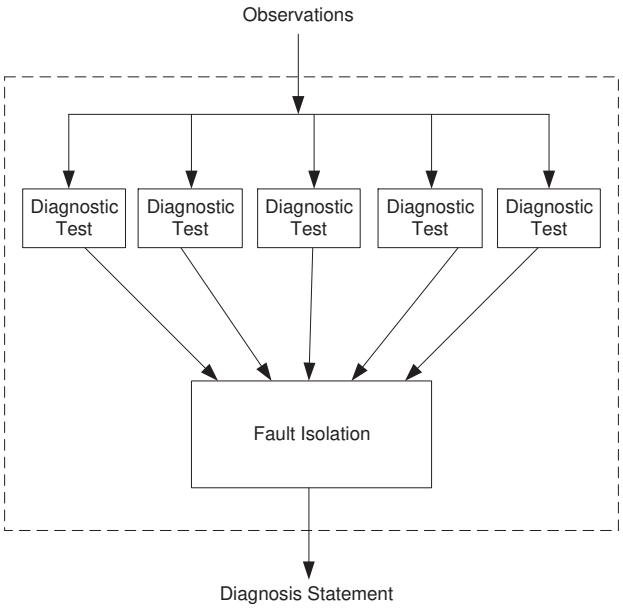
Model Based Diagnosis of Technical Processes

Mattias Nyberg, Erik Frisk



Preface

This version of this document is prepared for a course TSFS06, Diagnosis and Supervision, at the Department of Electrical Engineering, Linköpings universitet, spring semester 2020.

This text assumes basic knowledge in automatic control, linear algebra, mathematical statistics, probability theory, and logic. In some parts, methods from more advanced courses are used. Such parts of the text are not of central importance in this course and in those cases, references to relevant literature are given.

Contents

Chapter 1
Introduction to Fault Diagnosis

From a general perspective, including both the medical and technical case, diagnosis can be explained as follows. For a process there are observed variables or behavior for which there is knowledge of what is expected or normal. The task of diagnosis is to, from the observations and the knowledge, generate a *diagnosis*, i.e. to decide whether there is a fault or not and also to identify the fault. Thus the basic problems in the area of diagnosis are how the procedure for generating diagnoses should look like, what variables or behaviors that is relevant to study, and how to derive the knowledge of what is expected or normal.

This text focuses on diagnosis of technical systems and the goal is to find malfunctions in for example sensors and actuators. The observations are mainly signals obtained from the sensors, but can also be observations made by a human. Examples of such human observations is for example level of noise or vibrations. The diagnosis is computed by observing inconsistencies between observed variables and what is considered normal behavior. When the diagnosis is based on an explicit formal model of the system, the term *model based diagnosis* is used. Diagnosis of technical systems can be performed off-line or on-line. When on-line is considered, the diagnosis is usually automated so it is performed without involvement of humans. Most concepts described in this text are applicable to both off-line and on-line diagnosis.

1.1 The Use of Diagnosis

Diagnosis systems have found their way into many applications. In the context of model based diagnosis, some important areas that have been discussed in the literature are:

- Nearly all subsystems of aircrafts, e.g. aircraft control. systems, navigation systems, and engines.
- Emission control systems in automotive vehicles.
- Nuclear plants.
- Chemical plants.
- Gas turbines.
- Industrial robots.
- Electrical motors.

For these systems and also for technical processes in general, important reasons to incorporate diagnosis systems are:

- **Safety**
  In many technical systems a fault may cause serious personal damage. This is especially obvious in safety critical processes such as aircrafts and nuclear plants. For these systems, high reliability and security of the system is fundamental.
- **Environment Protection**
  In for example emission control systems in automotive vehicles, a fault may cause increased emissions. It has been concluded that a major part of the total emissions from cars originates from vehicles with malfunctioning emission control systems. Other important examples are nuclear plants and chemical plants in which a fault may cause serious damage to the environment.
- **Machine Protection**
  A fault can often cause damage to the machine. Therefore it is important that faults are detected as quickly as possible after they have occurred.

- **Availability**
  For many technical systems it is critical that the systems are running continuously. This is for example the case for gas turbines in power plants and industrial robots. The reasons may be economical as well as safety. With the help of a diagnosis system, early warnings can be obtained before a serious breakdown. When the fault has been detected, the system can be stopped until repair or rather be switched into a new mode. In the new mode, the performance of the system may be degraded but at least more serious breakdowns can be avoided.

- **Repairability**
  Closely connected to availability is repairability. A good diagnosis system will quickly identify the faulty component that should be replaced. In this way, time-consuming fault localization, is reduced, which will decrease total repair time.

- **Flexible Maintenance**
  Maintenance can be expensive since the machine/process often need to be taken out of operation. Therefore it is desirable to make sure that the machine is not taken out of operation for maintenance when there is no need for maintenance. Also, it is desirable to be able to plan maintenance stops in advance to be able to disturb the production as little as possible. A diagnosis system that detects faults early, desirably before more serious faults occur, can hopefully help both to avoid unnecessary maintenance and to indicate far in advance when a maintenance is needed.

1.1.1 A Short History

Manual diagnosis has been performed as long as there have existed technical systems, but automatic diagnosis started to appear first when computers, and on-line computing power, became available. In the beginning of the 70's, the first research reports on model based diagnosis were published. Some of the earliest areas that were investigated were chemical plants and aerospace applications. The research on model based diagnosis has since then been intensified during both the 80's and the 90's. Today, this is still an expansive research area.

Up to now, numerous methods for doing diagnosis have been published. Unfortunately many approaches are more ad hoc than systematic and it is fair to say that few general theories exist and there is not yet a complete understanding of the relations between different methods. This is reflected in the shortage of books in the area (see Section 1.12) and the fact that no general terminology has yet been agreed upon. However the importance of diagnosis is unquestioned. This can be exemplified by the computerized management systems for automotive engines used to control the engine. For these systems more than 50% of the software can nowadays be dedicated to diagnosis. The rest is for example for control.

1.2 Basic Definitions and Concepts

This section presents definitions and concepts that are central for the area of diagnosis. As a step towards a unified terminology, the IFAC (International Federation of Automatic Control) Technical Committee SAFEPROCESS has 1994 suggested preliminary definitions of some terms in the field. Following is a list of some common basic terms with explanations. The explanations are partly based on the definitions made by the IFAC Technical Committee SAFEPROCESS.

- **Fault**
  Unpermitted deviation of at least one characteristic property or variable of the system from acceptable/usual/standard/nominal behavior.

- **Failure**
  A fault that implies permanent interruption of a systems ability to perform a required function under specified operating conditions.

- **Disturbance**
  An unknown and uncontrolled input acting on the system.

- **Fault Detection**
  To determine if faults are present in the system and usually also the time when the fault occurred.

- **Fault Isolation**
  Determination of the location of the fault, i.e. which component or components that have failed.

- **Fault Identification**
  Determination of size and time-variant behavior of a fault.

- **Fault Accommodation**
  To reconfigure the system and/or the controller so that the operation can be maintained in spite of a present fault.

- **Fault Diagnosis**
  For the definition of this term, two common views exist in the literature. The first view includes fault detection, isolation, and identification, see for example (Gertler, 1991). The second view includes only fault isolation and identification, see for example (Isermann, 1984). Often the word *fault* is omitted so only the word *diagnosis* is used.

- **Diagnosis**
  The diagnosis system produces diagnoses. A diagnosis is a conclusion of what fault or combinations of faults that can explain the process behavior.

- **False Alarm**
  The event that an alarm is generated even though no faults are present.

- **Missed Alarm**
  The event that an alarm is *not* generated in spite of that a fault has occurred. This event may also be denoted *missed detection*.

- **Active (or Intrusive) Diagnosis**
  When the diagnosis is performed by actively exciting the system so that possible faults are revealed.

- **Passive Diagnosis**
  When the diagnosis is performed by passively studying the system without affecting its operation.

- **Fault Tolerant Control**
  Fault tolerant control is the whole chain of detecting and isolating the fault, and thereafter perform fault accommodation so that the fault is prevented from developing into a failure. The goal is to keep the plant availability but accept reduced performance in spite of the presence of faults.

Sometimes fault tolerant control is referred to as *active* fault tolerance when the controller is reconfigured when a fault is detected. For example if a fault in a sensor is detected, feedback from this sensor is replaced by feedback from an estimated value of the sensor output. In contrast to active fault tolerance there is also *passive* fault tolerance which means that one controller should provide satisfactory performance even in the presence of (limited sized) faults.

The term *fault diagnosis* is in this text used to denote the whole chain of fault detection, isolation, and identification. Diagnosis used in this way also serves as a name for the whole area of everything that has to do with diagnosis. If fault detection is excluded from the term *diagnosis*, as in the second view above, one gets a problem of finding a word describing the whole area. This can partly be solved by introducing the abbreviation FDI (Fault Detection and Isolation), which is common in literature taking the second view of the definition of the term diagnosis. As noted in some literature, FDI does not strictly contain fault identification. To solve this, also the abbreviation FDII (Fault Detection, Isolation, and Identification) has been used.

1.3 The Diagnosis System

The general structure of an application including a diagnosis system is shown in Figure 1.1. The plant, i.e. the system to be diagnosed,

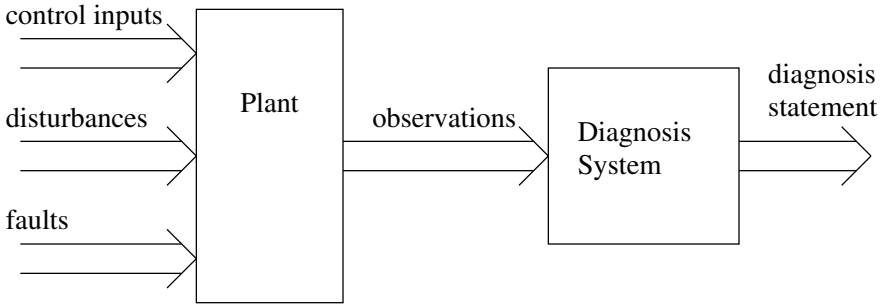


Figure 1.1: General structure of a diagnosis application.

is affected by *control inputs*, *disturbances*, and *faults*. Inputs to the

diagnosis system are the *observations* (which will also often be called the *measured data*). The observations consist of the known control inputs, measured sensor values, and possibly also some human observations. The task of the diagnosis system is to generate a *diagnosis statement* containing information about at least if a fault has occurred, and the location of the fault, i.e. in what component the fault occurred.

The diagnosis system can be considered to be a function from the observations to the diagnosis statement. In its simplest form, we could think of the diagnosis system as a table, mapping all possible observations to one diagnosis statement.

Example 1.1 Consider the electric circuit in Figure 1.2. The plant

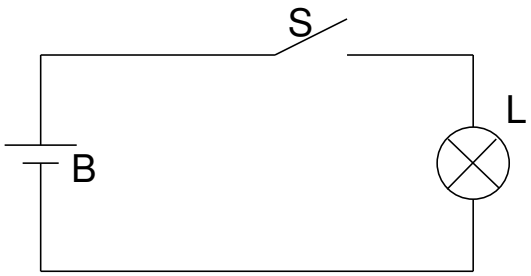


Figure 1.2: A simple electrical circuit.

consists of a battery (B), a switch (S), and a lamp (L). Assume now that only three types of faults can occur: the switch stuck in open position, the switch stuck in closed position, and the lamp is broken. The control input here is the desired switch position, either open or closed. The observations are the desired switch position and if the lamp is lit or not.

A diagnosis system for this simple system can be represented as in Table 1.1. Diagnosis is then performed by simple lookup in this table. For example, assume that the observation is ⟨"S open", "L not lit"⟩. Then according to Table 1.1, there are five explanations for the observation and the diagnosis statement becomes that either we have no faults, S is stuck open, L is broken, S is stuck closed and L broken, or S is stuck open and L broken.

Each of the explanations in a diagnosis statement is called a *diagnosis*. It is common to distinguish between *single faults* and *multiple faults*. A diagnosis can then indicate no fault, a single fault or a multi-

desired switch position	lamp observation	diagnosis statement
open	not lit	"no faults", "S stuck open", "L broken", "S stuck open and L broken", "S stuck closed and L broken"
open	lit	"S stuck closed"
closed	not lit	"S stuck open", "L broken", "S stuck open and L broken", "S stuck closed and L broken"
closed	lit	"no faults", "S stuck closed"

Table 1.1: A simple diagnosis system represented as a lookup table.

ple fault. In the first diagnosis statement in Table 1.1, there are two diagnoses indicating single faults, "S stuck open" and "L broken", and two diagnoses indicating a multiple fault, "S stuck open and L broken" and "S is stuck closed and L broken".

All diagnosis systems can in principle be represented as a table. However, when the number of possible faults increases, the table quickly becomes very large. Also, in a diagnosis problem described by continuous equations, it is not so easy to classify the observations as was done in Table 1.1. One further problem is to handle disturbances and noise, which makes it more complicated to derive "correct" diagnosis statements. In fact, how to find alternative representations of a diagnosis system, able to handle these problems, is the topic of this text.

1.4 Traditional and Model Based Diagnosis Systems

Traditionally, automated diagnosis has been performed by mainly limit checking. When for example a sensor signal level leaves its normal operating range, an alarm is generated. The normal operating range is predefined by using thresholds that may be dependent on the operating conditions. In for example an aircraft, the thresholds for different operating points defined by altitude and speed can be stored in a table. This use of thresholds as functions of some other variables can be

viewed as a kind of model based diagnosis. In addition to checking signal levels, also trends (e.g. the first derivative) of signals are often checked against thresholds.

Another traditional approach is duplication (or triplication or more) of hardware. This is called *hardware redundancy*. A typical example is to use two sensors measuring the same physical quantity, i.e. one of them is redundant. By comparing the two sensor outputs, a fault can be detected. However, it may be difficult to determine which of the two sensors that is faulty in case they differ. By introducing yet another sensor, i.e. to have three sensors measuring the same variable, also fault isolation is possible. Triple redundancy is common practice in safety critical components. The main advantage with hardware redundancy is that it is a highly reliable method of detecting faults. This is essential for example in inner control-loops of an aircraft. There are at least three issues associated with the use of hardware redundancy: hardware may be expensive, it requires space, and adds weight to the system. In addition, extra components increase the complexity of the system that in turn may introduce extra diagnostic requirements.

1.4.1 Model Based Diagnosis

As an alternative or complement to traditional approaches, model based diagnosis have shown to be useful either as a complement or on its own. The models used in model based diagnosis can be of any type, from logic based models to differential equations. Depending on the type of model, different approaches to model based diagnosis can be used, for example statistical approach (Basseville and Nikiforov, 1993), discrete event systems approach (Sampath et al., 1995), AI-based approaches (Reiter, 1987), and approaches within the framework of control theory. Compared to traditional limit checking, model based diagnosis has potentially the following advantages: because of the following reasons:

- It can provide higher diagnosis performance, for example smaller faults can be detected with a shorter detection time.
- It can be performed over a large operating range.
- It can to a higher degree be performed passively.
- The possibilities for isolation of faults increases.

- Disturbances can be compensated for, which implies that high diagnosis performance can be obtained in spite of the presence of disturbances.

Compared to using hardware redundancy, model based diagnosis may offer cost effective solutions due to:

- Model based diagnosis is generally applicable to more kinds of components. Some hardware components, such as the plant itself, can not be duplicated.
- No extra hardware is needed, which means for example that it is cheap in production (but not in development).

In model based diagnosis, a sensor output can, instead of being compared with another redundant sensor, be compared with the output from a model. The expression *analytical redundancy* is often used to highlight this idea that "analytical" models replace hardware redundancy. Analytical redundancy will be more exactly defined in Section 1.10.

It is important to note that model based diagnosis need not require a lot of computing power, it is highly dependent on the complexity of the model used. If simple models are used, generally simple algorithms are the result. Actually for the same level of performance of a model based system compared to a hardware based, it can be the case that model based diagnosis is *less* computationally intensive than traditional approaches. It can also be argued that traditional diagnosis is just a special case of model based diagnosis.

The main disadvantage of model based diagnosis is quite naturally the need for a reliable model and possibly a more complex design procedure. In the actual design of a model based diagnosis system, it is likely that the major part of the work is spent on building the model. This model can however be reused, e.g. in control design.

The accuracy of the model is usually the major limiting factor of the performance of a model based diagnosis system. Compared to the area of model based control, it is more critical that the model is good since model based diagnosis systems operates in open loop. More often than in control, a linear model is not sufficient for satisfactory performance.

There are situations where model based diagnosis never can replace hardware redundancy. This is the case for critical components, such as

sensors used in the inner control loop of an aircraft, where it is important to quickly and accurately detect a faulty sensor and immediately switch to another sensor.

Following is a simple example of an industrial application of model based diagnosis.

Example 1.2

Consider Figure 1.3, containing a principle illustration of a spark-ignited combustion engine. The air enters at the left side, passes the throttle and the inlet manifold, mixes with fuel at the cylinder inlets, and finally the air/fuel mixture enters the cylinders. The engine in the figure has three sensors measuring the physical variables air mass-flow, manifold pressure, and engine speed. The air flow W into the cylinders can be modeled as a function of manifold pressure p and engine speed n, i.e. $W = g(p, n)$. The physics behind the function g is involved and can be modeled by a black-box model. In engine management systems, one common solution is to represent the function g as a lookup-table. By using this lookup-table an estimation of the air mass-flow can be obtained.

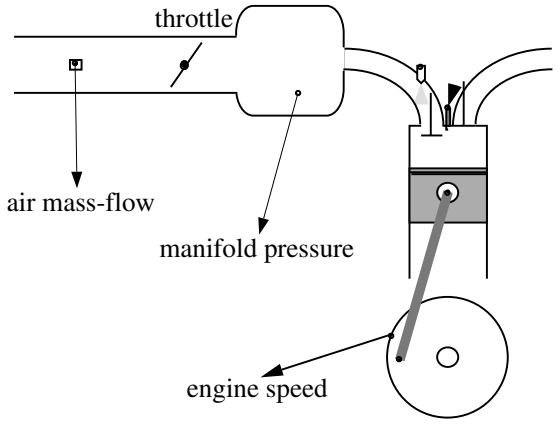


Figure 1.3: A principle illustration of an SI-engine.

When the measured air mass-flow significantly differs from the estimation, it can be concluded that a fault must be present somewhere in the engine. The fault can for example be that one of the three sensors are faulty or that a leakage have occurred somewhere between the air mass-flow sensor and the cylinder. This is an example of model

based diagnosis that is commonly used in production cars. Strictly speaking, in this simple version, only fault *detection* has been achieved. However, by using fault models (discussed further later in this chapter), this simple strategy can be expanded to also handle fault *isolation*.

1.5 Faults

As shown in Figure 1.4, a plant can often be separated into three subsystems: actuators, the process, and sensors. Depending on in what subsystem a fault occurs, a fault is classified to be an *actuator fault*, *process fault*, or *sensor fault*. Process faults are sometimes also called *system faults* or *component faults*.

Figure 1.4: General structure of a plant.

Typical sensor faults are short cut or cut-off in connectors and wirings, and drifts, i.e. changes in gain or bias. Also the time response can degrade due to a fault, i.e. the bandwidth of the sensor is decreased. Examples of process faults are increased friction, changed mass, leaks, components that get stuck or loose. Examples of faults in an actuator are short cut or cut-off in connectors and wirings. If the actuator includes an electrical amplifier, there can also be gain and bias faults. Actuators can by them self be relatively complex systems, containing for example DC-motors, controllers, and sensors. Therefore all examples of sensor and process faults are applicable also to actuators.

Another way of classifying faults is to study their time-variant behaviors. Figure 1.5 shows three typical time-variant behavior of faults. The solid line represents a so called *abrupt change*, i.e. the fault occurs abruptly and then stays present. The dash-dotted line represents an *incipient fault*, i.e., a fault that gradually increases in size. The dashed line represents an *intermittent fault*, i.e., a fault that occurs and disappears repeatedly.

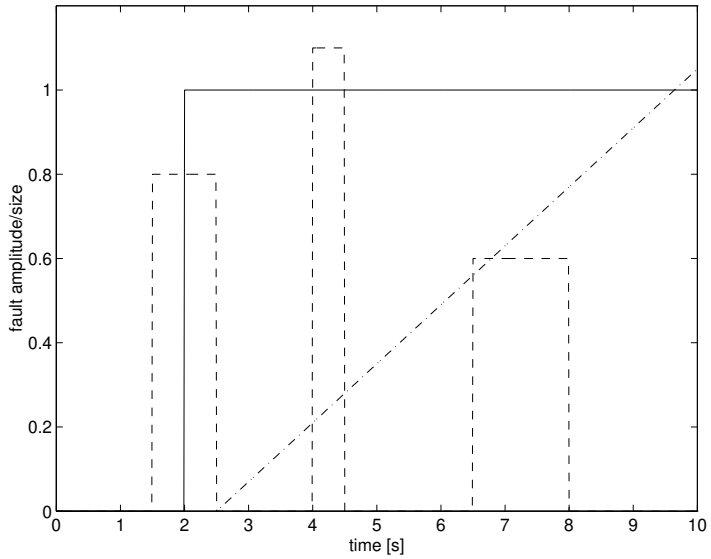


Figure 1.5: Different types of fault time-variant behavior.

In a diagnosis application it may not be sufficient to isolate a faulty (larger) component, e.g. a DC-motor. Often more detailed knowledge is required about the fault, e.g. what part of the DC-motor is faulty. Thus when designing a diagnosis system it is important to have knowledge about which faults that can occur or are most common, and also how different faults affect the system. Such knowledge can only be obtained from a domain expert and/or through extensive experiments.

1.6 Some Simple Examples of Model Based Diagnosis

In this section, we will exemplify how some different types of models and also standard mathematical techniques can be utilized for fault diagnosis.

1.6.1 Diagnosis Using Simulation Models

A simple example of model based diagnosis is the case when it is possible to both measure an output and, by means of a model, also estimate it. This is illustrated in Figure 1.6 where a *residual* $r(t)$ is

computed using the measured output $y(t)$ and estimated scalar output $\hat{y}(t)$. For example, the residual can be computed as

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

where $W(p)$ is any linear filter and p the time-differentiation operator. This filter can for example be a low-pass filter used to attenuate measurement noise or influence from model uncertainty. The model

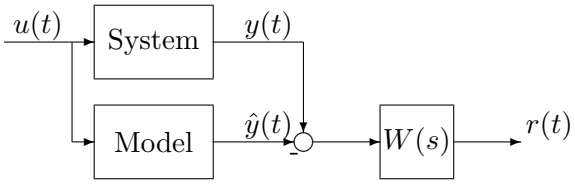


Figure 1.6: Simple fault detection system

used to estimate $\hat{y}(t)$ can be either linear or non-linear. If a fault occurs it will affect the measured output but not the estimated output. In this way the residual will deviate from zero and the fault can be detected.

To exemplify, consider the following linear single input, single output system

$$y = \frac{1}{s+2}(u + f)$$

where f is a fault acting on the input of the process, i.e. a fault in the actuator. An output estimator can in a simple case like this be written as

$$\hat{y} = \frac{1}{s+2}u$$

With a low-pass filter $W(s) = \frac{1}{s+1}$ we get the corresponding residual generator

$$r = \frac{1}{(s+1)}(y - \frac{1}{s+2}u) \tag{1.1}$$

Figure 1.7 shows the input and output of the process in fault free and faulty operation. Figure 1.8-a shows the time variant behavior of the fault, i.e. the fault occurs at $t = 7$, increases in amplitude until $t = 12$ when the fault disappears again. It is clear that by only observing the input and output in Figure 1.7 it is difficult to clearly

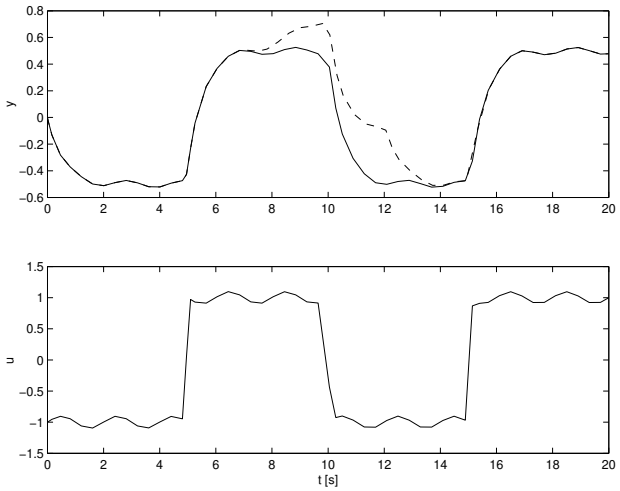


Figure 1.7: Fault free (solid lines) and faulty (dashed lines) simulation of the example.

detect that a fault has occurred since only minor differences in the signals occur. Computing the residual as in (1.1) we get a significantly simpler detection problem which is illustrated in Figure 1.8-b where the residual is 0 in the fault free case and differs from 0 when a fault occurs.

1.6.2 Diagnosis Using Parameter Estimation

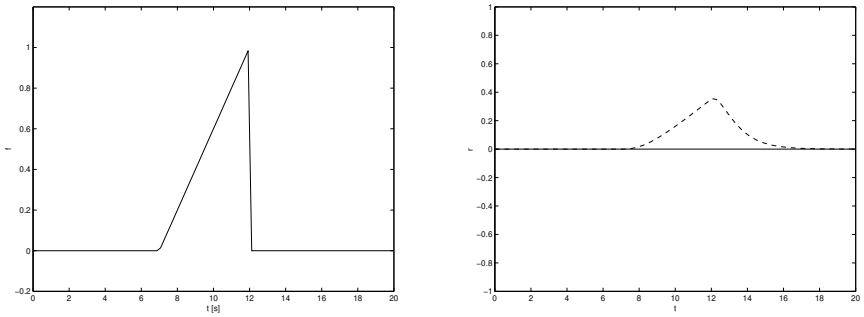
Consider now a case where the process can be modeled as

$$y(t) = G(q)u = \frac{b}{1+aq}u(t)$$

where q is the time-shift operator. Assume that the parameters a and b have nominal values a_0 and b_0, and there are two possible faults, one affecting parameter a and the other parameter b.

Next assume that a diagnosis system is constructed by using two parameter estimators estimating a and b respectively. The estimated values can be compared with the nominal values and we can set up two variables T_1 and T_2 according to

$$\begin{aligned} T_1 &= |\hat{a} - a_0| \\ T_2 &= |\hat{b} - b_0| \end{aligned}$$



(a) Time variant behavior of the fault.

(b) Residual in fault-free (solid line) and faulty (dashed-line) simulation.

Figure 1.8: Fault free and faulty simulation.

where $\hat{a}$ and $\hat{b}$ are estimated parameter values. The tasks of fault detection, isolation and identification can be performed by studying the values of T_1 and T_2. If for example T_1 is close to zero but T_2 is large, we conclude that the fault in parameter b has occurred.

Methods for the parameter estimation can be recursive, e.g. RLS (Recursive Least Square), or non-recursive. Information about parameter estimation methods can be found in the general system identification literature, see for example (Ljung, 1999), and will also be discussed later in Section 4.5.

1.6.3 Diagnosis with Fault Isolation

Consider the following system with 2 sensor signals y_1 and y_2, and one actuator signal u:

$$y_1 = 2u \tag{1.2a}$$
$$y_2 = 4u + 1 \tag{1.2b}$$

From these equations we can trivially derive two residuals:

$$r_1 = y_1 - 2u$$
$$r_2 = y_2 - 4u - 1$$

By using both equations (1.2), the input signal u can be eliminated and we get the relation

$$2y_1 - y_2 = -1$$

This relation can be used to form a third residual

$$r_3 = 2y_1 - y_2 + 1$$

In the fault-free case all the three residuals will be equal to zero. Now assume that the first sensor becomes faulty and therefore show the wrong value. Since residual r_1 is based on sensor y_1, it will then deviate from zero. Residual r_2 however, is calculated without using sensor y_1 and will therefore not be affected by the fault in sensor y_1. Lastly, residual r_3 will also be affected by the fault in sensor y_1 since its calculation includes y_1.

It can further be realized that a fault in sensor y_2 will not affect residual r_1, but will make r_2 and r_3 nonzero. Finally, a fault in the actuator u will affect r_1 and r_2, but not r_3. Thus, different faults in the process will make different residuals nonzero. This is the basis for isolation, which will be further discussed in Section 1.8.

1.7 Fault Detection in the View of Hypothesis Testing

Here we see how at least the fault detection problem can be formulated as a hypothesis testing problem. Later in Chapter 3, it will be seen that also the diagnosis problem, i.e. detection and isolation of faults, can be formulated by using the view of hypothesis testing. First note that a *hypothesis test* is defined as the decision problem of deciding between two possible decisions. One example is to decide if there is a fault present or not.

Assume that the process to be diagnosed contains a constant parameter θ which is zero when there is no fault and non-zero when there is a fault present. In this case, the two hypotheses can be written

$$\begin{aligned} H_0: &\quad \theta = 0 \quad \text{no fault} \\ H_1: &\quad \theta \neq 0 \quad \text{fault} \end{aligned}$$

Now the fault detection task becomes a hypothesis test between H_0 and H_1.

When constructing such a hypothesis test, we first find a so called *test quantity*. With test quantity we mean any quantity that is sensitive to the fault. Here the test quantity can for example be an estimation

of θ. Such a test quantity is close to zero if H_0 is true and non-zero if H_1 is true. Thus, by studying the test quantity we can determine if we should reject H_0, i.e. accept H_1, or not. If H_1 is accepted, an alarm should be generated.

Note that the previously mentioned residuals can easily be used to construct test quantities. A test quantity can for example be the mean value of a residual over a time window.

Because of disturbances and measurement noise, the test quantity is usually not exactly zero in the fault free case. Therefore we need to use a threshold that means that H_0 is rejected and H_1 is accepted if the test quantity is above the threshold.

The test quantities, which for example can be based on residuals, are central for diagnosis. As we have seen in the examples in Section 1.6, test quantities are used both to detect the faults, and by using several test quantities with different fault sensitivity properties, also fault isolation can be achieved.

1.8 Fault Isolation

Up to now, only fault detection has been considered. To achieve also fault isolation, several principles exist. In the area of automatic control, at least three different approaches can be distinguished: *fixed direction residuals*, *structured residuals*, and *structured hypothesis tests*. In AI, isolation has mainly been done using a logical reasoning about components.

The idea of *fixed direction residuals* (Beard, 1971) is to design a residual vector such that the residual responds in different directions depending on what fault that acts on the system. Fault isolation is then achieved by studying and classifying the direction of the residual. This approach has not been so much used in the literature, probably because the problems associated with designing a residual vector with desired properties.

The idea of *structured residuals* (Gertler, 1991) is to have a set of residuals, in which each individual residual is sensitive to a subset of faults. Recall that this was the case in the example in Section 1.6.3. By studying which residuals that responds, fault isolation can be achieved. Structured residuals have been widely used in the literature, in both theoretical and practical studies. The basic idea is quite simple and

many methods for constructing suitable residuals have been presented both for linear and non-linear systems.

Structured residuals is formalized and generalized with *structured hypothesis tests* (Nyberg, 1999), which will be the main approach in this text. In Section 1.7, a hypothesis test was used to detect faults. To achieve *isolation*, in addition to fault detection, a *set* of hypothesis tests can be used. With this approach, any fault model can naturally be used. Isolation by hypothesis testing is explored in detail in Chapter 3. Here, only a brief introduction is given.

For the set of hypothesis tests, let the different test quantities respond differently to different faults. We wish to make different test quantities sensitive to different subsets of faults. What test quantities are sensitive to what faults, can be described by the *influence structure* and the *decision structure*[1]. The basic difference between these two concepts is that the influence structure describes the influence of the faults in the ideal case, and the decision structure describes the influence in a more realistic case when also measurement noise and model errors are considered.

Four examples of influence structures are shown in Figure 1.9. A number 1 in the i:th row and the j:th column represents the fact that test quantity T_i is sensitive to fault j. A number 0 in the i:th row and the j:th column represents the fact that test quantity T_i is not sensitive to fault j. An X in the i:th row and the j:th column represents the fact that test quantity T_i is *sometimes* sensitive to fault j. For example in structure I, it can be seen that test quantity T_2 is sometimes sensitive to fault f_1, not sensitive to fault f_2, and always sensitive to fault f_3.

The isolation can ideally be performed by matching fault columns to the actual values of the test quantities. Consider for example influence structure II in Figure 1.9, and assume that test quantities T_1 and T_3 responds, but not T_2. We can then conclude that fault f_2 has occurred.

1.9 Decoupling

As was seen in the previous section, one goal of test quantity design must be to make each test quantity insensitive to a certain subset of

[1]The structured residuals method also uses influence/decision structures but under different names. Names that have been used are for example *incidence matrix* (Gertler and Singer, 1990), *residual structure*(Gertler, 1998), and *coding set* (Gertler, 1991).

I	f_1	f_2	f_3
T_1	1	1	0
T_2	X	0	1
T_3	1	1	1

II	f_1	f_2	f_3
T_1	1	1	0
T_2	1	0	1
T_3	1	1	1

III	f_1	f_2	f_3
T_1	0	X	X
T_2	X	0	X
T_3	X	X	0

IV	f_1	f_2	f_3
T_1	1	0	0
T_2	0	1	0
T_3	0	0	1

Figure 1.9: Examples of influence structures.

faults. To distinguish between these faults and the faults that the test quantity should be sensitive to, we use the terms *non-monitored* and *monitored* faults. That is, a test quantity should be highly sensitive to monitored faults but insensitive to non-monitored faults. An additional common requirement is also that test quantities should be insensitive to certain disturbances.

We say that non-monitored faults and these disturbances should be *decoupled*. When a test quantity is made *completely* insensitive to a fault or disturbance, the term *perfect decoupling* is used.

To achieve perfect decoupling, exact or very good models are needed. Also the number of faults or disturbances that can be decoupled is usually very limited. In practice, perfect decoupling is therefore often difficult to achieve. When perfect decoupling is not possible, the best one can do is to minimize the effect of these unwanted disturbances on the residual. The term *approximate decoupling* is used for this case, but is outside the scope of this text.

1.10 Analytical Redundancy

A sufficient and necessary condition to find test quantities is that the system contains *analytical redundancy*, which can be formally defined as follows:

Definition 1.1 (Analytical redundancy)**.** There exists analytical redundancy if there exists two or more different ways to determine a variable x by only using the observations $\mathbf{z}$, i.e. $x = f_1(\mathbf{z})$ and $x = f_2(\mathbf{z})$,

where $f_1(\mathbf{z}) \not\equiv f_2(\mathbf{z})$.

The meaning of the expression $f_1(\mathbf{z}) \not\equiv f_2(\mathbf{z})$ is that it is impossible to show that equality holds if no more model relations are utilized. In for example

$$2y + u = y + 2u - u + y$$

we can, without using more information, show that equality holds. On the other hand, consider

$$2y + u_1 = y + u_2$$

which can be shown to hold only if we also know some more model relations, e.g. $u_2 = y + u_1$.

A simple example of analytical redundancy is the case illustrated in Figure 1.6, i.e. it is possible to both measure and estimate an output. Another example is the following:

Example 1.3 Consider a system that can be described as

$$\begin{aligned} y &= \theta u \\ \dot{\theta} &= 0 \end{aligned}$$

where θ is an unknown constant (disturbance). The model equation $\dot{\theta} = 0$ is an easy way to analytically state that a parameter is constant. If the observations consist of one sample of y and u, we can only determine the variables y, u, and θ in one single way. Therefore analytical redundancy does not exist.

If instead, we have two samples of y and u, we get the relation

$$y(t_1) = \theta u(t_1) \tag{1.3a}$$
$$y(t_2) = \theta u(t_2) \tag{1.3b}$$

Since we now have an over-determined system of equations, analytical redundancy exists. For example, $y(t_2)$ can be determined both from the measurement and by calculating

$$y(t_2) = \frac{y(t_1)}{u(t_1)} u(t_2)$$

Since analytical redundancy exists we can try to decouple the unknown disturbance θ. An expression based on the equations (1.3), but that does not contain θ, is

$$0 = y(t_2)u(t_1) - y(t_1)u(t_2)$$

This expression can be used as a residual or test quantity, in which θ obviously has been decoupled.

Analytical redundancy exists in two forms:

- *Static Redundancy*
  The instantaneous/static relationship among sensor outputs, and actuator inputs of the system. The special case of static redundancy between outputs only, is called *direct redundancy.*
- *Temporal Redundancy*
  The relationship among histories or derivatives of sensor outputs and actuator inputs. Equations describing temporal redundancy are generally differential or difference equations.

Example 1.4 Consider a system with 3 outputs and one input:

$$\begin{aligned} y_1 &= g(u) \\ y_2 &= G(s)u \\ y_3 &= 2g(u) - G(s)u \end{aligned}$$

For this system, there is for example a static redundancy between the input u and the output y_1. In addition, temporal redundancy exists between the input u and the output y_2. Finally, by substituting the first two equations into the third, arriving at

$$y_3 = 2y_1 - y_2 \tag{1.4}$$

it can be seen that also direct redundancy is present. Each of the exemplified analytical redundancies can be used to derive residuals or test quantities. For example, residuals based on the redundancies discussed can be formed as

$$\begin{aligned} r_1 &= y_1 - g(u) \\ r_2 &= y_2 - G(s)u \\ r_3 &= y_3 - 2y_1 + y_2 \end{aligned}$$

It should be noted that if the system contains dynamics and detection time is not critical, analytical redundancy can often be approximated by static redundancy.

1.11 Engineering of Diagnosis Systems

This section aims at discussing the construction of model based diagnosis systems from an engineering, and more practical, point of view. When designing a diagnosis system for a real industrial application, additional considerations, than the ones treated in this text, must usually be taken into account. These considerations include developing-time constraints, market requirements as well as economical constraints. It is therefore not sure that these methods are always suitable for a direct implementation. However, the general concepts and ideas can certainly be useful in all diagnosis-system design.

1.11.1 Disturbances

If there would be perfect models available, we could in principle design perfectly functioning diagnosis systems, which always produce correct diagnoses. However, models used for model based diagnosis are in practice never perfect. Also there may be unknown inputs acting on the system, e.g. unmeasured and unmodeled load or friction. Finally measurement noise is normally present. All these factors can be seen as disturbances acting on the systems. These disturbances complicate the diagnosis task.

A desirable property of the diagnosis system is that it should be *robust* against disturbances, i.e. it should be able to perform the diagnosis task well in spite of the presence of disturbances. However few robust methods, useful in real applications, are available.

As mentioned in Section 1.9, it is sometimes possible to decouple disturbances in the same way as faults. However, this requires exact models of how the disturbances enters the systems, and also complicates the design of the test quantities. As with faults, disturbances can be modeled in a number of different ways. Section 2.7 includes a discussion on different principles to model influence of faults on a process. Almost all those principles can be used also for disturbance modeling. However, if little knowledge about the disturbances are available, they may

have to be modeled as arbitrary signals. As we will see later, this has implications on the lowest number of sensors the system must be equipped with.

Another way of "dealing" with disturbances is to neglect them in the basic design of the diagnosis system, and then later try to tune the diagnosis system so that the effects of the disturbances are minimized. This however requires the disturbances to be so small so that they can really be neglected.

1.11.2 A Procedure for Diagnosis System Design

An engineer working in the industry probably uses other terms and look upon the task of diagnosis somewhat differently compared to what is described in this text. However the procedure for constructing a model based diagnosis-system should approximately involve the following steps:

1. Obtain requirements on what faults that need to be diagnosed, time constraints such as detection time and isolation time, any requirements regarding fault identification.
2. Study and acquire knowledge about the system and particularly the faults that need to be diagnosed.
3. Build a model of the process in the fault free case. This step often consists of three parts: selection of model structure, parameter identification, and model validation.
4. Build fault models, i.e. models of how the faults influence the system.
5. By using the model, including the fault models, design test quantities to be used in the hypothesis tests. These test quantities must be chosen so that the isolation requirements are satisfied.
6. For each test quantity design a hypothesis test which at least includes a thresholding of the test quantity. The threshold needs to be tuned to get a good compromise between false alarms and missed detections.
7. Test the diagnosis system in simulation and if possible in reality. If the performance is not satisfactory, the hypothesis tests and possibly also the models need to be refined.

8. Do a final implementation of the diagnosis system.

Step 1 and step 2 can only be performed by acquiring information from experiments, long-time experience, and domain experts. There are systematic methods available: FMEA (Failure Mode Effect Analysis) and FTA (Fault Tree Analysis). Books discussing FMEA or FTA are (Bergman and Klefsjö, 1996; McDermott et al., 1996; Palady, 1995; Roberts, 1992).

Step 3, the model building, is probably the best documented of all steps. As noted before in this text, it is probably the major part of the work. There is generally available literature on modeling of specific classes of systems and also on general model building, e.g. in system identification literature (Ljung, 1999).

Step 4, constructing the fault models, requires as detailed knowledge as possible about the faults. In some applications, it is possible to implement the faults and then identify the fault model. However, for many applications, it is highly undesirable or impossible to implement faults, e.g. because of the risk for severe damage. Sometimes, useful data exists from previous real faults. If a detailed fault model can not be obtained, the engineer may have to use a very general fault model, e.g. the arbitrary fault signal discussed in Section 2.7.1. Even though this will result in a correct fault model, it may imply that isolation becomes difficult.

Step 5 and 6 are the main topic of this course. Many methods for especially step 5 have been developed. However there are still very few books on the subject (see Section 1.12) but many research papers have been written.

Step 7 is application dependent. In some applications, only simulations can be allowed, while in other, it is possible to do some practical experiments.

Little literature about step 8 is available. There is however much written about implementation of control systems, which can be partly applied also for implementation of diagnosis systems.

1.12 Bibliography

As said above, few books have so far been published about model based diagnosis. From the perspective of automatic control, some books available are (Natke and Cempel, 1997; Mangoubi, 1998; Sohlberg,

1998; Gertler, 1998; Chen and Patton, 1999; Russell and Braatz, 2000). Related books but with a slightly different focus, more on signal processing, are (Basseville and Nikiforov, 1993; Gustafsson, 2000). Also some edited books that have been published are (Patton et al., 1989; Nijmeijer and Fossen, 1999; Patton et al., 2000). In the area of AI, there is an edited book (Hamscher et al., 1992).

Chapter 2
Principles of Model Based Diagnosis

When constructing a model-based diagnosis system, a model of the process is needed. This model must contain the process behavior in the fault free case but also include a description of how and where different faults affect the process. In this chapter we describe the concept of formal models from the view of model based diagnosis. Further, we formally define the central concept *diagnosis*.

2.1 Models

A *model* is some description of a system. To be useful for mathematical tools or computers, the model must be described in a formal manner by using a modeling language. Here the basis is *first order logic* and we will briefly introduce the modeling language elements that are needed to, in the next section, introduce models for diagnosis. Step by step we will introduce propositions, logical connectives, predicates, functions, variables, and sentence.

For an example, consider the system bicycle. To describe that the bicycle is red we can introduce a *proposition*

BikeColorIsRed

Propositions can take values TRUE or FALSE, so if the bicycle is red, the proposition BikeColorIsRed is TRUE. If it is instead blue, the proposition BikeColorIsRed is FALSE.

Propositions can be combined by using logical connectives. There are five that will be considered in this text:

$\neg$	not
$\wedge$	and
$\vee$	or
$\rightarrow$	implies
$\leftrightarrow$	if-and-only-if

For example, to describe that a bicycle is red and a car is blue we can write BikeColorIsRed $\wedge$ CarColorIsBlue. To describe the fact that a blue car is a colored car we write CarColorIsBlue $\rightarrow$ CarColored. Note that CarColorIsBlue $\rightarrow$ CarColored is equivalent to that $\neg$CarColorIsBlue $\vee$ CarColored.

Instead of a proposition we can use a *predicate*, in this case Red$(\cdot)$, to write

$$\text{Red(Bicycle)} \tag{2.1}$$

The predicate Red$(\cdot)$ has *arity* one, or equivalently is *unary*, meaning that it takes one argument. Predicates of arity one are used to describe a property of an object.

In addition to describing an object using predicates of arity one, we can describe relationships between objects. This is done using predicates of arity two, also called *binary* predicates. For example we can write

$$\text{SameColor(Bicycle,Car)} \tag{2.2}$$

to describe that the objects Bicycle and Car have the same color.

In a model we could represent every fact by predicates of the types (2.1) and (2.2). However more flexibility is obtained if we use also *functions*. To represent properties of objects we use functions taking as argument an object and returning some object. For example the color of an object can be represented by the function Color$(\cdot)$. By combining this with the binary predicate Equal$(\cdot,\cdot)$, or $=$ for short, we can write

$$\text{Color(Bicycle)} = \text{Red}$$

instead of Red(Bicycle). This adds flexibility because the same predicate Color$(\cdot)$, can together with equality, be used to describe that the color of some car is blue, i.e. Color(Car) $=$ Blue.

For a more compact and flexible modeling language, we use also *variables* to refer to objects. For example, the variable c_{Bike} can be used instead of Color(Bicycle). Then we can write $c_{\text{Bike}} = \text{Red}$ instead of 2.1. The combination of the equality predicate, a variable, and an object, such as $c_{\text{Bike}} = \text{Red}$, is called *assignment*. The different objects that we can assign to a variable are called *values*. The set of possible values of a variable is called the *domain*. For example the domain of c_{Bike} could be $\{\text{Red}, \text{Black}, \text{Blue}\}$. Elements of a domain are implicitly assumed to be distinct, meaning that no two values are equal. In a model we can express the fact that a variable has a specific domain with the binary predicate $\in$. For example, to express the domain of the variable c_{Bike} we write

$$c_{\text{Bike}} \in \{\text{Red}, \text{Black}, \text{Blue}\}$$

We see that the first argument to the predicate is a variable, and the second is the domain set.

We can now combine variables with predicates and connectives to express more complicated things. For example, assume that we want to express the fact that if the pedals are rotating forward, then the back wheel is also rotating forward. We can for this introduce the variable r_{Ped} to represent the rotational state of the pedals and r_{BW} to represent rotational state of the back wheel. Then we write

$$r_{\text{Ped}} = \text{Forward} \rightarrow r_{\text{BW}} = \text{Forward}$$

One example of how this expression can be used is if we find that the back wheel is not rotating forward, we know that the pedals cannot be rotating forward.

By combining propositions, logical connectives, predicates, and variables into a valid expression we obtain what is called a *sentence*. A set of sentences can then be used to represent all knowledge we want to consider. In particular, a formal *model* of a system is a set of sentences.

2.2 Models for Diagnosis

A model, i.e. a set of sentences, can in general be used for any purpose, such as simulation or analysis of systems. Our particular purpose is diagnosis, and in this context it is useful to introduce some more

specialized objects and variables. A model to be used for diagnosis will be called a *diagnostic model*.

In diagnosis, the purpose is to use the model to reason about the presence of different faults, given some observations. Therefore, we will now, in the upcoming two sections, discuss how faults and observations are related to the model and the modeling language.

2.2.1 Faults and Components

To describe faults, a convention in the area of model based diagnosis is to introduce the concept of *component*. The idea is that a component is something that can brake; something that can be faulty or non-faulty. To represent the fact that components are faulty or non-faulty we can use propositions, unary predicates, or variable assignments. It may seem superfluous to have three ways to state that a component is faulty. The reason that we here introduce all three of them is that it turns out that depending on the context; all three of them are useful.

For the bicycle example, we could consider the chain to be a component. To describe that the chain is broken we can use a proposition,

$$\text{ChainBroken}$$

or a unary predicate,

$$\text{Broken}(\text{Chain}) \tag{2.3}$$

To describe that the chain is connected (not broken), we can without introducing any new predicates, write $\neg$ChainBroken or equivalently $\neg$Broken(Chain). Another alternative is to introduce new predicates and write

$$\text{ChainConnected}$$

or equivalently

$$\text{Connected}(\text{chain}) \tag{2.4}$$

This alternative has the disadvantage that we need to add further knowledge to represent the fact that the chain cannot be broken and connected at the same time. For example, if the representation based on (2.3) and (2.4) is used, we need to add into the model, the sentence

$$\text{Connected}(\text{chain}) \leftrightarrow \neg\text{Broken}(\text{Chain}) \tag{2.5}$$

Behavioral Modes

It is quite common to assume that each component is either non-faulty or faulty. However in general, a component can brake or become faulty in more than one way. In this case it is convenient to talk about the *behavioral mode* of a component instead of just saying that it is faulty or non-faulty. To emphasize that some behavioral modes represent the fact that a component is faulty, they are often called *fault modes*.

For an example consider digital inverters. We will assume that an inverter has the following behavioral modes:

behavioral mode	predicate symbol	physical meaning
no fault	OK	$(output = 0) \equiv (input = 1)$
stuck at 0	$SA0$	$output = 0$
shorted	$SHORT$	$output = input$
unknown	U	unknown input-output relation

Let I denote a specific component of the type inverter. Now we can generalize the idea of representing faults as unary predicates, as in (2.3), to also handle more than two behavioral modes. To describe that the inverter I is not faulty, we can use the unary predicate $OK(I)$. Similarly when I is stuck at 0, we write $SA0(I)$, and when shorted, we write $SHORT(I)$. When we want to describe that the inverter I has some fault, but it is unknown which, we can write $U(I)$. If we only want to say that I is faulty but do not want to specify the exact fault mode, we can write $\neg OK(I)$.

To represent behavioral modes, we can instead of using unary predicates, for each component introduce a variable to represent the behavioral mode of that component. For example, let b_{Chain} be a variable representing the behavioral mode of the chain. Then we can write

$$b_{\text{Chain}} = \text{Broken} \quad \text{and} \quad b_{\text{Chain}} = \text{Connected}$$

The domain of the variable b_{Chain} could in this example be $\{\text{Broken}, \text{Connected}\}$.

An advantage of representing faults of components with variables assigned to behavioral modes is that we automatically obtain the property that a component can not be in two modes at the same time, for example that the chain cannot be broken and connected at the

same time. That is, we do not need to add any extra sentences such as (2.5) into the model.

Note that in the example of (2.3) and (2.4), where we assumed only two behavioral modes of the component, it was sufficient with one sentence of the type (2.5). However, when there are more behavioral modes per component, more sentences of the type (2.5) are needed. For example, for the inverter described above, with four behavioral modes, six sentences of the type (2.5) would be needed.

As was stated above, in the case we define only one proposition or one unary predicate to describe the behavioral mode of a component, e.g. ChainBroken or Broken(Chain), we do not need a sentence like (2.5). The reason is that logic has built in, the property that predicates (and propositions) cannot be TRUE and FALSE simultaneously. Note however that this only works as long as there are only two behavioral modes per component.

System Behavioral Modes

Above we have introduced *behavioral mode* of a single component. Often it is convenient to refer to all behavioral modes of all components in a system. For this we introduce *system behavioral modes*. Sometimes, when we want to emphasize the fact that we refer to the behavioral mode of a component and not the whole system, we will write *component behavioral mode*.

For an example, suppose that we have a system consisting of two inverters I_1 and I_2. To express the system behavioral mode where both are non-faulty, we write $OK(I_1) \wedge OK(I_2)$. To express the system behavioral mode where I_1 is shorted and I_2 is non-faulty, we write $SHORT(I_1) \wedge OK(I_2)$.

For another example of system behavioral modes, consider a system consisting of a gas tank with potential leakages. The tank is also equipped with a pressure sensor. The tank and the sensor are considered to be the diagnosed components. We decide that all tank leakages, regardless of their area, belong to the same behavioral mode "leakage". We also decide that all faults in the pressure sensor belong to the behavioral mode "pressure sensor fault". Each diagnosed component also has the "no-fault" mode. In the example we have two possible behavioral modes per component. This means that we have in total 4

different system behavioral modes:

no fault	**NF**
pressure sensor fault	**PSF**
leakage	**L**
pressure sensor fault and leakage	**PSF**&**L**

(2.6)

As seen in the table, each system behavioral mode has been associated with one abbreviation, obtained by combining abbreviations of component behavioral modes. Throughout the text we will use the convention to write abbreviations of system behavioral modes with boldface letters and **NF** is used to refer to the system behavioral mode where all diagnosed components are in the mode no-fault.

The set of all behavioral modes will be denoted Ω, and in the example, $\Omega = \{\mathbf{NF}, \mathbf{PSF}, \mathbf{L}, \mathbf{PSF}\&\mathbf{L}\}$. Note that a consequence of the definitions is that only one of the system behavioral modes in Ω can be present at the same time.

Let p be the number of components and n_i the number of different behavioral modes for the i:th component. The total number of possible system behavioral-modes, and the cardinality of the set Ω, is then $\prod_{i=1}^{p} n_i$.

Sometimes it will be convenient to represent system behavioral modes using tuples instead of abbreviations as in (2.6). For instance in cases where the component behavioral modes do not have unique abbreviations, it is not possible to represent system behavioral modes using the principle illustrated in (2.6). When using tuples we assume that the components are ordered. For example if we order the components in the water tank example according to (Tank, PressureSensor), the system behavioral modes become as follows.

(NF,NF)
(NF,PSF)
(L,NF)
(L,PSF)

2.2.2 Observations

In addition to faults and components, also the observations are central for the diagnosis task. As described in Chapter 1, the idea of observations is that a human operator, or an automatic diagnosis system, observes or measures something on the system. Then, from the

knowledge of observations together with the model, diagnosis can be performed.

We have above introduced the concept of models formally, and discussed how faults can be represented in the modeling language. We can describe also observations in the same language. Even though there are several possibilities, as for the representation of faults, the most common is to represent an observation as a assignment of a value to a variable contained in the model. Usually only parts of the system are possible to observe. In the context of our model this means that only some variables can be assigned values through observations. These variables are often called *observables*.

As an example, consider a person that observes that the back wheel of the bicycle is not rotating, i.e. is at standstill. If the person wants to use the model to perform diagnosis, he must consider this knowledge together with the model. If we assume that the model contains a variable r_{BW} representing the rotational speed of the back wheel, he can do that by introducing a sentence with the assignment

$$r_{\mathrm{BW}} = 0$$

In the case of binary observations we could use also a proposition to represent each observation. For example the observation that a lamp is lighted or not can be represented with the proposition Light.

2.3 Diagnosis

In this section we will define what we mean by a *diagnosis*. In this, we follow the ideas of so called *consistency based diagnosis* which is a diagnosis approach from the field of AI.

First we need the notion of a *consistent* model. A model contains a number of variables and possibly also a number of propositions. Each variable can take a value in its domain. Each proposition is either TRUE or FALSE. If we assign values to the variables and propositions, without regarding the model, it is likely that we obtain a contradiction in some of the model sentences. If we instead assign values by carefully studying the model, it is likely that we can find values such that a contradiction can be avoided.

For example, consider the model

$$r_{\mathrm{Ped}} = \mathrm{Forward} \rightarrow r_{\mathrm{BW}} = \mathrm{Forward}$$

If we assign the value Forward to both variables r_{Ped} and r_{BW} we get no contradiction. If, on the other hand we assign Forward to r_{Ped} and Backward to r_{BW} we would get a contradiction.

We say that a model is *consistent* if we can assign a truth-value to each proposition and a value to each variable within the domain of that variable, such that the model is satisfied.

For example, the model

$$r_{\text{Ped}} = \text{Forward} \rightarrow r_{\text{BW}} = \text{Forward}$$
$$r_{\text{Ped}} = \text{Forward}$$

is consistent since we can assign the value Forward to r_{BW} and obtain no contradiction.

If we expand the model as

$$r_{\text{Ped}} = \text{Forward} \rightarrow r_{\text{BW}} = \text{Forward} \tag{2.7a}$$
$$r_{\text{Ped}} = \text{Forward} \tag{2.7b}$$
$$r_{\text{BW}} = \text{Backward} \tag{2.7c}$$

we can never find values to the two variables r_{ped} and r_{BW} such that all three model sentences are TRUE. Thus, the model (2.7) is inconsistent.

We are now ready to formally define the term diagnosis. In accordance with the discussion in this chapter, we consider the model, the observations, and behavioral mode assignments, to be sets of sentences.

Definition 2.1 (Diagnosis)**.** Given a diagnostic model $\mathcal{M}$ and observations $\mathcal{O}$, a *diagnosis* is an assignment $\mathcal{D}$ of a behavioral mode to each diagnosed component such that

$$\mathcal{M} \bigcup \mathcal{O} \bigcup \mathcal{D}$$

is consistent[1].

Note that this definition of diagnosis is in agreement with the task of a diagnosis system as formulated in Section 1.3. Note also that for the same observations, there can be several diagnoses, i.e. there can be several possible explanations for what we are observing.

In the definition of diagnosis we used the expression "assignments of behavioral modes to each component". Equivalent is to use the

[1]Here *consistent* means the same thing a *satisfiable*.

expression system behavioral mode. Furthermore, in the area of AI and consistency based diagnosis, and especially in the context of algorithms for computing diagnoses, a set of assignments of a behavioral mode to each component, is called a *candidate*. Thus a diagnosis could also be defined as a system behavioral mode, or candidate, consistent with the model and the observations.

2.3.1 Examples

To get a feeling for the terminology and the concepts introduced in the previous section we will now study two examples. We will create models that contain a mixture of behavioral modes, variables, and observables.

Consider first the subsystem of a bicycle consisting of pedals, a chain, and a back wheel. In a simplified case we can assume that in this subsystem the only part that can break is the chain. Therefore the chain is our only diagnosed component. Its behavioral modes are Broken and Connected. By knowledge of how a bicycle is constructed we know that if we move the pedals forward and the chain is connected, then the back wheel will also move forward. All this knowledge can formally be represented in a model as follows.

$$b_{\text{Chain}} \in \{\text{Broken}, \text{Connected}\}$$
$$r_{\text{Ped}} \in \{\text{Backward}, \text{Standstill}, \text{Forward}\}$$
$$r_{\text{BW}} \in \{\text{Backward}, \text{Standstill}, \text{Forward}\}$$
$$b_{\text{Chain}} = \text{Connected} \wedge r_{\text{Ped}} = \text{Forward} \rightarrow r_{\text{BW}} = \text{Forward}$$

Note that this model has three variables: the behavioral mode b_{Chain}, and the observables r_{Ped} and r_{BW}.

If we observe that the pedals are rotating forward and the back wheel is not, we can introduce this observational knowledge as the additional sentences

$$r_{\text{Ped}} = \text{Forward}$$
$$r_{\text{BW}} = \text{Standstill}$$

Then there is only one possible assignment to the three variables:

$$b_{\text{Chain}} = \text{Broken}$$
$$r_{\text{Ped}} = \text{Forward}$$
$$r_{\text{BW}} = \text{Standstill}$$

All other assignments result in a contradiction. Thus, the only diagnosis is $b_{\text{Chain}} = \text{Broken}$.

Next we consider a larger system consisting of a battery, a switch, a lamp, and connecting cables. In this system we define three diagnosed components: the battery, the switch, and the lamp. Then we have the following component variables and their corresponding domains:

$$b_{\text{Battery}} \in \{\text{Charged}, \text{Drained}\}$$
$$b_{\text{Switch}} \in \{\text{OK}, \text{StuckClosed}, \text{StuckOpen}\}$$
$$b_{\text{Lamp}} \in \{\text{OK}, \text{Broken}\}$$

In addition to these variables we also have the observable variable switchBut, the position of the switch button, with its domain:

$$\text{switchBut} \in \{\text{OFF}, \text{ON}\}$$

Another observable is the light of the lamp that we choose to represent with the proposition Light.

To represent the fact that we have voltage or not at the switch and at the lamp, we use the propositions SwitchVolt and LampVolt respectively. Finally, to describe the functionality of the circuit we add these model sentences:

$$\text{switchBut} \in \{OFF, ON\}$$
$$b_{\text{Battery}} = \text{Charged} \rightarrow \text{SwitchVolt}$$
$$b_{\text{Switch}} = \text{OK} \rightarrow (\text{switchBut} = \text{ON} \wedge \text{SwitchVolt} \leftrightarrow \text{LampVolt})$$
$$b_{\text{Switch}} = \text{StuckClosed} \rightarrow \text{LampVolt}$$
$$b_{\text{Switch}} = \text{StuckOpen} \rightarrow \neg\text{LampVolt}$$
$$b_{\text{Lamp}} = \text{OK} \rightarrow (\text{LampVolt} \leftrightarrow \text{Light})$$
$$b_{\text{Lamp}} = \text{Broken} \rightarrow \neg\text{Light}$$

Note here that there is no model sentence for the case of drained battery. The idea is that if the battery is charged, we know that it has the capability to deliver voltage to the switch. However, if the battery is drained, we do not know the degree of drainage so it is unsure if the battery can deliver voltage or not. Therefore the behavior of the battery in the behavioral mode Drained is not possible to predict. This is the reason why we by purpose have omitted to add a sentence describing the battery behavior when it is drained.

Then assume that we observe that the switch button is on and there is no light. This is represented by introducing the observational sentences:

$$\begin{aligned} &\text{switchBut} = \text{ON} \\ &\neg\text{Light} \end{aligned}$$

The different values of propositions and variables, consistent with the model and the observations, are:

b_{Battery}	b_{Switch}	b_{Lamp}	SwitchVolt	switchBut	LampVolt	Light
Charged	OK	Broken	TRUE	ON	TRUE	FALSE
Charged	StuckOpen	OK	FALSE	ON	FALSE	FALSE
Drained	OK	OK	FALSE	ON	FALSE	FALSE
Charged	StuckOpen	Broken	TRUE	ON	FALSE	FALSE
Charged	StuckClosed	Broken	TRUE	ON	TRUE	FALSE
Drained	StuckOpen	OK	TRUE	ON	FALSE	FALSE
Drained	StuckOpen	OK	FALSE	ON	FALSE	FALSE
Drained	StuckClosed	OK	FALSE	ON	FALSE	FALSE
Drained	OK	Broken	FALSE	ON	FALSE	FALSE
Drained	OK	Broken	TRUE	ON	TRUE	FALSE
Drained	StuckOpen	Broken	FALSE	ON	FALSE	FALSE
Drained	StuckOpen	Broken	TRUE	ON	FALSE	FALSE
Drained	StuckClosed	Broken	FALSE	ON	FALSE	FALSE
Drained	StuckClosed	Broken	TRUE	ON	TRUE	FALSE

In this table we see that for the assignment

$$b_{\text{Battery}} = \text{charged} \tag{2.8a}$$
$$b_{\text{Switch}} = \text{OK} \tag{2.8b}$$
$$b_{\text{Lamp}} = \text{broken} \tag{2.8c}$$

it is possible to find values of the other variables and propositions such that the model is consistent with the observations. Therefore (2.8) is a diagnosis.

The total set of diagnoses can be represented in a table as:

b_{Battery}	b_{Switch}	b_{Lamp}
Charged	OK	Broken
Charged	StuckOpen	OK
Drained	OK	OK
Charged	StuckOpen	Broken
Charged	StuckClosed	Broken
Drained	StuckOpen	OK
Drained	StuckClosed	OK
Drained	OK	Broken
Drained	StuckOpen	Broken
Drained	StuckClosed	Broken

Note that for some diagnoses there are more than one possible set of values of the other variables and propositions. Thus the number of diagnoses is less than the number of assignments.

2.4 Characterization of Diagnoses

As stated in the previous section, the task of diagnosis in the consistency based framework is to, given model and observations, find the set of diagnoses, i.e. behavioral mode assignments that are consistent with the observations. In this section we will investigate how to represent or *characterize* this set of diagnoses. The set is, except for trivial systems, very large. Humans or computers performing diagnosis could therefore run into problems with processing power and memory if the set is represented by listing each of its elements. Therefore, it is especially interesting to investigate characterizations that are efficient in terms of memory and processing power requirements.

One alternative is to characterize a set of diagnoses as a *disjunction*[2] of the diagnoses itself.

Example 2.1 Let us return to Example 1.1 from Chapter 1. If we observe that the desired switch position is "closed" and the lamp is "lit". Then we saw that there are two diagnoses: "no faults" and "S stuck closed". Assume that we use the symbol OK to denote no fault, SC to denote stuck closed, and SO to denote stuck open. Then the

[2] meaning *or*

two diagnoses can be written $OK(S) \wedge OK(L)$ and $SC(S) \wedge OK(L)$. These diagnoses can be characterized by the logical expression

$$\Big(OK(S) \wedge OK(L)\Big) \vee \Big(SC(S) \wedge OK(L)\Big) \tag{2.9}$$

Note that $SO(S) \wedge OK(L)$ is a candidate but not a diagnosis since it is not consistent with the imagined model. The model is here not formally defined, but for any sensible model it would not be consistent that the switch is stuck open and the lamp is lit.

Except for representing candidates and diagnoses as $OK(S) \wedge OK(L)$ and $SC(S) \wedge OK(L)$ etc., it is quite common to use a set representation. For example, the candidate $SC(S) \wedge OK(L)$ would be written $\{SC(S), OK(L)\}$. A diagnosis statement can then be represented as a set of sets, e.g. the diagnosis statement (2.9) would be written $\{\{OK(S), OK(L)\}, \{SC(S), OK(L)\}\}$. Note that to represent a candidate with a set, such as $\{SC(S), OK(L)\}$, can be seen as a set of sentences, where each sentence is an unary predicate. Thus, this representation is perfectly legal to use directly in the definition of diagnosis, i.e. Definition 2.1.

Often not all components are mentioned in the set. Components not mentioned are assumed to be non-faulty. For example $\{SC(S)\}$ means the same as $\{SC(S), OK(L)\}$. With this principle, consider the first diagnosis statement in Table 1.1 and let BR denote broken. Then this diagnosis statement can be written as

$$\{\{\},\ \{SO(S)\},\ \{BR(L)\},\ \{SO(S),\ BR(L)\},\ \{SC(S),\ BR(L)\}\} \tag{2.10}$$

2.4.1 Minimal Diagnoses

The set notation will prove to be practical in the context of so called *minimal diagnoses*. In principal, a diagnosis is said to be minimal if there are no "simpler" diagnoses. If the set representation is used, with non-faulty components not included, a minimal diagnosis can formally be defined as follows:

Definition 2.2 (Minimal Diagnosis)**.** A diagnosis $\mathcal{D}$ is minimal if no $\mathcal{D}' \subset \mathcal{D}$ (proper subset) is a diagnosis.

For example, in the diagnosis statement (2.10), the only minimal diagnosis is $\{\}$. Another example is the third diagnosis statement in

Table 1.1, i.e. for the observation switch "closed", lamp "not lit". In this case, there are two minimal diagnoses: $\{SO(S)\}$ and $\{BR(L)\}$.

The reason for the interest in minimal diagnoses comes from the fact that minimal diagnoses are under some conditions a powerful characterization of *all* diagnoses:

Definition 2.3 (Minimal Diagnosis Hypothesis)**.** The *Minimal Diagnosis Hypothesis* holds if every superset of a minimal diagnosis is also a diagnosis.

The *Minimal Diagnosis Hypothesis* does not always hold, and it is difficult to formulate an exact criterion when it holds (de Kleer et al., 1992). However, a sufficient condition for the *Minimal Diagnosis Hypothesis* to hold is that each component has only two behavioral modes: a no-fault mode and a faulty mode, and that the faulty mode has no behavior specified in the model. The minimal diagnosis hypothesis is further discussed in Section 3.5.2 in the context of computing diagnosis.

For illustration, consider again the circuit model example from Section 2.3.1:

$$\begin{aligned}
&\text{switchBut} \in \{\text{OFF}, \text{ON}\}\\
&b_{\text{Battery}} = \text{Charged} \rightarrow \text{SwitchVolt}\\
&b_{\text{Switch}} = \text{OK} \rightarrow (\text{switchBut} = \text{ON} \wedge \text{SwitchVolt} \leftrightarrow \text{LampVolt})\\
&b_{\text{Switch}} = \text{StuckClosed} \rightarrow \text{LampVolt}\\
&b_{\text{Switch}} = \text{StuckOpen} \rightarrow \neg\text{LampVolt}\\
&b_{\text{Lamp}} = \text{OK} \rightarrow (\text{LampVolt} \leftrightarrow \text{Light})\\
&b_{\text{Lamp}} = \text{Broken} \rightarrow \neg\text{Light}
\end{aligned}$$

Here we see that there are model sentences describing the behavior of the switch in the modes StuckClosed and StuckOpen. Also there is a model sentence describing the behavior of the lamp in the mode Broken. Assume that we have observed that the switch button is OFF but the lamp is lit. This means that one possible diagnosis is

$$b_{\text{Battery}} = \text{Charged} \wedge b_{\text{Switch}} = \text{StuckClosed} \wedge b_{\text{Lamp}} = \text{OK}$$

In the set notation this diagnosis could be written as

$$\{\text{StuckClosed(Switch)}\}$$

If the minimal diagnosis hypothesis would hold, also

$$\{\text{StuckClosed}(\text{Switch}), \text{Broken}(\text{Lamp})\}$$

would be a diagnosis, which is clearly false since the model contains a sentence describing the fact that if the lamp is broken it is not lit.

2.5 Continuous and Dynamic Models

So far we have described all concepts in the scope of discrete and static models. This means that all variables have been assumed to have discrete and finite domains and no notion of time has been included in the models. Many physical systems are by nature dynamic and are naturally modeled by variables with continuous domains. Thus, the behavior of the system is not a direct consequence of external stimuli but also depends on internal states and changes over time. Models of such systems, which include the notion of time and derivatives, are extensively studied in e.g. the area of automatic control and signal processing.

First, the extension to consider variables with continuous domains is trivial. For example, assume we have a variable x with the domain all real numbers. In the model we write

$$x \in \mathbb{R}$$

Assignments of continuous variables are handled as for discrete variables. Next, the notion of time means that variables might change values over time. If we have two real variables x and z, together with a relation $z = x$, and want to express that the relation holds in a time interval Δ we can write

$$\forall t \in \Delta : x(t) \in \mathbb{R} \tag{2.11a}$$
$$\forall t \in \Delta : z(t) \in \mathbb{R} \tag{2.11b}$$
$$\forall t \in \Delta : z(t) = x(t) \tag{2.11c}$$

Note that the time interval can be $\Delta =]-\infty, \infty[$.

To be able to handle dynamics, we also need to include differentiated variables in the model. If we have a relation $y = \dot{x}$, valid in a time

interval Δ, this can be expressed as

$$\forall t \in \Delta : x(t) \in \mathbb{R} \tag{2.12a}$$
$$\forall t \in \Delta : y(t) \in \mathbb{R} \tag{2.12b}$$
$$\forall t \in \Delta : y(t) = \frac{dx(t)}{dt} \tag{2.12c}$$

As seen the notation becomes a bit clumsy, because of the need to include $\forall t \in \Delta$ everywhere. A similar notion can be described for systems that work in discrete time also.

If we want to investigate consistency of a model of the type (2.11) we need to assign a value to z and x at each time instance. So what we are really interested in is z and x seen as functions of time, or *trajectories*. By viewing z and x as variables taking values in a domain of all real functions over the time interval Δ, we get a much cleaner notation. Let $\mathcal{D}(\Delta, \mathbb{R})$ denote the set of all real functions defined on Δ. Instead of (2.11) we write simply

$$\begin{aligned} x &\in \mathcal{D}(\Delta, \mathbb{R}) \\ z &\in \mathcal{D}(\Delta, \mathbb{R}) \\ z &= x \end{aligned}$$

Instead of (2.12) we can write

$$\begin{aligned} x &\in \mathcal{D}(\Delta, \mathbb{R}) \\ y &\in \mathcal{D}(\Delta, \mathbb{R}) \\ y &= \dot{x} \end{aligned}$$

The expressiveness of dynamic models cannot only be used to describe that values varies over time, but also to describe that values do not vary over time. For example, consider

$$\begin{aligned} c &\in \mathcal{D}(\Delta, \mathbb{R}) \\ 0 &= \dot{c}, \end{aligned}$$

which means that the trajectory of the variable c is constant.

2.5.1 Diagnosis in Dynamic Models

Above we described how variables could represent trajectories. For sake of simplicity, behavioral mode variables in dynamic models are assumed

to be constant during the time interval considered. For example this means that, in the simplified presentation, that either a fault is present during the whole time interval or it is not present at any time during the interval.

An observational variable would typically change over time. If we consider an observational variable to be a snapshot of some part of the system for one certain time instance, we could compute a diagnosis for each such snapshot. However, we would not take advantage of the knowledge that the model is persistent over time, i.e. it does not change. For example, consider a sensor system for measuring the ambient air pressure, denoted with the variable p_a. If the sensor is OK, a variable x becomes equal to the air pressure. The time scale in this example is considered to be seconds, which means that a sensor value is obtained once every second . Using this time scale, the ambient air pressure can be assumed to be constant. Thus we have the system model

$$
\begin{aligned}
&\text{sensor} \in \{\text{OK}, \text{Bad}\} \\
&p_a \in \mathcal{D}(\Delta, \mathbb{R}) \\
&x \in \mathcal{D}(\Delta, \mathbb{R}) \\
&\dot{p}_a = 0 \\
&\text{sensor} = \text{OK} \rightarrow x = p_a
\end{aligned}
$$

If we use a snapshot observation we may observe that $x = 101kPa$. One second later we observe that $x = 96kPa$. The model with sensor assigned to OK is consistent with both these individual observations. Thus, sensor = OK is a diagnosis for each of these observations. However, we know that the observation $x = 101kPa$ followed by $x = 96kPa$ one second later implies that there must be some fault in the sensor.

To utilize the fact that our model is persistent over time we need to handle observations that are collected, not only in a snapshot, but over a time window. In the example, we need to handle the two observations simultaneously. This can be done by including the explicit time of the observations. Then the model extended with the observations results

in the set of sentences

$$
\begin{aligned}
&\text{sensor} \in \{\text{OK}, \text{Bad}\}\\
&p_a \in \mathcal{D}(\Delta, \mathbb{R})\\
&x \in \mathcal{D}(\Delta, \mathbb{R})\\
&\dot{p}_a = 0\\
&\text{sensor} = \text{OK} \rightarrow x = p_a\\
&x(t_0) = 101kPa\\
&x(t_1) = 96kPa
\end{aligned}
$$

The only diagnosis for this model and observations is sensor = Bad.

In the area of automatic control, state-space equations are typically used to model dynamic systems. These can directly be used in our modelling language. That is, a state space equation can be written as

$$
\begin{aligned}
&x \in \mathcal{D}(\Delta, \mathbb{R}^n)\\
&u \in \mathcal{D}(\Delta, \mathbb{R}^k)\\
&y \in \mathcal{D}(\Delta, \mathbb{R}^m)\\
&\dot{x} = f(x, u)\\
&y = g(x, u)
\end{aligned}
$$

The variable u is typically called input and y output of the system. However, this distinction is of no importance from a diagnosis perspective; what is important is instead that both u and y are typically known and therefore become observational variables.

Consider a cart of mass 2 kg moving on a rail. An unknown but constant force represented by a variable F is affecting the cart and the velocity is represented by a variable v. The speed of the cart is measured with a speed sensor producing a value y that equals v if the sensor is fault free. In a state-space model this knowledge can be

represented as

$$\begin{aligned}
&\text{sensor} \in \{\text{OK}, \text{Bad}\}\\
&v \in \mathcal{D}(\Delta, \mathbb{R})\\
&F \in \mathcal{D}(\Delta, \mathbb{R})\\
&y \in \mathcal{D}(\Delta, \mathbb{R})\\
&\dot{F} = 0\\
&\dot{v} = 0.5F\\
&\text{sensor} = \text{OK} \rightarrow y = v
\end{aligned}$$

Now assume that we measure the speed to be 5 km/h at time 0. This observation is added so the total knowledge becomes

$$\begin{aligned}
&\text{sensor} \in \{\text{OK}, \text{Bad}\}\\
&v \in \mathcal{D}(\Delta, \mathbb{R})\\
&F \in \mathcal{D}(\Delta, \mathbb{R})\\
&y \in \mathcal{D}(\Delta, \mathbb{R})\\
&\dot{F} = 0\\
&\dot{v} = 0.5F\\
&\text{sensor} = \text{OK} \rightarrow y = v\\
&y(0) = 5
\end{aligned}$$

For this set of sentences we can assign y to any trajectory that passes 5 at time instance $t = 0$, for example the constant trajectory $y \equiv 5$. Further assign v to the same constant trajectory, assign F to the constant trajectory $y \equiv 0$, and let sensor = OK. These assignments make all model sentences true and thus, sensor = OK is a diagnosis. Note that the same assignments to y, v, and F, combined with sensor = Bad is also consistent. Thus sensor = Bad is also a diagnosis. In fact, as long as we observe the speed at only one or two time instances both sensor = OK and sensor = Bad will always be diagnoses.

If we measure the speed at three time instances, say $t = 0$, $t = 1$, and $t = 2$ then we will be able to actually perform diagnosis. Assume that we measure the speed to be 5 km/h at time $t = 0$, 6 km/h at time $t = 1$, and 7 km/h at time $t = 2$. An assignment of the variables y, v, F, and *sensor*, consistent with the model and observations, is $y(t) = t + 5$, $v(t) = t + 5$, $F(t) = 2$, sensor = Bad. The same assignments of y, v,

and F is also consistent with sensor = OK. Thus, we have the two diagnoses sensor = Bad and sensor = OK.

If we measure the speed at the same three time instances, to be 5 km/h at time $t = 0$, 7 km/h at time $t = 1$, and 7 km/h at time $t = 2$, it is not possible to find assignments of y, v, and F that are consistent with sensor = OK. However we can find an assignment of all variables consistent with sensor = Bad. Thus, sensor = Bad is the only diagnosis.

In the models described in this chapter so far, we have explicitly included the domain as a part of the model. From now on, the domain is often obvious from the context and therefore not written out explicitly.

2.6 Detectability and Isolability

Given a model, an important property is if it is at all possible to use the model to detect and isolate different faults. This is called detectability and isolability respectively and is a property of the model.

Let $\mathbf{z}$ be the tuple of observables. Then we introduce the concept of *observation set* as follows.

Definition 2.4 (Observation Set)**.** Given a diagnostic model $\mathcal{M}$ together with the assignment of the system behavioral mode to b, the observation set $\mathcal{O}_b$ is the set of all possible values of the observations vector $\mathbf{z}$ such that the model $\mathcal{M}$ is consistent.

Then we can define isolability and detectability as follows.

Definition 2.5 (Isolability)**.** A system behavioral mode b_1 is isolable from the system behavioral mode b_2, in model $\mathcal{M}$, if $\mathcal{O}_{b_1} \not\subseteq \mathcal{O}_{b_2}$.

Definition 2.6 (Detectability)**.** A system behavioral mode b is detectable in model $\mathcal{M}$ if b is isolable from the system behavioral mode **NF** in the model $\mathcal{M}$.

As an example, consider again the gas tank example (2.6). If all sizes of leakage and all sizes of sensor faults (also size 0) are considered, it is likely that we have the following relations.

$$\begin{array}{llll}
\mathcal{O}_{\mathbf{NF}} \subseteq \mathcal{O}_{\mathbf{NF}}, & \mathcal{O}_{\mathbf{NF}} \subseteq \mathcal{O}_{\mathbf{PSF}}, & \mathcal{O}_{\mathbf{NF}} \subseteq \mathcal{O}_{\mathbf{L}}, & \mathcal{O}_{\mathbf{NF}} \subseteq \mathcal{O}_{\mathbf{PSF\&L}}, \\
\mathcal{O}_{\mathbf{PSF}} \not\subseteq \mathcal{O}_{\mathbf{NF}}, & \mathcal{O}_{\mathbf{PSF}} \subseteq \mathcal{O}_{\mathbf{PSF}}, & \mathcal{O}_{\mathbf{PSF}} \not\subseteq \mathcal{O}_{\mathbf{L}}, & \mathcal{O}_{\mathbf{PSF}} \subseteq \mathcal{O}_{\mathbf{PSF\&L}}, \\
\mathcal{O}_{\mathbf{L}} \not\subseteq \mathcal{O}_{\mathbf{NF}}, & \mathcal{O}_{\mathbf{L}} \subseteq \mathcal{O}_{\mathbf{PSF}}, & \mathcal{O}_{\mathbf{L}} \subseteq \mathcal{O}_{\mathbf{L}}, & \mathcal{O}_{\mathbf{L}} \subseteq \mathcal{O}_{\mathbf{PSF\&L}}, \\
\mathcal{O}_{\mathbf{PSF\&L}} \not\subseteq \mathcal{O}_{\mathbf{NF}}, & \mathcal{O}_{\mathbf{PSF\&L}} \subseteq \mathcal{O}_{\mathbf{PSF}}, & \mathcal{O}_{\mathbf{PSF\&L}} \not\subseteq \mathcal{O}_{\mathbf{L}}, & \mathcal{O}_{\mathbf{PSF\&L}} \subseteq \mathcal{O}_{\mathbf{PSF\&L}}
\end{array} \tag{2.13}$$

For a compact representation of which behavioral modes that are isolable from other we can set up an *isolability matrix* where X represents the fact that the mode of the row is *not* isolable from the mode of the column. For the example we have

	NF	**PSF**	**L**	**PSF**&**L**
NF	X	X	X	X
PSF	0	X	0	X
L	0	X	X	X
PSF&**L**	0	X	0	X

(2.14)

In an ideal case, where all modes corresponding to faults in the system are isolable from all other modes, the part of the isolability corresponding to fault modes is an identity matrix.

2.7 Fault Modeling

An important concept in model based diagnosis is fault modeling. The fault model is the formal representation of the knowledge of possible faults and how they influence the process. In general, better fault models imply better diagnosis performance, e.g. smaller faults can be detected and more different types of faults can be isolated.

We will in this section give some examples of common fault modeling principles. However, when constructing a diagnosis system one should not be limited to the examples given here, but instead always choose the fault model that is best suited for the particular application, e.g. in terms of isolation requirements, accuracy requirements, time-response requirements, and computing power available.

In a formal model in form of a set of sentences we have shown in Section 2.3 how information about faults can be entered into the model. For example, when we specified the behavior of the lamp in its behavioral modes, we wrote

$$b_{\text{Lamp}} = \text{OK} \rightarrow (\text{LampVolt} \leftrightarrow \text{Light})$$
$$b_{\text{Lamp}} = \text{Broken} \rightarrow \neg\text{Light}$$

A *fault model* is a part of the model describing a component for a case when the component is in a behavioral mode classified as a fault mode.

Looking into the model, the fault model is what is written to the right of the implication arrow for the fault modes. In the example, there is only one fault model and that is ¬Light. We can consider a fault model for a system behavioral mode or for a component behavioral mode. In both cases the fault model can consist of one or more sentences (equations). In this section we will discuss only fault models and therefore not write out the implication arrow or any behavioral mode condition to the left of it.

2.7.1 Fault Signals

Commonly faults are modeled as unrestricted unknown input signals. When fault signals are used, a specific fault is usually modeled as a *scalar* signal. Fault modeling by signals is very general and can describe a wide variety of faults. However, to use fault models that are *too* general may imply that it becomes impossible to isolate between different faults (see Section 2.8 for a more thorough discussion on this topic).

In a general nonlinear state-space model, we can write

$$\begin{aligned} \dot{x}(t) &= g\big(x(t), u(t), f(t)\big) \\ y(t) &= h\big(x(t), u(t), f(t)\big) \end{aligned}$$

The signal $f(t)$ here represents an arbitrary fault that can for example be an actuator fault or a sensor fault.

The following example shows how, in a linear state-space description, the three different types actuator-, sensor-, and process-faults can be modeled by using fault signals.

Example 2.2 Consider a linear state-space model of a process:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned}$$

Consider also an additive actuator fault f_a, a general process fault f_p, and an additive sensor fault f_s, all modeled as fault signals. Then the influence of the faults to the process can be generally described by the state-space description

$$\begin{aligned} \dot{x} &= Ax + Bu + Bf_a + Ef_p \\ y &= Cx + Du + Df_a + Ff_p + f_s \end{aligned}$$

where the matrices E and F describe how the process fault enters the system.

The next example shows, for a linear system in the frequency domain, how a process fault can be modeled as a fault signal.

Example 2.3 Consider for example a process fault represented by a change $\Delta G_f(s)$ in the transfer function $G(s)$:

$$y = (G(s) + \Delta G_f(s))u$$

If the fault signal f is defined as $f = \Delta G_f(s)u$ then the fault can also be modeled as

$$y = G(s)u + f$$

Note here that if also a fault in the sensor is modeled as a fault signal, it becomes impossible to isolate between the sensor fault and the process fault.

It is also possible to include some restrictions on the fault signal $f(t)$. An example of a natural restriction is that the value of a fault signal $f(t)$ is limited in range. Another example is that the bandwidth of $f(t)$ is limited to some value. In general it is advantageous to include restrictions into the fault models. The reason is that the isolation task becomes easier the more restrictive fault models we have.

2.7.2 Deviations in Constant Parameters

Another common fault model is to model faults as deviations of constant parameters from their nominal value, e.g. (Isermann, 1993). Sensor faults that are typically modeled in this way are "gain-errors" and "off-sets" ("biases"). Another typical example is a signal whose variance is constant and low in the fault-free case, and when a fault is present the variance increases. Further on, some typical process faults consist of a deviation of a physical parameter, such as a frictional coefficient, a mass, or a leakage area (a leakage area is in the fault-free case 0).

Fault modeling by constant plant parameters is exemplified in the following example:

Example 2.4 Consider a model of an amplifier:

$$y(t) = gu(t) + v(t) \qquad v(t) \sim N(0, \sigma)$$

where $u(t)$ is the input, $y(t)$ the output, g the amplifying gain, and $v(t)$ is a noise signal with variance σ^2.

Lets say that the parameter g, i.e. the gain of the amplifier, is 10 in the nominal (fault-free) case and a gain fault is represented as a deviation from this nominal value. Further, the parameter σ^2 is assumed to be 0.01 in the nominal case and any increase of this value is considered to be a fault.

A parameter modeling a fault does not have to be scalar. A model consisting of a two-dimensional parameter is illustrated in the following example.

Example 2.5 Consider again the engine in Example 1.2 on page 17. One component that is often affected by a fault is the air-mass flow sensor. The reason for such a fault is that dirt and salt enters with the air into the engine. A faulty air-mass flow sensor makes the control system inject the wrong amount of fuel, which in turn results in increased emissions. In Figure 2.1, the *characteristic curve* of such a faulty air-mass flow sensor is shown. This sensor was found by the

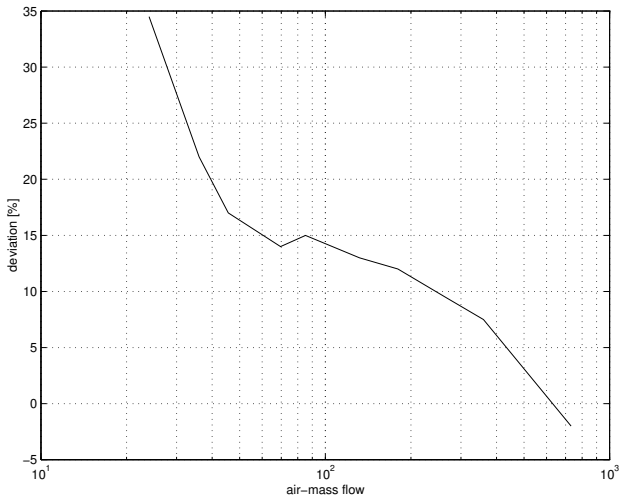


Figure 2.1: The characteristic curve of a faulty air-mass flow sensor.

service technicians after that the driver had complained about heavy smoke from the exhaust pipe.

On the vertical axis we read the deviation from the correct value for different air-mass flow. Thus a no-faulty sensor would have a

characteristic curve equal to 0. It can be noted that for large air-mass flows, the deviation is neglectable but for small, the deviation is higher than 30%.

Assume that the characteristic curve in Figure 2.1 can always be approximated by a straight line, described by a slope k and an offset m, i.e. $y = kx + m$. Then a vector-valued parameter $\theta = [k,\ m]$, can be used to model a deviation in the characteristic curve.

Also when modeling faults as deviations in constant parameters, it is possible, and often advantageous, to include some restrictions on the fault. For example the size of a bias or a gain-error is usually limited by the system.

2.7.3 Abrupt Changes

A quite common fault model is to consider abrupt changes of variables, e.g. see (Basseville and Nikiforov, 1993), representing for example a component that suddenly breaks. This is illustrated in Figure 1.5 as the solid line. It is assumed that a variable or signal has a constant value θ_0 before an unknown change-time t_{ch} and then jumps to a new constant value θ_1. The parameters θ_0 and θ_1 can be unknown or known. The main difference between this type of fault model compared to the constant parameter changes above is that the change time and the transient behavior due to the change is considered in the model.

Example 2.6 Consider an electrical connector. One possible fault is a sudden "connection cut-off" at time t_{ch}. A model for this fault is

$$y_s(t) = (1 - c(t))x(t)$$

where

$$c(t) = \begin{cases} \theta_0 = 0 & t < t_{ch} \\ \theta_1 = 1 & t \geq t_{ch} \end{cases}$$

That is, the fault model is based on an abrupt change in the signal $c(t)$. Since the levels θ_0 and θ_1 are known at beforehand, this fault can be described by the single parameter t_{ch}.

Note that the abrupt change model can also be used to model any abrupt change, and not only changes of the level of a signal. For example, we can assume that the derivative or the variance of a signal changes abruptly.

2.7.4 Incipient Faults

Incipient faults are faults that gradually develop from no fault to a larger and larger fault. This is illustrated in Figure 1.5 as the dash-dotted line. An incipient fault could for example be a slow degradation of a component or developing calibration errors of a sensor. Modeling of incipient faults are exemplified in the following example:

Example 2.7 Let $c(t)$ represent the "size" of the fault. If the fault is incipient, then $c(t)$ becomes

$$c(t) = \begin{cases} 0 & t < t_{ch} \\ \tau(t - t_{ch}) & t \geq t_{ch} \end{cases}$$

The fault can therefore be modeled as the parameter $\theta = [t_{ch}\ \tau]$. This fault model can in fact be seen as special case of the abrupt change model.

2.7.5 Intermittent Fault

An *intermittent* fault is a fault that occurs and disappears repeatedly. This was shown in Figure 1.5 as the dashed line. A typical example of an intermittent fault is a loose connector.

Example 2.8 Consider a sensor measuring a state x. The model of this (sub-) system can be written

$$y_s(t) = c_1(t)x(t)$$

where y_s is the sensor output and x is the state. The function $c_1(t)$ is our model of the loose contact. For some t, there is no contact and therefore $c_1(t) = 0$. For other t, the contact is perfect and $c_1(t) = 1$. That is, $c_1(t)$ is a function that switches between 0 and 1 at unknown time instances.

2.8 General or Restrictive Fault Models?

We have now presented a number of different principles for fault modeling. Some of them say that the fault behaves in a restrictive way, e.g. a constant bias, while others are general, e.g. the fault signal. When modeling and designing a diagnosis system, there is usually a

choice between using restrictive or general fault models. One of the considerations is that a restrictive fault model makes it easier to isolate between different faults. This is illustrated by the following example.

Example 2.9 Consider a system modeled as

$$y(t) = u_1(t) + u_2(t)$$

where $y(t)$, $u_1(t)$ and $u_2(t)$ are measured in a time window. Suppose that the signals $u_i(t)$ are actuator signals and that we want to model faults in the actuators. One possible fault model is then to assume that a fault causes a constant deviation in the gain. That is, we have the following fault model:

$$u_i(t) = \theta_i \bar{u}_i(t) \tag{2.15}$$

where $\bar{u}$ is the controlled and desired actuator value, and each θ_i is $\theta_i = 1$ in the fault free case and $\theta_i \neq 1$ if a fault is present.

If now $u_1(t)$ and $u_2(t)$ are assumed to be varying signals, then we can uniquely isolate between the faults in actuator 1 and 2. The reason is that when a fault occurs in actuator 1, the model of the system will be

$$y(t) = \theta_1 \bar{u}_1(t) + \bar{u}_2(t), \qquad \theta_1 \neq 1 \tag{2.16}$$

and the output signals generated by this system can in general *not* be explained by

$$y(t) = \bar{u}_1(t) + \theta_2 \bar{u}_2(t), \qquad \theta_2 \neq 1 \tag{2.17}$$

which is the model valid when actuator 2 is faulty. Another way of seeing this is to set the signal $y(t)$ in (2.16) and (2.17), equal to each other and then solve for the constant θ_2. That is, we start with

$$\theta_1 \bar{u}_1(t) + \bar{u}_2(t) = \bar{u}_1(t) + \theta_2 \bar{u}_2(t) \tag{2.18}$$

and solve for the constant θ_2, i.e.

$$\theta_2 = \frac{\theta_1 \bar{u}_1(t) + \bar{u}_2(t) - \bar{u}_1(t)}{\bar{u}_2(t)} = (\theta_1 - 1)\frac{\bar{u}_1(t)}{\bar{u}_2(t)} + 1, \qquad \theta_1 \neq 1 \tag{2.19}$$

Now we see that in general, the expression on the right is not constant so the equality in (2.19) can not be fulfilled for any constant θ_1. That

is, when assuming $\bar{u}_1(t)/\bar{u}_2(t)$ not constant, it is possible to isolate between actuator fault 1 and 2.

Next assume that instead of the fault model (2.15), we have a much more general fault model:

$$u_i(t) = \theta_i(t)\bar{u}_i(t)$$

That is, the constraint that θ_i is constant has been relaxed. Instead $\theta_1(t)$ and $\theta_2(t)$ are two arbitrary signals. On one hand, we have now a fault model that is able to model a much larger class of faults. On the other hand, we get problems with the isolation between actuator fault 1 and 2. In fact, as long as $u_2(t) \neq 0$, all faults in actuator 1 produce outputs $y(t)$ that can equally well be explained by a fault in actuator 2. This can be seen by again setting up equation (2.18) and solving for $\theta_2(t)$:

$$\theta_2(t) = \frac{\theta_1(t)\bar{u}_1(t) + \bar{u}_2(t) - \bar{u}_1(t)}{\bar{u}_2(t)} = (\theta_1(t) - 1)\frac{\bar{u}_1(t)}{\bar{u}_2(t)} + 1 \qquad (2.20)$$

For each possible fault modeled by $\theta_1(t) \not\equiv 1$, we can always find $\theta_2(t)$ so that the equality in (2.20) is fulfilled.

The advantage of using a general fault model is clearly that no or not much knowledge about the fault behavior is needed. As was shown in the previous example, the disadvantage is that in a system with several of the faults modeled by general fault models, it can be difficult or even impossible to isolate between the different faults. Restrictive fault models make it easier to isolate between faults but requires more information about the faults and may result in a more complex diagnosis system. Thus, fault modeling is always a trade off between, on one hand good isolability and on the other hand, low requirements on fault knowledge and low complexity of the diagnosis system. It is worth stressing that obtaining models on how different faults occur and influence a process may be extremely difficult to come by. It is often a difficult task to model a fault free process, obtaining faulty data for model building is often rare, especially early in a development process.

Chapter 3
Diagnosis Systems for Fault Isolation

In the previous chapter we formalized the meaning of models, especially with the perspective of diagnosis. We also defined the term diagnosis. However, we only stated *what* diagnoses are but not *how* to compute them. How to construct a diagnosis system that computes diagnoses, given a model and observations, is the topic of the present chapter. With the term fault isolation we mean the process of computing the diagnoses given some observations or some processed version of the observations.

We spend the major part of the chapter aiming at computing all diagnoses. It might seem strange that the task of a diagnosis system is not to pick out one system behavioral mode, for example *the* most probable one, but instead to give all system behavioral modes that can explain the observations. Further, the set of all diagnoses may not be useful in a practical application, especially if multiple faults are considered, since the number of diagnoses then may become very large. However it is important to note that sometimes this corresponds to a desired functionality since in cases where it is difficult or even impossible to decide which system behavioral mode that is present, it is very useful for e.g. a service technician to get to know that there are more than one system behavioral mode that can explain the behavior of the process. If the diagnosis system was forced to pick out one

system behavioral mode in cases like this, the probability of making an erroneous guess may become too high. However, in most cases there are good reasons to not consider the set of all diagnoses. Therefore we will later in Section 3.7 also discuss how to compute not all but a limited and prioritized set of diagnoses.

3.1 Diagnosis Systems

As said in the Chapter 1, the output from a diagnosis system is the *diagnosis statement* S. If we represent S as a set, the task of a pure consistency based diagnosis system can be described as follows.

> Given a set of observations, the task of a consistency based diagnosis system is to generate a set S, the *diagnosis statement*, which contains the system behavioral modes that can explain the observations; the diagnoses.

Note that we assume that the diagnosis system is *passive*, which means that it by no means can affect the process. We also assume that the diagnosis system is deterministic and has no memory, i.e. the same observations will always give the same diagnosis statement.

In terms of *decision theory* (e.g. see (Berger, 1985)), the diagnosis system is a *decision rule* $\delta(z)$, i.e. a function from the observations z to the diagnosis statement:

$$\delta : \mathcal{Z} \longrightarrow \mathcal{P}(\Omega)$$

where $\mathcal{Z}$ is the set of all possible observations and $\mathcal{P}(\Omega)$ is the *power set* of Ω, i.e. the set of all subsets of Ω. Since we mainly consider systems controlled by computerized control systems, the observations usually consist of inputs $u(t)$ and outputs $y(t)$ from the system. Thus, we can write $S = \delta(z) = \delta([u\ y])$. Here, z is used to denote the whole observed data-set, which usually consists of all known and measured variables of the system up to present time or a subset of this data. One choice is to use a fixed size time window.

3.1.1 The Architecture of a Diagnosis System

Model based diagnosis is a complex task and it is therefore advantageous to divide the task into smaller subtasks. In this text, we therefore

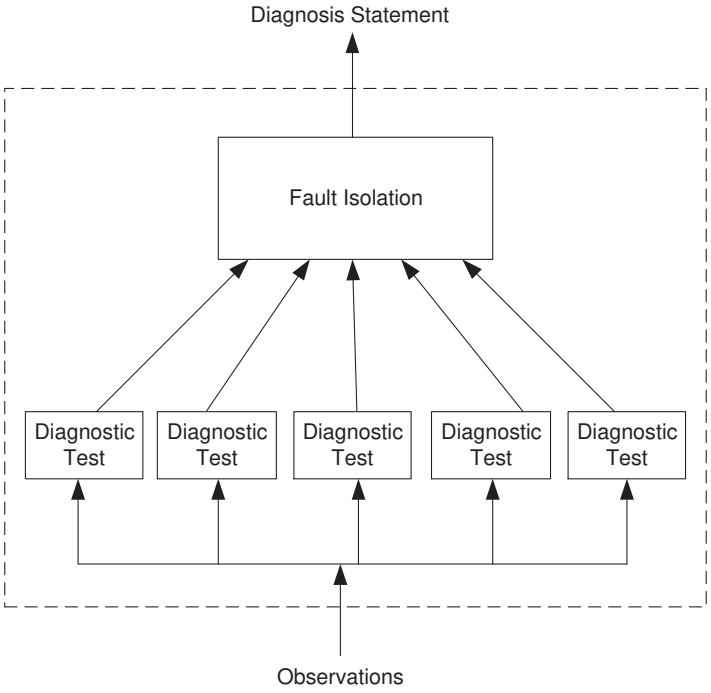


Figure 3.1: The basic architecture of a diagnosis system.

assume that a diagnosis system $\delta(z)$ consists of a number of *diagnostic tests* $\delta_k(z)$ and an *isolation logic*, see Figure 3.1.

The purpose of each diagnostic test is to investigate if some specific system behavioral modes are present or not. We assume that each diagnostic test $\delta_k(z)$ is binary in the sense that there are two possible outcomes. The construction of diagnostic tests is a central topic in fault diagnosis and will therefore be discussed in all of the remaining chapters.

The purpose of the *isolation logic* is to combine the information from the diagnostic tests to form the diagnosis statement S. The basic idea of fault isolation is that different diagnostic tests respond differently to different system behavioral modes, and by utilizing this fact, fault isolation can be performed. The exact procedure for fault isolation will be described later in this chapter.

3.1.2 Sound and Complete Diagnosis Statements

A diagnosis statement S is said to be *complete* if all diagnoses are included in S. Further, the diagnosis statement S is said to be *sound* if all system behavioral modes included in S are diagnoses. A goal when designing a diagnosis system can be that the diagnosis statement should be always sound and complete. Note that even though the diagnosis statement S is both complete and sound, it may in a practical application be very large.

3.2 Diagnostic Tests

The general architecture of a diagnosis system was shown in Figure 3.1 and this section will be devoted to further discussions about the diagnostic tests. Using the general view of hypothesis testing, we first describe how the diagnostic tests can be seen as binary *hypothesis tests*.

The classical, statistical or decision theoretic, definition of *hypothesis test* is adopted, e.g. see (Berger, 1985; Lehmann, 1986; Casella and Berger, 1990), which means that we consider hypothesis tests that are binary in the sense that the outcome of a hypothesis test is one out of two possible decisions. This should be distinguished from "multiple hypothesis testing" that can also be found in literature, e.g. (Basseville and Nikiforov, 1993).

Each diagnostic test $\delta_k(z)$ generates a sub-diagnosis statement S_k, i.e. $S_k = \delta_k(z)$. Since the test is binary there are two possible statements: S_k^0 and S_k^1. For each hypothesis test δ_k, we need to find a *rejection region* $\mathcal{R}_k$, i.e. a subset of the set of observations $\mathcal{Z}$ where the so called null-hypothesis is rejected. The function of a hypothesis test can be written as

$$S_k = \delta_k(z) = \begin{cases} S_k^1 & \text{if } z \in \mathcal{R}_k \\ S_k^0 & \text{if } z \notin \mathcal{R}_k \end{cases} \tag{3.1}$$

Usually the rejection region is not explicitly utilized in the decision between S_k^1 and S_k^0. Instead the decision is done via a *test quantity* (often also called test statistic). The test quantity is a function $T_k(\mathbf{z})$ from the observations $\mathbf{z}$ to a scalar value which is to be compared with a threshold J_k. Typically if $T_k(\mathbf{z}) \geq J_k$, then H_k^0 is rejected and otherwise not rejected. The rejection region of each test is thereby

implicitly defined. Formally the hypothesis test δ_k can now be written as

$$S_k = \delta_k(z) = \begin{cases} S_k^1 & \text{if } T_k(z) \geq J_k \\ S_k^0 & \text{if } T_k(z) < J_k \end{cases} \tag{3.2}$$

When we divide the observations into inputs $u(t)$ and outputs $y(t)$, the sample data $\mathbf{z}$ for each hypothesis is

$$\mathbf{z}(t) = \begin{bmatrix} u(t-N) & u(t-N+1) & \dots & u(t) \\ y(t-N) & y(t-N+1) & \dots & y(t) \end{bmatrix} \tag{3.3}$$

To simplify the notation, we have assumed that unit sample-time is used. The sample data (3.3) corresponds to the use of a finite time window. This time window can be a sliding window, which means that the data $\mathbf{z}$ becomes a function of time t and that consecutive data sets are overlapping. Another choice is to let consecutive data sets be non-overlapping. The time window can also be infinite. This corresponds to that $N = \infty$ in (3.3). An example of when an infinite time window is desirable is when recursive techniques are used to calculate the test quantities. This is also the case when using test quantities based on residuals that are realized as IIR-filters, e.g. see Chapter 6.

The test quantity is a function $T_k(z)$ from the sample data $\mathbf{z}$, to a scalar value which is to be thresholded by a threshold J_k. Thus δ_k will have a structure according to Figure 3.2. The test quantity $T_k(z)$ is

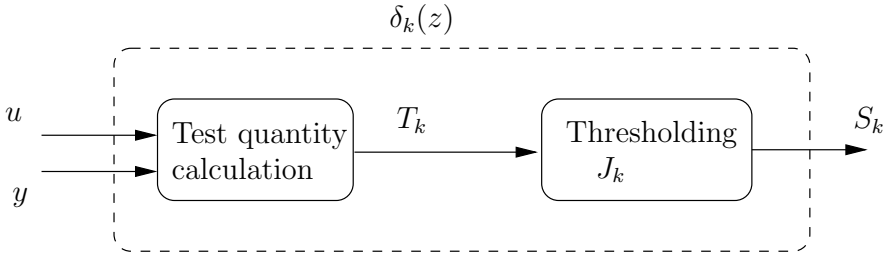


Figure 3.2: Hypothesis test $\delta_k(z)$.

in many statistical texts instead called a *test statistic*. However, the name *test statistic* indicates that $T_k(z)$ is a random variable which in general may not always be a desired view. Thus, the more neutral

term test quantity is used which may or may not be considered to be a stochastic variable, all depending on circumstances and tools used when designing the test. The test quantity $T_k(z)$ may for example be based on a *residual generator* or a sum of squared prediction errors of a parameter estimator. Sometimes, a purely deterministic view is taken and $T_k(z)$ is seen just as a function of the data and not as a random variable.

3.3 Column Matching Approach

In this section we introduce the *column matching approach* to fault isolation. Given a set of diagnostic tests, we can, as a part of the diagnosis system design, store a table of how the different tests respond when the system is in different system behavioral modes. An example is the following table.

	NF	F_1	F_2	F_3
$T_1(z)$	0	0	1	0
$T_2(z)$	0	0	1	1
$T_3(z)$	0	1	0	1

(3.4)

Then, when the diagnosis system is up and running, the actual response from the diagnostic tests is compared with the stored responses. For each match between a column and the actual response, the system behavioral mode of the column is considered to be a diagnosis.

Digging deeper into this approach we realize that the stored table comes in two versions that we call the *influence structure* and the *decision structure*[1]. The influence structure describes how the faults ideally affect the diagnostic tests, and the decision structure, describes how the diagnosis statement depends on the diagnostic tests.

3.3.1 Influence Structure

To get an overview of how faults in different system behavioral modes *ideally* affect the test quantities of the diagnostic tests, it is useful to set up an *influence structure*. With *ideally*, we mean that the system behaves exactly in accordance with the model, e.g. no unmodeled

[1]Similar structures in the fault diagnosis literature are also called for example *residual structure* or *incidence matrix*.

disturbances exist and there is no measurement noise. An influence structure is a table or matrix containing 0:s, 1:s, and X:s. An example of an influence structure is

	NF	F_1	F_2	F_3
$T_1(z)$	0	0	1	0
$T_2(z)$	0	0	1	1
$T_3(z)$	0	X	0	1

(3.5)

A 0 in the k:th row and the j:th column means that if the system behavioral mode present in the system, is equal to the system behavioral mode of the j:th column, then the test quantity $T_k(z)$ will not be affected, i.e. it will be exactly zero. A 1 in the k:th row and the j:th column means that for *all* faults belonging to the system behavioral mode of the j:th column, $T_k(z)$ will always be affected, i.e. it will be non-zero. An X in the k:th row and the j:th column means that for *some* faults belonging to the system behavioral mode of the j:th column, $T_k(z)$ will under some operating conditions be affected, i.e. it will be non-zero. The influence structure (3.5) includes only single faults, but it is also possible to include multiple faults as will be described later in this section.

The influence structure is derived by studying the equations describing the process model and how the test quantities $T_k(z)$ are calculated. This is illustrated in the following example:

Example 3.1 Consider again Example 1.2 on page 17. The airflow W past the throttle can be modeled as a non-linear function of the throttle angle α and the manifold pressure p:

$$W = (1 - \cos\alpha)\Phi(p) \tag{3.6}$$

where the $d\Phi(p)/dp = 0$ for supersonic air-speeds which occurs for all $p < 53$ kPa (Heywood, 1992). The throttle angle α is always between 0 and $\pi/2$.

Three system behavioral modes are considered: no fault NF, air mass-flow sensor fault M, and manifold pressure sensor fault P. For both M and P, the faults are modeled as an arbitrary signal added to the sensor signals:

$$W_s = W + f_M \tag{3.7a}$$

$$p_s = p + f_P \tag{3.7b}$$

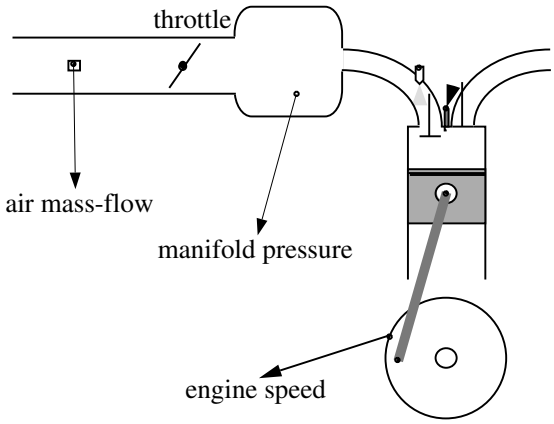


Figure 3.3: A principle illustration of an SI-engine.

where the index s indicates sensor signals. As test quantity, we can use

$$T(z) = T([W_s, \alpha_s, p_s]) = W_s - (1 - \cos\alpha_s)\Phi(p_s) \tag{3.8}$$

To see how the faults affects the test quantity, we can substitute (3.6) and (3.7) into (3.8):

$$\begin{aligned} T(z) &= W + f_M - (1 - \cos\alpha)\Phi(p + f_P) = \\ &= f_M + (1 - \cos\alpha)\Phi(p) - (1 - \cos\alpha)\Phi(p + f_P) \end{aligned}$$

We see that a fault belonging to M will always affect $T(z)$. Also, a fault belonging to P will surely affect $T(z)$ if $p > 53$kPa, but not always when $p < 53$kPa.

This means that the influence structure for the test quantity $T(z)$ becomes

	NF	M	P
$T(z)$	0	1	X

(3.9)

Let s_{kj} denote the entry in the k:th row and the j:th column of an influence structure. Then the interpretation, or semantics, of 0:s, 1:s, and X:s can be formalized as

$$F_p = F_j \;\rightarrow\; T_k(z) = 0 \qquad \text{if } s_{kj} = 0 \tag{3.10a}$$

$$F_p = F_j \;\rightarrow\; T_k(z) \neq 0 \qquad \text{if } s_{kj} = 1 \tag{3.10b}$$

where F_p denotes the present system behavioral mode.

Note that the implication, denoted by the arrow, is not symmetric. Note also that the interpretation of X is implicitly contained in these two formulas.

By using the formulas (3.10), it is possible to formally describe the interpretation of a whole influence structure. We will exemplify this below, by giving the interpretation of the influence structure (3.5), but note first that $F_p \notin \{F_2\}$ is equivalent to that $F_p \in \Omega - \{F_2\}$. The symbol $\Longleftrightarrow$ will be used to denote equivalence. Now, the interpretation of the influence structure (3.5) becomes

$$
\begin{array}{lllll}
T_1 = 0 & \leftarrow & F_p \in \{NF, F_1, F_3\} & \Longleftrightarrow & T_1 \neq 0 \rightarrow F_p = F2 \\
T_1 \neq 0 & \leftarrow & F_p = F2 & \Longleftrightarrow & T_1 = 0 \rightarrow F_p \in \{NF, F_1, F_3\} \\
T_2 = 0 & \leftarrow & F_p \in \{NF, F_1\} & \Longleftrightarrow & T_2 \neq 0 \rightarrow F_p \in \{F_2, F_3\} \\
T_2 \neq 0 & \leftarrow & F_p \in \{F_2, F_3\} & \Longleftrightarrow & T_2 = 0 \rightarrow F_p \in \{NF, F_1\} \\
T_3 = 0 & \leftarrow & F_p \in \{NF, F_2\} & \Longleftrightarrow & T_3 \neq 0 \rightarrow F_p \in \{F_1, F_3\} \\
T_3 \neq 0 & \leftarrow & F_p = F_3 & \Longleftrightarrow & T_3 = 0 \rightarrow F_p \in \{NF, F_1, F_2\}
\end{array}
$$

By using *if-and-only-if* relations, these formulas can be written on a slightly shorter form:

$$
\begin{array}{lllll}
T_1 = 0 & \leftrightarrow & F_p \in \{NF, F_1, F_3\} & \Longleftrightarrow & T_1 \neq 0 \leftrightarrow F_p = F2 \\
T_2 = 0 & \leftrightarrow & F_p \in \{NF, F_1\} & \Longleftrightarrow & T_2 \neq 0 \leftrightarrow F_p \in \{F_2, F_3\} \\
T_3 = 0 & \leftarrow & F_p \in \{NF, F_2\} & \Longleftrightarrow & T_3 \neq 0 \rightarrow F_p \in \{F_1, F_3\} \\
T_3 \neq 0 & \leftarrow & F_p = F_3 & \Longleftrightarrow & T_3 = 0 \rightarrow F_p \in \{NF, F_1, F_2\}
\end{array}
$$

As seen, the *if-and-only-if* relation can only be used with rows, in the influence structure, which have no X:s.

The influence structure corresponds to the case where ideal conditions holds. If this was the case, we could derive the diagnosis statement S by using the influence structure, the formulas (3.10), and the values of the test quantities $T_k(\mathbf{z})$. In practice, the model is not perfect; unmodeled disturbances affect the process, and there is measurement noise. All this means that the formulas (3.10) are not valid and can therefore not be used in practice to form the diagnosis statement S. Instead we will use the decision structure as will be described in Section 3.3.2.

Multiple Faults

Also multiple faults can be included into the influence structure (and also in the decision structure described in the next section). For example, if the influence structure (3.5) is expanded we obtain the following influence structure.

	NF	F_1	F_2	F_3	$F_1\&F_2$	$F_2\&F_3$	$F_1\&F_3$	$F_1\&F_2\&F_3$
$T_1(z)$	0	0	1	0	1	1	0	1
$T_2(z)$	0	0	1	1	1	1	1	1
$T_3(z)$	0	X	0	1	X	1	1	1

(3.11)

The part of the table corresponding to multiple faults, is constructed using the assumption that two faults cannot compensate out each other. If this is not true, i.e. some faults can compensate out each other, we can not use so many 1:s in the columns for the multiple faults. To be on the safe side, one can use the convention to only use X:s for multiple faults. Another possibility is to study each test $T_k(\mathbf{z})$ separately and conclude if the entry should be 1 or X, or even 0. Note that if the multiple fault part of the table can be derived from the single fault part, e.g. as in (3.11), there is actually no need to explicitly store the multiple fault part. Instead, they can be retrieved on the fly in the process of matching columns.

3.3.2 Decision Structure

In practice, we have to relax the assumptions of ideal conditions and the formulas (3.10) can be replaced by a formulation based on the use of thresholded test quantities. Doing this, we obtain a *decision structure*. Still letting s_{kj} denote the entry in the k:th row and the j:th column, the new interpretation of 0:s, 1:s, and X:s becomes

$$\mathbf{F_p} = \mathbf{F_j} \;\rightarrow\; T_k(\mathbf{z}) < J_k \qquad \text{if } s_{kj} = 0 \tag{3.12a}$$

$$\mathbf{F_p} = \mathbf{F_j} \;\rightarrow\; T_k(\mathbf{z}) \geq J_k \qquad \text{if } s_{kj} = 1 \tag{3.12b}$$

The implications are usually not completely true, but we assume that they hold up to a probability sufficiently close to 1. This corresponds to the basic assumptions, discussed in Section 3.4.1, that when H_k^0 is rejected, we assume that H_k^1 holds. However, there is a conflict between the two rules (3.12a) and (3.12b). To make the assumption

that (3.12a) holds reasonable in the case of noise, thresholds must be chosen relatively high[2]. This means that the thresholds must be chosen relatively high. Further, this violates the assumption that (3.12b) holds. To achieve reasonable assumptions, some or probably most 1:s from the influence structure must therefore be replaced by X:s. It might seem that another choice is to replace 0:s by X:s, but the problem with this is that for all small faults, the assumption of (3.12b) still not becomes reasonable.

An example of a decision structure is obtained by considering the influence structure (3.5) which can be transformed to (by replacing 1:s with X:s), for instance the following decision structure:

	NF	$\mathbf{F_1}$	$\mathbf{F_2}$	$\mathbf{F_3}$
$\delta_1(\mathbf{z})$	0	0	X	0
$\delta_2(\mathbf{z})$	0	0	X	1
$\delta_3(\mathbf{z})$	0	X	0	X

(3.13)

Because the decision structure is related to the whole hypothesis tests and not only the test quantities, we use δ_k to label the rows instead of T_k. In cases when the NF-column only contains 0:s, and this is trivially understood, the NF-column will sometimes not be included.

The process of replacing 1:s with X:s is further illustrated by the following example:

Example 3.2 Consider again Example 3.1. When the system behavioral mode M is present, we have that

$$T(z) = f_M + v$$

where v is a signal that represents model errors, disturbances, and measurement noise. Even for the system behavioral mode NF, which implies $f_M = 0$, the test quantity $T(z)$ will *not* be zero. This means that the threshold J must be raised above zero. Then for small f_M, $T(z)$ will not reach the threshold.

If the influence structure (3.9) would be used as decision structure, we would have the rule

$$F_p = M \rightarrow T(z) \geq J$$

[2]This corresponds to that the significance level α_k must be low, see Section 3.4.2.

However, according to what was said above, the implication will not hold for a small f_M. This means that to obtain the decision structure, the 1 in (3.9) must be replaced by an X, i.e.

	NF	M	P
δ	0	X	X

A decision structure can be used to derive the diagnosis statement by a simple column matching as described in the beginning of Section 3.3. Another way is to use the table together with the formulas (3.12). Consider for example the decision structure (3.13), which has the interpretation

$$
\begin{array}{lllll}
T_1 < J_1 & \leftarrow & F_p \in \{\, NF, F_1, F_3\} & \Longleftrightarrow & T_1 \geq J_1 \rightarrow F_p = F2 \\
T_2 < J_2 & \leftarrow & F_p \in \{\, NF, F_1\} & \Longleftrightarrow & T_2 \geq J_2 \;\rightarrow\; F_p \in \{\, F_2, F_3\} \\
T_2 \geq J_2 & \leftarrow & F_p = F_3 & \Longleftrightarrow & T_2 < J_2 \;\rightarrow\; F_p \in \{NF, F_1, F_2\} \\
T_3 < J_3 & \leftarrow & F_p \in \{\, NF, F_2\} & \Longleftrightarrow & T_3 \geq J_3 \;\rightarrow\; F_p \in \{F_1\ ,\ F_3\}
\end{array}
$$

Now if $T_1 < J_1$, $T_2 \geq J_2$, and $T_3 \geq J_3$, we know by using the rules, that $F_p \in \{\, F_2, F_3\}$ and $F_p \in \{F_1\ ,\ F_3\}$. This means that F_3 must be the present system behavioral mode.

3.3.3 The Importance of Using X:s in the Decision Structure

When 1:s are replaced with X:s and vice verse, the produced diagnosis statement will in general change. If X:s are not inserted in the right places, the diagnosis statement may become incorrect. The next example will demonstrate the isolation procedure and also highlight this difference between X:s and 1:s.

Example 3.3 Consider the following two decision structures

	NF	F_1	F_2	F_3
$\delta_1(z)$	0	0	1	0
$\delta_2(z)$	0	0	1	1
$\delta_3(z)$	0	1	0	1

	NF	F_1	F_2	F_3
$\delta_1(z)$	0	0	X	0
$\delta_2(z)$	0	0	X	1
$\delta_3(z)$	0	X	0	X

(3.14)

Assume that these decision structures represent two different diagnosis systems but with the same set of test quantities and thresholds. Then Table 3.1 contains a comparison between the diagnosis statement generated from the left structure and the diagnosis statement generated

from the right structure. The leftmost column lists all possible results of thresholding the test quantities and, for example the second row in Table 3.1, $(0, 0, 1)$, corresponds to $T_1 < J_1$, $T_2 < J_2$, and $T_3 > J_3$.

			Left Struct.	Right Struct.
T_1	T_2	T_3	S	S
0	0	0	$\{NF\}$	$\{NF, F_1, F_2\}$
0	0	1	$\{F_1\}$	$\{F_1\}$
0	1	0	$\{\}$	$\{F_2, F_3\}$
0	1	1	$\{F_3\}$	$\{F_3\}$
1	0	0	$\{\}$	$\{F_2\}$
1	0	1	$\{\}$	$\{\}$
1	1	0	$\{F_2\}$	$\{F_2\}$
1	1	1	$\{\}$	$\{\}$

Table 3.1: The diagnosis statements using the left and the right decision structure in (3.14).

Remember from Section 3.3.1 that a 1 in the column for F_j means that we assume that all faults, belonging F_j, affects the test quantity. This can in most cases not be assumed and therefore, the right decision structure in (3.14) is normally the formally correct one to use.

3.4 Structured Hypothesis Tests Approach

Structured hypothesis tests represent another view on the fault isolation problem. The basic principle is that during the design of the diagnosis system, we carefully formulate the null and alternative hypotheses of each hypothesis test in the diagnosis system. Typically we represent these hypotheses using sets of system behavioral modes. Then the diagnosis statement can be formed by a simple intersection of alternative hypotheses from the tests that have responded, i.e. the tests that have rejected their null hypothesis.

3.4.1 Hypothesis Tests

As was stated in Section 3.2, each diagnostic test $\delta_k(z)$ generates a sub-diagnosis statement S_k, i.e. $S_k = \delta_k(z)$. In analogy with the

diagnosis statement S, also the sub-diagnosis statements S_k will here be represented by sets.

The null hypothesis for the k:th hypothesis test, i.e. H_k^0, is that the system behavioral mode, present in the process, belongs to a specific set M_k of system behavioral modes. The system behavioral modes in M_k correspond to the non-monitored faults. The alternative hypothesis H_k^1 is that the present system behavioral mode does not belong to M_k. This means that if hypothesis H_k^0 is rejected, and thus H_k^1 is accepted, the present system behavioral mode can not belong to M_k, i.e. it must belong to the complement set of M_k denoted M_k^C. In this way, each hypothesis test can contribute with a piece of information about which system behavioral modes that can be present.

Let F_p denote the system behavioral mode present in the system (the process). Then for the k:th hypothesis test, the null hypothesis and the alternative hypothesis can be written

$$H_k^0 : F_p \in M_k \qquad \text{some behavioral mode in } M_k \text{ can explain observations} \tag{3.15a}$$

$$H_k^1 : F_p \in M_k^C \qquad \text{no behavioral mode in } M_k \text{ can explain observations} \tag{3.15b}$$

The convention used here and also common in hypothesis testing literature, is that when H_k^0 is rejected; we *assume* that H_k^1 is true. Further, when H_k^0 is *not* rejected, we will for the present not assume anything. This latter fact will be slightly modified in Section 3.4.2, where we discuss how we can assume something also when H_k^0 is *not* rejected. However, it always holds that $M_k \subseteq S_k^0 \subseteq \Omega$ where S_k^0 is the decision taken when H^0 is not rejected. Using these principles means that

$$S_k = \begin{cases} S_k^1 = M_k^C & \text{if } H_k^0 \text{ is rejected } (H_k^1 \text{ accepted}) \\ S_k^0 \subseteq \Omega & \text{if } H_k^0 \text{ is } \textit{not} \text{ rejected} \end{cases}$$

Much of the engineering work involved in constructing a diagnosis system is to use the model of the system to construct the individual hypothesis tests, i.e. decide which hypotheses to test and device a procedure to determine if the null hypothesis should be rejected or not.

An Alternative Representation of the Hypothesis Tests

Above, we used the set M_k to formulate the null and the alternative hypotheses. An alternative, and more standard in the world of hypothesis testing, is to use a parameter θ together with parameter sets Θ_k^0 corresponding to the different null hypotheses.

To illustrate this, assume that the process to be diagnosed is modeled with a model $\mathcal{M}(\theta)$. The model $\mathcal{M}(\theta)$ includes the fault models and consists of differential equations, algebraic equations, and possibly also stochastic parts. The parameter vector θ is called the *fault state* and represents the true but unknown fault situation of the process.

That is, all information about possible faults of the process is collected in the parameter θ. At least one value of the fault state θ always correspond to the no-fault system behavioral mode. The *fault state-space*, i.e., the space over which θ can vary will be denoted Θ.

Note that we have chosen the convention that θ is not dependent on time which corresponds to the assumption that the system behavioral mode is constant over the time window considered. Even though this may seem to be a limitation, this is not the case since we will be quite liberal regarding the definition of the parameter vector θ, e.g. elements are allowed to be not only constant values but also signals.

The classification of different faults into system behavioral modes corresponds to a *partition* of the fault-state space Θ. Each system behavioral mode γ is associated with a subset Θ_γ of Θ. Thus all sets Θ_γ are pairwise disjoint and $\Theta = \cup_{\gamma\in\Omega}\Theta_\gamma$. If system behavioral mode γ is present in the system, then we know that $\theta \in \Theta_\gamma$. The fact that all sets Θ_γ are pairwise disjoint reflects the fact that only one system behavioral mode can be present at the same time. With the general model $\mathcal{M}(\theta)$, each system behavioral mode γ can now be seen as a model of the process, namely the model $\mathcal{M}(\theta)$, where $\theta \in \theta_\gamma$.

Example 3.4 Consider a system described by the following equations:

$$\begin{aligned}\dot{x}(t) =& f(x(t), u(t))\\ y_1(t) =& h_1(x(t)) + b_1\\ y_2(t) =& h_2(x(t)) + b_2\end{aligned}$$

The constants b_1 and b_2 represents sensor bias faults and it is assumed that only positive biases can occur. It is natural to let $\theta_1 = b_1$ and $\theta_2 = b_2$, and thus $\theta = [\theta_1\ \theta_2] = [b_1\ b_2]$. Four system behavioral modes

are considered: "no fault" **NF**, "bias in sensor 1" **B1**, "bias in sensor 2" **B2**, "bias in both sensor 1 and sensor 2" **B1**&**B2**. The sets Θ, $\Theta_{\mathbf{NF}}$, $\Theta_{\mathbf{B1}}$, $\Theta_{\mathbf{B2}}$, and $\Theta_{\mathbf{B1\&B2}}$ become

$$\begin{aligned}
\Theta &= \{[b_1\ b_2];\ b_1 \geq 0, b_2 \geq 0\} \\
\Theta_{\mathbf{NF}} &= \{[0\ 0]\} \\
\Theta_{\mathbf{B1}} &= \{[b_1\ 0];\ b_1 > 0\} \\
\Theta_{\mathbf{B2}} &= \{[0\ b_2];\ b_2 > 0\} \\
\Theta_{\mathbf{B1\&B2}} &= \{[b_1\ b_2];\ b_1 > 0, b_2 > 0\}
\end{aligned}$$

We can now use the definition of the sets Θ_γ to describe the hypotheses. This is done via sets Θ_k^0 which are defined as

$$\Theta_k^0 = \bigcup_{\gamma \in M_k} \Theta_\gamma$$

The hypotheses can now be expressed as

$H_k^0 : \theta \in \Theta_k^0$ some value of $\theta \in \Theta_k^0$ can explain observations
$H_k^1 : \theta \notin \Theta_k^0$ *no* value of $\theta \in \Theta_k^0$ can explain observations

Depending on the context, hypotheses will in the text be represented either by the M_k-sets or the Θ_k^0-sets. However, it should be remembered that both representations are equivalent.

3.4.2 Constructing Hypothesis Tests for Diagnosis

The definition (3.2) means that we need to design a test quantity $T_k(z)$ such that it is low if the data z match the hypothesis H_k^0, i.e. a system behavioral mode in M_k can explain the data. Also if the data come from a system behavioral mode not in M_k, $T_k(z)$ should be large. This is formalized by the *power function* $\beta_k(\theta)$:

$$\beta_k(\theta) = P(\text{reject } H_k^0 \mid \theta) = P(T_k(z) \geq J_k \mid \theta)$$

We want the power function to be low for $\theta \in \Theta_k^0$ and large for $\theta \notin \Theta_k^0$. To be able to make the assumption that H_k^1 is true when H_k^0 is rejected,

we need to design the hypothesis tests such that the *significance level* α, defined as

$$\alpha = \sup_{\theta \in \Theta_k^0} \beta_k(\theta)$$

has a small value. This implies that the threshold J_k must be set relatively high. This in turn means that the value of $\beta_k(\theta)$ does not necessarily become large for all values $\theta \notin \Theta_k^0$. For instance, if the present system behavioral mode is F_i, then for some $\theta \in \Theta_{F_i}$, the probability to reject H_k^0 may be very small. This is the reason why we up to now, have assumed that $S_k^0 = \Omega$, i.e. we can not assume anything when H_k^0 is not rejected.

Now if it actually holds that the power function is large for all $\theta \in \Theta_{F_i}$, then we do not take any large risk if we assume that F_i has not occurred when H_k^0 is not rejected. If this is the case, F_i should be excluded from S_k^0.

How the test quantities $T_k(z)$ are constructed depends on the actual case and only for some specific classes of systems and fault models, general design procedures have been proposed, e.g. linear systems with fault modeled as inputs which will be described in Chapter 6.

To develop the actual hypothesis tests, we first need to decide the set of hypotheses to test. One solution is to use one hypothesis test for each system behavioral mode. In this case, the set of hypothesis tests can be indexed by $\gamma \in \Omega$, i.e. δ_γ, and becomes

$$H_\gamma^0 : F_p \in M_\gamma \tag{3.16a}$$

$$H_\gamma^1 : F_p \in M_\gamma^C \tag{3.16b}$$

$$\gamma \in \Omega \tag{3.16c}$$

It turns out that some system behavioral modes are related to other system behavioral modes such that for some values of θ they are impossible to separate. This has implications on how the sets M_k can be chosen. For example for most system behavioral modes, the limit when the fault size goes to zero is equal to the system behavioral mode "no fault". This means that when system behavioral mode NF, i.e. no fault, is present, most null hypothesis can not be rejected. The implication is that for almost all sets M_k it is possible to include NF. For example assume that $M_1 = \{F_1\}$ but small faults belonging to F_1 are impossible to distinguish from NF. Then M_1 can be extended to

$M_1' = \{F_1, NF\}$. When the null hypothesis is rejected, we can now conclude $F_p \notin \{F_1, NF\}$ which is a stronger conclusion, and therefore more desirable, compared to $F_p \notin \{F_1\}$.

3.4.3 The Fault Isolation

In the view of hypothesis testing, fault isolation logic becomes a simple intersection operation. This procedure will be described in this section, and we will also study the relationship to the column matching approach described in Section 3.3. The principle of the fault isolation is illustrated by the following example.

Example 3.5 Assume that the set of all possible system behavioral modes is $\Omega = \{NF, F_1, F_2, F_3\}$ and that the diagnosis system contains the following set of three hypothesis tests:

$$\begin{aligned} H_1^0 &: F_p \in M_1 = \{NF, F_1\} \qquad S_1^0 = \Omega \\ H_1^1 &: F_p \in M_1^C = \{F_2, F_3\} \qquad S_1^1 = \{F_2, F_3\} \end{aligned}$$

$$\begin{aligned} H_2^0 &: F_p \in M_2 = \{NF, F_2\} \qquad S_2^0 = \Omega \\ H_2^1 &: F_p \in M_2^C = \{F_1, F_3\} \qquad S_2^1 = \{F_1, F_3\} \end{aligned}$$

$$\begin{aligned} H_3^0 &: F_p \in M_3 = \{NF, F_3\} \qquad S_3^0 = \Omega \\ H_3^1 &: F_p \in M_3^C = \{F_1, F_2\} \qquad S_3^1 = \{F_1, F_2\} \end{aligned}$$

Then if only H_1^0 is rejected, we draw the conclusions that $F_p \in S_1^1$, $F_p \in S_2^0$, $F_p \in S_3^0$. That is, $F_p \in S_1^1 \cap S_2^0 \cap S_3^0 = \{F_2, F_3\} \cap \Omega \cap \Omega = \{F_2, F_3\}$, i.e. the present system behavioral mode is either F_2 or F_3. If both H_1^0 and H_2^0 are rejected, we draw the conclusion that $F_p \in \{F_2, F_3\} \cap \{F_1, F_3\} \cap \Omega = \{F_3\}$, i.e. the present system behavioral mode is F_3.

From the example above, it is clear that the isolation logic is a simple intersection operation, i.e. the diagnosis statement S can be expressed as

$$S = \bigcap_k S_k \tag{3.17}$$

As stated in Section 3.1.2, a desirable goal when designing a diagnosis system is that the diagnosis statement should always be sound

and complete. When 1:s and X:s are used in accordance with the formulas (3.12), the diagnosis statement will in fact always be complete. If a structure with only 1:s are used, in a case where X:s should have been used, the diagnosis statement will not be complete. This can be seen in Example 3.3, in the first, third, and the fifth row in Table 3.1.

To get a sound diagnosis statement, i.e. all conclusions in the statement are really diagnoses, is more difficult to guarantee. However with ideal test quantities a sufficient condition is to have enough hypothesis tests according to the result below. With ideal test quantities we mean that $T_k = 0$ if and only if the observations can be explained by any of the system behavioral modes in M_k. This means implicitly that we assume that there is no noise in the process.

Theorem 3.1. Let a diagnosis system be constructed with one hypothesis test for each system behavioral mode F_i, i.e. $H_i^0 : F_p = F_i$. Assume that ideal test quantities are used and let the decision structure be chosen such that the formulas (3.10) are valid. Then the diagnosis statement S will always be *logically sound*.

3.4.4 Relation to Column Matching Approach

There is a strong relationship between structured hypothesis tests and the column matching approach, i.e. the relationship between forming the diagnosis statement S by using the decision structure, and by taking the intersection of the individual diagnosis statements S_k. For example, the sets S_k^0 and S_k^1 for the decision structure (3.13), are

$$\begin{aligned} S_1^0 &= \{\mathbf{NF}, \mathbf{F_1}, \mathbf{F_2}, \mathbf{F_3}\} & S_1^1 &= \{\mathbf{F_2}\} \\ S_2^0 &= \{\mathbf{NF}, \mathbf{F_1}, \mathbf{F_2}\} & S_2^1 &= \{\mathbf{F_2}, \mathbf{F_3}\} \\ S_3^0 &= \{\mathbf{NF}, \mathbf{F_1}, \mathbf{F_2}, \mathbf{F_3}\} & S_3^1 &= \{\mathbf{F_1}, \mathbf{F_3}\} \end{aligned}$$

That is, the set S_k^0 contains all system behavioral modes that have 0 or X in row δ_k of the decision structure. Also S_k^1 contains all system behavioral modes that have 1 or X in the same row. In this way, the decision structure can be seen as an overview of a diagnosis system based on structured hypothesis tests. When the result of a test is S_k^0, then the system behavioral modes with 0:s and X:s in the decision structure, are the possible present system behavioral modes. When the result is S_k^1, then the system behavioral modes with 1:s and X:s are the possible present system behavioral modes.

From a decision structure we can also read out which tests that will respond, i.e. which null hypothesis that will be rejected, when a particular system behavioral mode is present. For the decision structure (3.13), we know that if NF is the present fault mode, then *no* tests will respond, because the corresponding column has only zeros. Also, if F_3 is the present system behavioral mode, then test δ_1 will *not* respond, test δ_2 *will* respond, and test δ_3 *may* respond.

3.4.5 Performance Issues

The principle of structured hypothesis tests, and also the column matching approach, has the advantage that several fault modes per component can be handled easily. One further advantage is that the isolation logic becomes very simple.

One disadvantage of structured hypothesis tests, and the column matching approach, is that multiple faults are not handled in a very efficient way. Assume for example that we have a system with 20 components that are to be diagnosed. Assume also that each of these components have two possible fault modes. This means that the total number of possible system behavioral modes becomes $3^{20} \approx 3.5 \cdot 10^9$. The size of the sets S_k^1 and S will typically be of almost the same order of magnitude. Clearly this makes both the storage of S_k^1 and S, and the intersection operation (3.17) intractable. Thus, the principles of structured hypothesis tests and column matching are only solutions for relatively small systems or when only a limited multiplicity of fault is considered.

To see how a limitation of the multiplicity of faults affects the number of system behavioral modes, consider again our example system with 20 components but assume this time that only single or double faults are considered. The total number of system behavioral modes becomes $1 + 20 \cdot 2 + \binom{20}{2} \cdot 4 = 801$. Also the sets S_k^1 becomes much smaller. For example, a diagnostic test δ_i testing the null hypothesis that two components are in the NF-mode, will have a sub-diagnosis statement S_i^1 with 80 elements. In many applications, this number should be small enough so that the storage of S_k and S, and the intersection operation (3.17), can be handled.

The minimal hitting-set approach, to be presented in Section 3.5, will not be able to handle more than two behavioral modes per component. On the other hand, in many applications, it will have a better

performance with regards to memory and processing power usage.

3.4.6 Examples

This section contains three examples that illustrate how to construct a set of hypothesis tests for the purpose of using the fault isolation approach structured hypothesis tests. Systematic methods of constructing test quantities are described in Chapter 4.

Faults Modeled as Parameter Changes

Consider a process that can be modeled as

$$y(t) = \theta_1 u_1(t) + \theta_2 u_2(t) + \theta_3 u_3(t)$$

Faults are modeled as deviations in the constant parameters θ_i. The fault state vector is $\theta = [\theta_1\ \theta_2\ \theta_3]$. Four behavioral modes are considered:

NF	$\theta = [1\ 1\ 1]$
F_1	$\theta_1 \neq 1,\ \ \theta_2 = \theta_3 = 1$
F_2	$\theta_2 \neq 1,\ \ \theta_1 = \theta_3 = 1$
F_3	$\theta_3 \neq 1,\ \ \theta_1 = \theta_2 = 1$

To diagnose this system, we use four hypothesis tests whose null hypotheses are defined by the sets M_k:

$$\begin{aligned} M_0 &= \{NF\} \\ M_1 &= \{NF, F_1\} \\ M_2 &= \{NF, F_2\} \\ M_3 &= \{NF, F_3\} \end{aligned}$$

The null and alternative hypotheses become

$$\begin{aligned} H_k^0 &: F_p \in M_k \\ H_k^1 &: F_p \in M_k^C \end{aligned}$$

for $k = 0, 1, 2, 3$. Then we have that $S_k^1 = M_k^C$, and S_k^0 is chosen as $S_k^0 = \Omega$.

As test quantities, we use the functions

$$T_0(z) = \sum_{t=0}^{N} \big(y - \hat{y}\big)^2 = \sum_{t=0}^{N} \big(y - u_1 - u_2 - u_3\big)^2 \tag{3.18a}$$

$$T_1(z) = \min_{\theta_1} \sum_{t=0}^{N} \big(y - \hat{y}\big)^2 = \min_{\theta_1} \sum_{t=0}^{N} \big(y - \theta_1 u_1 - u_2 - u_3\big)^2 \tag{3.18b}$$

$$T_2(z) = \min_{\theta_2} \sum_{t=0}^{N} \big(y - \hat{y}\big)^2 = \min_{\theta_2} \sum_{t=0}^{N} \big(y - u_1 - \theta_2 u_2 - u_3\big)^2 \tag{3.18c}$$

$$T_3(z) = \min_{\theta_3} \sum_{t=0}^{N} \big(y - \hat{y}\big)^2 = \min_{\theta_3} \sum_{t=0}^{N} \big(y - u_1 - u_2 - \theta_3 u_3\big)^2 \tag{3.18d}$$

Note that these functions are in principle parameter estimators and that $T_k(z)$ is the sum of squared simulation errors. It is obvious that the functions (3.18) are small when the present system behavioral mode belongs to the corresponding set M_k. For example if F_1 is the present system behavioral mode, then $T_1(z)$ will produce a good estimate of θ_1 which implies that the simulation error and $T_1(z)$ will become small. Also, for at least "large" faults, the functions (3.18) are large when the present system behavioral mode does not belong to the corresponding set M_k. For example if F_1 is the present system behavioral mode, and the fault is "large", then $T_0(z)$, $T_2(z)$, and $T_3(z)$ will all become large. All this means that the functions (3.18) satisfy our requirements on test quantities.

Faults Modeled as Arbitrary Input Signals

Consider a process that can be modeled as

$$\begin{aligned} x(t+1) &= Ax(t) + B\big(u(t) + f_u(t)\big) \\ y_1(t) &= C_1 x(t) + f_1(t) \\ y_2(t) &= C_2 x(t) + f_2(t) \end{aligned}$$

The faults are modeled as the signals f_u, f_1, and f_2, representing an actuator fault and faults in sensor 1 and 2 respectively. The fault state vector is $\theta = [f_u(t)\ f_1(t)\ f_2(t)]$. Four system behavioral modes are

considered:

$$\begin{array}{ll} NF & \theta = [0\ 0\ 0] \\ F_u & \theta = [f_u(t)\ 0\ 0],\quad f_u(t) \not\equiv 0 \\ F_1 & \theta = [0\ f_1(t)\ 0],\quad f_1(t) \not\equiv 0 \\ F_2 & \theta = [0\ 0\ f_2(t)],\quad f_2(t) \not\equiv 0 \end{array}$$

To diagnose this system, we use two hypothesis tests with hypotheses

$$H_1^0 : F_p \in M_1 = \{NF, F_1\}$$
$$H_1^1 : F_p \in M_1^C = \{F_u, F_2\}$$

$$H_2^0 : F_p \in M_2 = \{NF, F_2\}$$
$$H_2^1 : F_p \in M_2^C = \{F_u, F_1\}$$

To calculate the test quantities, we first use the following two observers

$$\hat{x}(t+1) = A\hat{x}(t) + Bu(t) - K(y_1(t) - \hat{y}_1(t)) \tag{3.19a}$$
$$\hat{y}_1(t) = C_1\hat{x}(t) \tag{3.19b}$$

$$\hat{x}(t+1) = A\hat{x}(t) + Bu(t) - K(y_2(t) - \hat{y}_2(t)) \tag{3.20a}$$
$$\hat{y}_2(t) = C_2\hat{x}(t) \tag{3.20b}$$

Then the test quantities can be defined as

$$T_1(z) = \sum_{t=0}^{N} |y_2(t) - \hat{y}_2(t)|$$
$$T_2(z) = \sum_{t=0}^{N} |y_1(t) - \hat{y}_1(t)|$$

These test quantities $T_k(z)$ are zero or small if the present system behavioral mode belongs to the corresponding sets M_k. For example, if F_1 is the present system behavioral mode, then the observer (3.20) will produce a good estimate $\hat{y}_2(t)$ since the calculation of $\hat{y}_2(x)$ is not affected by a fault in sensor 1. This means that $T_1(z)$ will become

small. Also when F_1 is present, it can be shown that $T_2(z)$ will become large or at least non-zero. This means that $T_1(z)$ and $T_2(z)$ serves well as test quantities. This configuration of observers, in which each observer is fed by only one of the output signals, is called a *dedicated observer scheme*(Clark, 1979).

The Polybox Example

The Polybox example is a standard example within the AI-field of model-based diagnosis research (Kleer and Williams, 1987). Here, this example is discussed in the perspective of SHT, and will exemplify the handling of multiple faults.

The system consists of five components: three multipliers M1, M2, and M3, and two adders A1 and A2. The components are connected according to Figure 3.4. The observations consists of one sample of all inputs a, b, c, d, and e, and the outputs f and g, i.e. a vector of 7 elements. The complete list of single-fault system behavioral modes is:

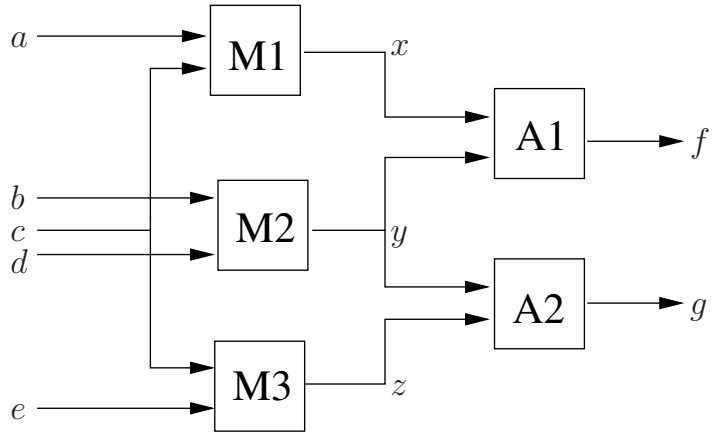


Figure 3.4: The Polybox example.

NF	no-fault
A1	arbitrary fault in component A1
A2	arbitrary fault in component A2
M1	arbitrary fault in component M1
M2	arbitrary fault in component M2
M3	arbitrary fault in component M3

Since no knowledge about the faults is known, a very general fault model is adopted. For example, the model of the adder A1 is

$$f = x + y + \mathbf{f_f} \tag{3.21}$$

where $\mathbf{f_f}$ is an arbitrary unknown constant. It is assumed that when a fault is present in component A1, $\mathbf{f_f} \neq 0$, and otherwise $\mathbf{f_f} = 0$. Further on, for example multiplier M1 is modeled as

$$x = ac + \mathbf{f_x} \tag{3.22}$$

By putting together all the component models, we obtain one model $\mathcal{M}(\theta)$ with $\theta = [\mathbf{f_f}\ \mathbf{f_g}\ \mathbf{f_x}\ \mathbf{f_y}\ \mathbf{f_z}]$, i.e. θ is a constant vector with 5 scalar elements. Depending on the specific fault, θ then takes different values. For example in the fault free case (i.e. the system behavioral mode NF is present), $\theta = [0\ 0\ 0\ 0\ 0]$ and the model describing the system is then

$$f = ac + bd \tag{3.23}$$
$$g = bd + ce \tag{3.24}$$

Another example is the combined model for the system behavioral mode A1:

$$f = ac + bd + \mathbf{f_f} \tag{3.25}$$
$$g = bd + ce \tag{3.26}$$

where $\theta = [f_f\ 0\ 0\ 0\ 0] \neq 0$. The set of all possible θ-values is $\Theta = \mathcal{R}^5$.

To limit the discussion, only multiple-fault system behavioral modes with two faulty components are considered. The models for the multiple-fault system behavioral modes are obtained by combining the models for the single-fault behavioral modes. For example the model for the system behavioral mode A1&M1 is

$$f = ac + \mathbf{f_x} + bd + \mathbf{f_f} \tag{3.27}$$
$$g = bd + ce \tag{3.28}$$

The set of all system behavioral modes considered is then:

$$\Omega = \{NF, A1, A2, M1, M2, M3, A1\&A2, A1\&M1, A1\&M2, A1\&M3, A2\&M1, A2\&M2, A2\&M3, M1\&M2, M1\&M3, M2\&M3\}$$

By studying the models for the different system behavioral modes, it can be realized that all system behavioral modes in the set $\{A1, M1, A1\&M1\}$ are represented by in principle equivalent models. Compare for example (3.25) and (3.27). This is true also for the sets $\{A2, M3, A2\&M3\}$ and $\{A1\&A2, A2\&M1, A2\&M2, M1\&M2, M1\&M3, M2\&M3\}$. The implication is that the only possible hypotheses to test are the following four:

$$H_0^0 : F_p \in \{NF\}$$
$$H_0^1 : F_p \notin \{NF\}$$

$$H_1^0 : F_p \in \{NF, A1, M1, A1\&M1\}$$
$$H_1^1 : F_p \notin \{NF, A1, M1, A1\&M1\}$$

$$H_2^0 : F_p \in \{NF, M2\}$$
$$H_2^1 : F_p \notin \{NF, M2\}$$

$$H_3^0 : F_p \in \{NF, A2, M3, A2\&M3\}$$
$$H_3^1 : F_p \notin \{NF, A2, M3, A2\&M3\}$$

The first of these hypothesis tests does not help us in the isolation task, but can be useful for detecting the faults.

An example of test quantities for the four hypothesis tests is

$$T0 = |f - ac - bd| + |g - bd - ce|$$
$$T1 = |g - bd - ce|$$
$$T2 = |f - g - ac + ce|$$
$$T3 = |f - ac - bd|$$

3.5 Minimal Hitting Set Approach

As was seen in Sections 3.3 and 3.4, isolation could be performed by using a column matching approach or an intersection operation. As concluded above, when no limitation is made on the multiplicity of faults, both these approaches have complexity problems when the number of components grows. In this section, an alternative approach to the calculation of the diagnosis statement S is presented.

For the approach to work we will assume only one fault mode per component and no fault models. It may seem strange to assume *no fault models*, considering that fault modelling was described in Section 2.7 as important to be able to perform fault isolation. But, for example the general fault models in Section 2.7.1 can be said to imply that we have no model for the faulty behavior since there is no restriction on the, for example, sensor behavior in case of a fault. With such assumptions we can construct an efficient algorithm for calculating the diagnosis statement S.

In the approach, the null hypotheses are represented as sets of components. Then a so called *minimal hitting set* algorithm can be used to compute the set of minimal diagnoses. In the case of only two behavioral modes per component and no fault models, we have learned from Section 2.4.1 that these minimal diagnoses characterize the set of all diagnoses, i.e. the diagnosis statement.

3.5.1 Conflicts

Using the minimal hitting set approach, the concept of *conflicts* is important since it is the basis for computing the diagnoses. A conflict is some assumption of behavioral modes that is *not* consistent with the observations. Even though the minimal hitting set approach is limited to the case of only two behavioral modes per component and no fault models, the concept of conflicts is more general and has a wider usage. Therefore we will in this section first present conflicts without the limitation to only two behavioral modes per component and no fault models.

Formally conflicts can be defined as follows.

Definition 3.1 (Conflict)**.** A set of behavioral mode assignments $\mathcal{C}$

(possibly partial) is a *conflict* if

$$\mathcal{M} \cup \mathcal{O} \cup \mathcal{C}$$

is not consistent where $\mathcal{M}$ is a diagnostic model and $\mathcal{O}$ the set of observations.

Similar to the definition of kernel diagnoses, *minimal conflicts* can be defined as follows:

Definition 3.2 (Minimal Conflict)**.** A conflict $\mathcal{C}$ is minimal if the only conflict implied by $\mathcal{C}$ is $\mathcal{C}$ itself.

Example 3.6 Suppose that we have two inverters in the configuration illustrated in Figure 3.5. When the observations are as indicated in

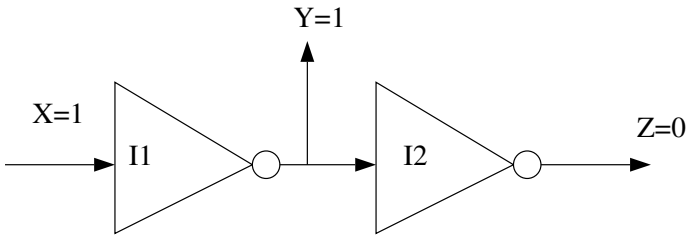


Figure 3.5: A simple inverter circuit.

the figure, i.e. $X = 1$, $Y = 1$, and $Z = 0$, we have for example the following conflicts:

$$OK(I_1) \wedge OK(I_2)$$
$$OK(I_1)$$

The first conflict implies the second, and can therefore not be minimal. The second conflict is minimal since it does not imply any other conflicts. Note here that $OK(I_2)$ is not a conflict because $OK(I_2)$ is consistent with the observations. By assuming that the inverters can have the behavioral modes described for the inverter in Section 2.2.1, the set of

conflicts increase.

$$
\begin{array}{ll}
OK(I_1) \wedge SA0(I_2), & OK(I_1) \wedge SHORT(I_2) \\
OK(I_1) \wedge U(I_2), & SA0(I_1) \\
SA0(I_1) \wedge OK(I_2) & SA0(I_1) \wedge SHORT(I_2) \\
SA0(I_1) \wedge SA0(I_2) & SA0(I_1) \wedge U(I_2) \\
SHORT(I_2) & SHORT(I_1) \wedge SHORT(I_2) \\
U(I_1) \wedge SHORT(I_2) &
\end{array}
$$

In total, the set of minimal conflicts are:

$$OK(I_1), \quad SA0(I_1), \quad SHORT(I_2)$$

3.5.2 Relations Between Diagnoses and Conflicts

The relationship between conflicts and diagnoses is described in the following theorem:

Theorem 3.2. Suppose that $\{\pi_1, \pi_2, \dots\}$ is the set of *all conflicts*. Then the candidate $\mathcal{D}$ is a diagnosis if and only if

$$\{\neg\pi_1, \neg\pi_2, \dots\} \bigcup \mathcal{D}$$

consistent.

Proof. See (de Kleer et al., 1992) for a proof of a similar result. ■

In fact, the set of minimal conflicts contain all information needed so it is possible to state a stronger theorem compared to Theorem 3.2:

Theorem 3.3. Suppose that $\{\pi_1, \pi_2, \dots\}$ is the set of *all minimal conflicts*. Then the candidate $\mathcal{D}$ is a diagnosis if and only if

$$\{\neg\pi_1, \neg\pi_2, \dots\} \bigcup \mathcal{D} \tag{3.29}$$

is consistent.

Proof. See (de Kleer et al., 1992) for a proof of a similar result. ■

Example 3.7 Consider again Example 3.6. To illustrate Theorem 3.3, suppose that we want to test if $\mathcal{D}_1 = SHORT(I_1) \wedge OK(I_2)$ is a diagnosis. We take the set of negations of the minimal conflicts, which is

$$\{\neg OK(I_1), \neg SA0(I_1), \neg SHORT(I_2)\}$$

Then to test consistency of (3.29) we form the union with $\mathcal{D}_1$:

$$\{\neg OK(I_1), \neg SA0(I_1), \neg SHORT(I_2), SHORT(I_1) \wedge OK(I_2)\}$$

All sentences are trivially satisfied by the following assignment:

$$\begin{aligned} OK(I_1) &= F \\ SA0(I_1) &= F \\ SHORT(I_2) &= F \\ SHORT(I_1) &= T \\ OK(I_2) &= T \end{aligned}$$

This assignment is also consistent with the assumption that each component can only be in one behavioral mode at a time. Since all sentences are consistent, Theorem 3.3 implies that $\mathcal{D}_1$ is a diagnosis.

Suppose then that we want to test if $\mathcal{D}_2 = SHORT(I_1) \wedge SHORT(I_2)$ is a diagnosis. We form the union between the negated minimal conflicts and $\mathcal{D}_2$:

$$\{\neg OK(I_1), \neg SA0(I_1), \neg SHORT(I_2), SHORT(I_1) \wedge SHORT(I_2)\}$$

It is clearly not possible to assign a value to $\neg SHORT(I_2)$ such that the last two sentences evaluate to TRUE. Therefore, $\mathcal{D}_2$ can according to Theorem 3.3, not be a diagnosis.

3.5.3 Conflicts and Diagnoses Under MDH

We will now present a special case, in which the relation between conflicts and diagnoses becomes especially powerful. Assume first that only the behavioral modes corresponding to OK and $\neg OK$ are used. Secondly, assume that the behavioral mode $\neg OK$ has no model, which

means that conflicts can only contain the OK-mode.[3] With these two assumptions; the following result can be shown:

Lemma 3.4 (Sufficient conditions for MDH)**.** Assume that only behavioral modes OK and $\neg OK$ are used and that $\neg OK$ has no model, then the minimal diagnosis hypothesis(see Definition 2.3) holds.

See (de Kleer et al., 1992) for a proof.

A useful tool in this special case is to represent negated conflicts as sets. For example a conflict $OK(a) \wedge OK(b)$ becomes negated $\neg OK(a) \vee \neg OK(b)$. This negated conflict can be represented as

$$\{\neg OK(a), \neg OK(b)\}$$

Assume that a negated conflict is a proper superset of another negated conflict, e.g. $\{\neg OK(a), \neg OK(b), \neg OK(c)\} \supset \{\neg OK(a), \neg OK(b)\}$. This relation means that the conflict represented by the larger set implies the conflict represented by the smaller set, i.e. $OK(a) \wedge OK(b) \wedge OK(c) \rightarrow OK(a) \wedge OK(b)$. According to Definition 3.2, $OK(a) \wedge OK(b) \wedge OK(c)$ can then not be a minimal conflict. That is, with the set representation, a minimal conflict is a conflict that is not a proper superset of some other conflict.

To illustrate how the set representation can be useful for deriving diagnoses, consider first an example of how diagnoses are derived with logic sentences. Suppose that $OK(a) \wedge OK(b)$ and $OK(b) \wedge OK(c)$ are the only minimal conflicts. The negations of these are

$$\neg OK(a) \vee \neg OK(b)$$
$$\neg OK(b) \vee \neg OK(c)$$

The diagnoses are according to Theorem 3.3 the mode assignments that are consistent with both negated minimal conflicts. It can be realized that only the following mode assignments fulfill this requirement and are therefore also the diagnoses:

$$OK(a) \wedge \neg OK(b) \wedge OK(c)$$
$$OK(a) \wedge \neg OK(b) \wedge \neg OK(c)$$
$$\neg OK(a) \wedge \neg OK(b) \wedge OK(c)$$
$$\neg OK(a) \wedge \neg OK(b) \wedge \neg OK(c)$$
$$\neg OK(a) \wedge OK(b) \wedge \neg OK(c)$$

[3] This implies that fault exoneration, see 3.8, is implicitly *not* assumed, i.e. faults may be visible but not for sure.

Let us now do the same thing using the set representation of conflicts. We will also use the set representation of diagnoses demonstrated in Section 2.4.1. The two negated conflicts become

$$\{\neg OK(a), \neg OK(b)\} \tag{3.30a}$$
$$\{\neg OK(b), \neg OK(c)\} \tag{3.30b}$$

and the diagnoses

$$\{\neg OK(b)\} \tag{3.31a}$$
$$\{\neg OK(b), \neg OK(c)\} \tag{3.31b}$$
$$\{\neg OK(a), \neg OK(b)\} \tag{3.31c}$$
$$\{\neg OK(a), \neg OK(b), \neg OK(c)\} \tag{3.31d}$$
$$\{\neg OK(a), \neg OK(c)\} \tag{3.31e}$$

Now study the criterion of Theorem 3.3, which means that each diagnosis must be consistent with all of the negated conflicts. With the set representation, this corresponds to that each diagnosis in (3.31) must have a non-empty intersection with each of the negated conflicts (3.30). Let us now summarize this conclusion in a theorem:

Theorem 3.5. Assume that each component has only the behavioral modes OK and $\neg OK$, and that the behavioral mode $\neg OK$ has no model. Let $\{\neg\pi_1, \neg\pi_2, \dots\}$ be the set of negated minimal conflicts (each negated conflict $\neg\pi_i$ is a set $\neg\pi_i = \{\neg\pi_{i1}, \neg\pi_{i2}, \dots\}$). If it for the candidate $\mathcal{D}$ (also a set) and all $\neg\pi_i$ holds that

$$\neg\pi_i \cap \mathcal{D} \neq \emptyset$$

then $\mathcal{D}$ is a *diagnosis*.

See (de Kleer et al., 1992) for a proof.

As said in Section 2.4.1, the set of minimal diagnoses is a powerful characterization of all diagnoses if the *Minimal Diagnosis Hypothesis* holds, a slightly modified version of the theorem above can in fact be used to derive minimal diagnoses:

Theorem 3.6. Assume that each component has only the behavioral modes OK and $\neg OK$, and that the behavioral mode $\neg OK$ has no model. Let $\{\neg\pi_1, \neg\pi_2, \dots\}$ be the set of negated minimal conflicts

(each negated conflict π_i is a set $\neg\pi_i = \{\neg\pi_{i1}, \neg\pi_{i2}, \dots\}$). If the candidate $\mathcal{D}$ (also a set) is a *minimal* set such that for all $\neg\pi_i$ it holds that

$$\neg\pi_i \cap \mathcal{D} \neq \emptyset$$

then $\mathcal{D}$ is a *minimal diagnosis*.

See (de Kleer et al., 1992) for a proof.

3.5.4 The Working Principle of the Algorithm

Based on the concept of conflicts, we will in this section present an algorithm that given a set of conflicts computes all minimal diagnoses. First we illustrate the ideas by considering an example with four diagnostic tests and four components. A decision structure could be set up as follows:

	NF	$F1$	$F2$	$F3$	$F4$	$F1\&F2$	$F1\&F3$	$F1\&F4$	$\dots F1\&F2\&F3\&F4$
δ_1	0	X	0	0	X	X	X	X	... X
δ_2	0	X	X	0	0	X	X	X	... X
δ_3	0	0	X	X	0	X	X	0	... X
δ_4	0	0	X	0	0	X	0	0	... X

(3.32)

Note that the assumption of no fault model means that the decision structure can not contain any 1:s. Note also that the X:s in a multiple-fault column is the "union" of the X:s in the corresponding single fault columns.

In this section we will use the word *candidate* for a system behavioral mode that is an element of the diagnosis statement S. The fault isolation can now be performed as in the following example. First assume that the diagnostic test δ_2 has responded, i.e. its null hypothesis has been rejected. Assume also that δ_2 is the only test that has responded. We see in the decision structure (3.32) that the single faults $F1$ and $F2$ are candidates. By construction of the table, we also know directly that all multiple faults including $F1$ or $F2$, e.g. $F1\&F2$ and $F2\&F3$, are the remaining candidates. Then if also δ_3 responds, the set of candidates, i.e. S, is updated. The single fault $F1$ can no longer be a candidate, but instead all multiple faults including $F1\&F3$ become candidates. The old candidate $F2$, and all multiple faults including $F2$, are still candidates. In this way the set of candidates can be updated

for every new diagnostic test that responds. This is in principle the approach used by diagnosis algorithms such as (Reiter, 1987; Kleer and Williams, 1987), developed within the field of AI.

To more precisely describe the algorithm, we need a different representation of the null hypotheses H_k^0, the sub-diagnostic statement S_k^1, and the diagnosis statement S. First, the null hypothesis is represented as a set C_k of those components that in the null hypothesis are assumed to be non-faulty. Second, the sub-diagnosis statement S_k^1, which corresponds to the alternative hypothesis H_k^1, is represented by the same set C_k but with the opposite interpretation, namely that some component in the set must be faulty. Third, the diagnosis statement S is represented using a set of sets, where each set represents a minimal diagnosis.

When H_k^0 is rejected the set C_k becomes a conflict. Then if we collect such conflict sets from all the tests with a rejected null hypothesis, we obtain a set $\prod$ of conflict sets. A diagnosis is then, according to Theorem 3.5, a set that has a non-empty intersection with every conflict set in $\prod$. Further, a minimal diagnosis is, according to Theorem 3.6, a minimal set that has a non-empty intersection with every conflict set in $\prod$.

The task of computing all minimal diagnoses, which also characterize all diagnoses, can then be transformed to the problem of finding all minimal hitting sets to a given set of sets. An algorithm for this will in the next section be written down explicitly. Here we will first illustrate the working principle on an example.

The minimal diagnosis computation is most easily illustrated using a subset-superset lattice. Figure 3.6 shows such a lattice for the Polybox example. Each node represents a candidate and for example $[M1, M2]$ means $\neg OK(M1) \wedge \neg OK(M2)$. The edges in the figure represent subset/superset relationship between candidates. The set of minimal diagnoses is incrementally computed as follows. Whenever a new conflict is detected, any previous minimal diagnosis that does not explain the new conflict is replaced by one or more superset diagnoses that are minimal based on this new information. This is accomplished by replacing any invalidated minimal diagnosis by a set of new candidates, each of which contains the old minimal diagnosis and one assumption from the new conflict. Note that these new candidates are diagnoses by construction. However, the new diagnoses need not be minimal. Therefore, any of the new diagnoses that is a

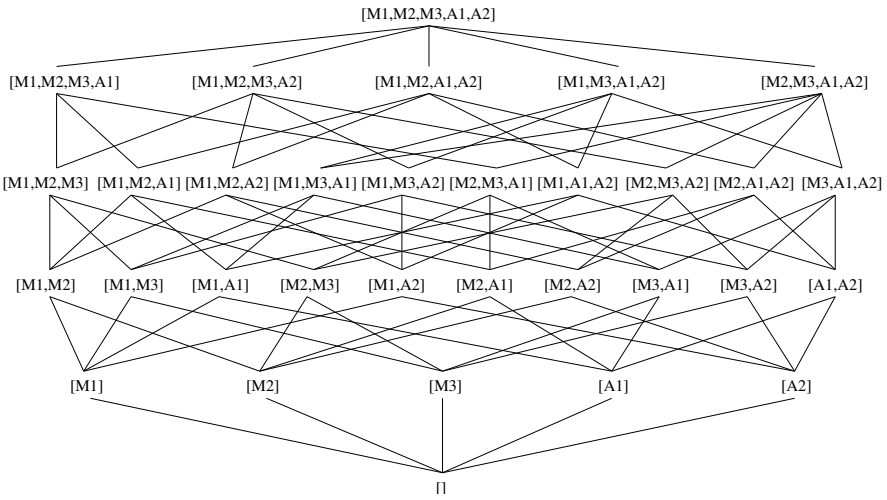


Figure 3.6: Subset-superset lattice for the Polybox example.

superset of any other minimal diagnosis, or is duplicated by another, is eliminated. The remaining diagnoses are minimal and are added to the set of minimal diagnoses. This procedure is then iterated for any conflict not processed. Note that the lattice in Figure 3.6 is only used to illustrate the procedure, the algorithm do not need to represent the whole lattice. This is fortunate since the lattice grows exponentially in size with number of components.

The procedure is now thoroughly exemplified using the Polybox example where we assumed no fault models and that the only considered behavioral modes for each component are OK and $\neg OK$.

Initially, before any measurements are made, there are no conflicts and thereby any candidate is also a diagnosis. Thus, the set of minimal diagnoses is initially the set holding only an empty set, i.e. $\{\{\}\}$. In other words, all nodes in the lattice represents a diagnosis.

We assume that a first conflict recognized is $\{A1, M1, M2\}$. This conflict invalidates the single minimal diagnosis $\{\}$ since the conflict states that at least one of the components $A1$, $M1$, or $M2$ is faulty. This invalidated minimal diagnosis is then removed from the set of minimal diagnoses and the next step is to consider the immediate supersets containing the invalidated minimal diagnosis and one element from the conflict. These new sets are, by construction, diagnoses but not necessary minimal. In this case, the diagnoses are $\{A1\}$, $\{M1\}$, and

$\{M2\}$. None of these diagnoses are supersets of any of the old minimal diagnoses (since there were no old minimal diagnoses left after the empty diagnosis {} were invalidated). The new set of minimal diagnoses is therefore $\{\{A1\}, \{M1\}, \{M2\}\}$. This is illustrated in Figure 3.7. The

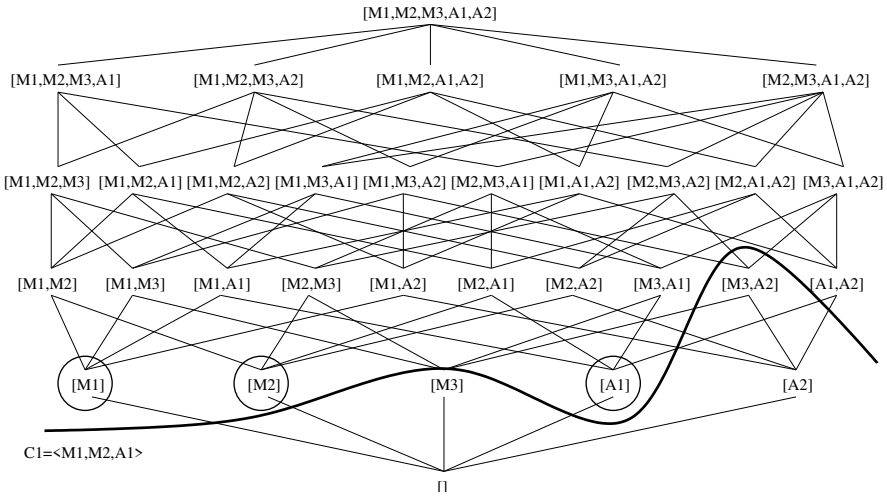


Figure 3.7: Subset-superset lattice after the conflict $\{A1, M1, M2\}$ has been processed. All candidates above the solid line are diagnoses and are represented by the new minimal candidates. The new minimal diagnoses are circled.

set of minimal diagnoses, indicated by circles, is a small representation of *all* diagnoses, i.e. all nodes in the lattice above the solid line.

The next conflict that is encountered in the Polybox example is $\{A1, A2, M1, M3\}$, inferred from the observation $g = 12$. The first step is again to see if any of the minimal diagnoses are invalidated by the new conflict. Using a similar result as Theorem 3.5, a diagnosis is invalidated if it has an empty intersection with a conflict. In this case, the minimal diagnoses $\{A1\}$ and $\{M1\}$ have non-empty intersections with the new conflict, but not $\{M2\}$. Thus, $\{M2\}$ is no longer a valid diagnosis. Similar to what was done when the empty diagnosis {} was invalidated by the first conflict, $\{M2\}$ is replaced by new diagnoses which are the immediate supersets of $\{M2\}$ with an additional element from the conflict, i.e. the new diagnoses are: $\{M2, A1\}$, $\{M2, A2\}$, $\{M2, M1\}$, and $\{M2, M3\}$. This is noted in Figure 3.8 where the new minimal diagnoses are circled. However, two of them are encircled with

a dashed line. This is because these two are supersets of the minimal diagnoses $\{M1\}$ and $\{A1\}$ respectively which makes these two diagnoses non-minimal. They can then be pruned from the new set of minimal diagnoses which then becomes $\{\{M1\}, \{A1\}, \{M2, M3\}, \{M2, A2\}\}$.

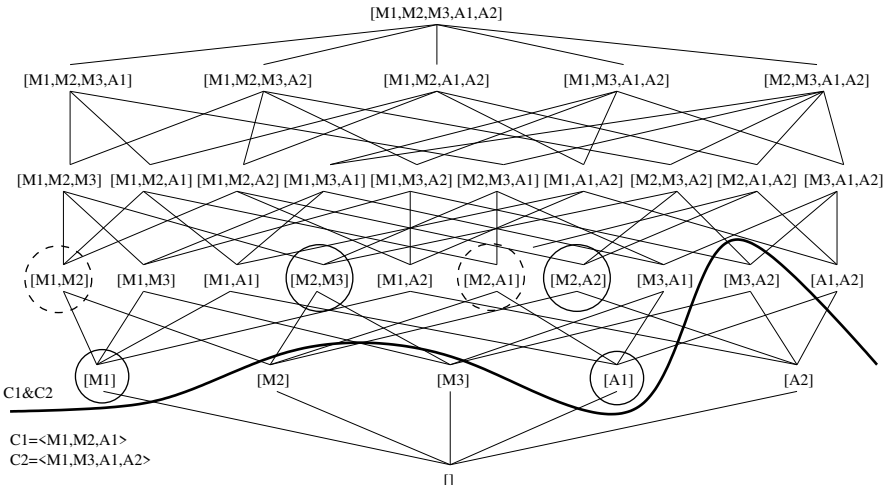


Figure 3.8: Subset-superset lattice after the second conflict $\{M1, M3, A1, A2\}$ has been processed. All candidates above the solid line are valid diagnoses and are represented by the new minimal diagnoses. The new minimal diagnoses are encircled and the two new diagnoses that were eliminated because of their non-minimality are dash-circled.

The working principle exemplified above can be summarized by the following steps:

1. Initialize the set of minimal diagnoses to hold only the empty set, i.e. $\{\{\}\}$.
2. Given a (new) conflict, find out if any minimal diagnosis is invalidated, i.e. has an empty intersection with the conflict.
3. Extend any invalidated diagnosis to a set of new diagnoses consisting of the invalidated diagnosis and an element from the new conflict.
4. Remove any new diagnoses that are not minimal, i.e. are supersets of any other minimal diagnosis.

5. Iterate from 2 for all new conflicts.

3.5.5 Algorithm Description

We will now present the minimal hitting set algorithm in a more algorithm-like form. The algorithm was originally described in the context of AI-diagnosis, see (Kleer and Williams, 1987).

Given a diagnosis statement S, in the form of a set of sets, and a sub-diagnosis statement S_k^1, in the form of a set, the following algorithm finds an updated set S_{new}.

Algorithm 1.
Input: a diagnosis statement S, and a sub-diagnosis statement S_k^1
Output: the updated diagnosis statement S_{new}
$S_{old} = S$
forall $D_i \in S$ do
  if $D_i \cap S_k^1 = \emptyset$ then
    Remove D_i from S_{old}
    forall $b \in S_k^1$ do
      $D_{new} := D_i \cup \{b\}$
      forall $D_j \in S_{old}$ do
        if $D_j \subseteq D_{new}$ then
          goto LABEL1
        endif
      next
      $S_{add} := S_{add} \cup \{D_{new}\}$
      LABEL1
    next
  endif
next
$S_{new} := S_{old} \cup S_{add}$

The algorithm is used in an iterative manner as follows. First when only one diagnostic test has responded, say δ_1, the diagnosis statement is simply $S = \{\{b\} \mid b \in S_1^1\}$, without using the algorithm. When a second diagnostic test responds, say δ_2, the updated diagnosis statement is obtained from the algorithm with the inputs S and S_2^1. For each further test δ_k that responds, the previous output S_{new} is used as input S and S_2^1 together with S_k^1.

Example

To illustrate the algorithm, consider again the example represented by (3.32). Assume first that only test δ_1 responds. The sub-diagnosis statement is $S_1^1 = \{F1, F4\}$, and as said above, S is calculated without using the algorithm resulting in $S = \{\{F1\}, \{F4\}\}$.

Assume then that also the test δ_2 responds. This means that the algorithm is called with the inputs $S = \{\{F1\}, \{F4\}\}$ and $S_2^1 = \{F1, F2\}$. In the first for-loop, assume first that $D_i = \{F1\}$. This means that the condition $D_i \cap S_k^1 = \emptyset$ is not fulfilled. Therefore, $\{F1\}$ is not removed from S_{old}. The interpretation of this is that $\{F1\}$ is still a candidate even after that the sub-diagnosis statement $S_2^1 = \{F1, F2\}$ has been considered.

Next, $D_i = \{F4\}$, and this time the condition $D_i \cap S_k^1 = \emptyset$ is fulfilled. This means that $\{F4\}$ is no longer a valid candidate. The second loop is entered and in the first step, $D_{new} := D_i \cup \{b\} = \{F1, F4\}$ is formed as a potential new candidate. In the third for-loop, it is checked if D_{new} is a superset of any of the old candidates in S_{old}. If this is the case it can be shown that D_{new} is redundant. In the example, $\{F1\}$ was not removed from S_{old} and it holds that $\{F1\} \subseteq \{F1, F4\}$. Therefore $D_{new} = \{F1, F4\}$ is redundant and does not need to be added to S_{add}. Finally, $D_{new} := \{F2, F4\}$ is formed and this time it does not hold that $\{F1\} \subseteq \{F2, F4\}$ and thus, $\{F2, F4\}$ is added to S_{add}. The output from the algorithm becomes

$$S_{new} := S_{old} \cup S_{add} = \{\{F1\}, \{F2, F4\}\}$$

Assume then that the test δ_3 responds. This means that the algorithm is called with the inputs $S = \{\{F1\}, \{F2, F4\}\}$ and $S_3^1 = \{F2, F3\}$. In the first for-loop, $\{F1\}$ is found to be not a candidate when the new sub-diagnosis statement S_3^1 is considered. This results in that $\{F1\}$ is removed from S_{old} but $\{F1, F2\}$ and $\{F1, F3\}$ are added to S_{add}. Then $\{F2, F4\}$ is found to be still a candidate and is kept in S_{old}. The output from the algorithm becomes

$$S_{new} := S_{old} \cup S_{add} = \{\{F2, F4\}, \{F1, F2\}, \{F1, F3\}\}$$

If no more diagnostic tests responds, this is our final diagnosis statement. Note that the set of all system behavioral modes represented by this

expression is

$$S_{new} := \{F2\&F4, F1\&F2, F1\&F3, F1\&F2\&F3, \\ F2\&F3\&F4, F1\&F2\&F4, F1\&F3\&F4, F1\&F2\&F3\&F4\}$$

3.5.6 Connection to Structured Hypothesis Tests

In this section we take a closer look at the connection between the minimal hitting set approach and structured hypothesis tests. We let both S_k^1 and S be represented as sets of behavioral modes in accordance with Section 3.4.1.

Firstly, remove from each S_k^1 all the multiple faults. We denote these new sets obtained with $\bar{S}_k^1$. The idea is that from the assumption of no fault models, the single faults in $\bar{S}_k^1$ are enough to represent the whole set S_k^1. For example, the set S_1^1 for the first diagnostic test in (3.32) is

$$S_1^1 = \{F1, F4, F1\&F2, F1\&F3, F1\&F4, F2\&F4, F3\&F4, F1\&F2\&F3, \\ F2\&F3\&F4, F1\&F2\&F4, F1\&F3\&F4, F1\&F2\&F3\&F4\}$$

The set $\bar{S}_1^1$ which contains only single faults becomes

$$\bar{S}_1^1 = \{F1, F4\}$$

It is seen that all system behavioral modes in S_1^1 can simply be derived from $\bar{S}_1^1$. For example, since $F1$ is an element of $\bar{S}_1^1$ it follows that all multiple faults containing $F1$ must be elements of S_1^1, e.g. $F1\&F2$, $F1\&F3$, $F1\&F3\&F4$, and $F1\&F2\&F3\&F4$.

The assumption of no fault model will make it possible to also simplify the representation of the diagnosis statement S in a similar manner. Note first that since there is no fault model, we can not exclude the possibility that a faulty component behaves exactly as it was fault-free. This means that if $F1\&F2$ is found to be a diagnosis, i.e. faults in components 1 and 2 can explain the observations, it is always the case that also $F1\&F2\&F3$ is a diagnosis, i.e. fault in components 1, 2, and 3 can explain the observations. Thus, if a system behavioral mode $F1\&F2$ is contained in S, then all system behavioral modes containing $F1\&F2$ must also be included in S, e.g. $F1\&F2\&F3$, and $F1\&F2\&F3\&F4$. In this sense, the system behavioral modes $F1\&F2\&F3$ and $F1\&F2\&F3\&F4$ are redundant and could therefore

be removed from S. If all such redundant system behavioral modes are removed, we obtain a set that we shall denote $\bar{S}$.

Consider now an example where only test δ_1 and δ_2 in (3.32) have responded. Using the formula (3.17), the diagnosis statement can be calculated as follows:

$$\begin{aligned} S = S_1^1 \cap S_2^1 = \{&F1, F4, F1\&F2, F1\&F3, F1\&F4, F2\&F4, F3\&F4, \\ &F1\&F2\&F3, F2\&F3\&F4, F1\&F2\&F4, F1\&F3\&F4, F1\&F2\&F3\&F4\} \cap \\ \{&F1, F2, F1\&F2, F1\&F3, F1\&F4, F2\&F3, F2\&F4, F1\&F2\&F3, \\ &F2\&F3\&F4, F1\&F2\&F4, F1\&F3\&F4, F1\&F2\&F3\&F4\} = \\ = \{&F1, F2\&F4, F1\&F2, F1\&F3, F1\&F4, F1\&F2\&F3, F2\&F3\&F4, \\ &F1\&F2\&F4, F1\&F3\&F4, F1\&F2\&F3\&F4\} \end{aligned}$$

As seen all the three sets S_1^1, S_1^1, and S becomes relatively large. Note then that these sets can equally well be represented by the much smaller sets $\bar{S}_1^1$, $\bar{S}_1^1$, and $\bar{S}$:

$$\begin{aligned} \bar{S}_1^1 &= \{F1, F4\} \\ \bar{S}_2^1 &= \{F1, F2\} \\ \bar{S} &= \{F1, F2\&F4\} \end{aligned}$$

In this discussion, it is clearly illustrated that to choose a representation based on $\bar{S}_k^1$ and $\bar{S}$, which corresponds to the representation in the minimal hitting set approach, is much more memory efficient than to use sets of system behavioral modes.

3.6 Isolability Properties of a Diagnosis System

In Section 2.6, an *isolability matrix* was introduced to illustrate the fault isolability properties of a given *system* or, perhaps more correctly, a given *model*. A similar table can be used to illustrate the isolability properties of a given *diagnosis system*, i.e. which faults can a given diagnosis system isolate from each other. The isolability properties of the model is an upper limit of what is possible while the isolability properties of a given diagnosis system may have substantially lower isolability possibilities. This may be by design if we for example is not

interested in isolating all faults or we do not have the computational power to run all needed tests in real-time. Another, very common, situation where the isolability performance of the diagnosis system does not correspond to the ideal performance is when the detectability properties of some of the tests are not as good as expected from the model. Thus, after a design it is of interest to evaluate the isolability performance of the diagnosis system.

For this, consider for example a diagnosis system where, after evaluating the performance of the residual generators, the following decision structure is used to draw conclusions.

	NF	F_1	F_2	F_3
$\delta_1(z)$	0	X	0	X
$\delta_2(z)$	0	0	X	X

In this section, only decision structures with X are considered but the reasoning is possible to extend also to cover general decision structures. Also, only single fault modes are considered.

In Section 2.6, when evaluating the isolability properties of the model, the isolability matrix was derived using observation sets. A similar approach is possible also for deriving the isolability matrix for a diagnosis system, but a more simple approach is here adopted. As in Section 3.4, the decision from a test T_k is S_k^1 in case of an alarm and S_k^0 in case of no alarm. With the restriction to only X-structures and single faults it holds that $S_k^0 = \Omega$, i.e. in case of no alarm no restriction of possible faults are concluded, and the diagnosis statement is formed as (3.17).

Then, for a given diagnosis system, a fault F_i can be said to be isolable from fault F_j if and only if there exists a subset of the tests in the diagnosis system such that when they alarm, the diagnosis statement S includes F_i but not F_j. Since all $S_k^0 = \Omega$, i.e. no restriction in case of no alarm, it holds that

$$\text{Fault } F_i \text{ is isolable from fault } F_j \Leftrightarrow \exists k.\ F_i \in S_k^1 \wedge F_j \notin S_k^1$$

Applying this rule to the decision structure above gives that the isola-

bility matrix becomes

	NF	F_1	F_2	F_3
NF	X	X	X	X
F_1	0	X	0	X
F_2	0	0	X	X
F_3	0	0	0	X

Thus, mode F_3 is uniquely isolable from all other modes, while neither mode F_1 nor F_2 is isolable from mode F_3. This means that isolability is *not* necessarily a symmetric relation.

3.7 Focusing

As noted in the beginning of this chapter, it is often interesting to compute not all diagnoses, but a more focused subset. Different principles for this exist in the literature, but minimal diagnoses and most probable diagnoses are the two most common. A special case of most probable diagnoses is so called minimal cardinality diagnoses, i.e. diagnoses, represented as sets, of minimal cardinality. How to compute minimal diagnoses has already been described in the context of the minimal hitting set algorithm. How to compute most probable diagnoses, or minimal cardinality diagnoses will not be described in this text but the interested reader is referred to (Kleer and Williams, 1989) and (de Kleer et al., 1992).

3.8 Exoneration

We end the chapter by discussing the concept of exoneration. Consider a component, that from its observation, seems to be non faulty. Another way of saying this is that the component does not violate its no-fault behavioral model. The question is:

> Can we draw the conclusion that this component really is in behavioral mode OK?

Suppose for example that an inverter is faulty and randomly outputs 0:s and 1:s. However, the random sequence happens to coincide with the correct outputs. In this case, it is impossible from the observations to conclude that the inverter is faulty. In other words, if we observe an

inverter that seems to be functioning correctly, we can never assume that it is not faulty. However, if we see that the input does not corresponds to the predicted output, the inverter can be concluded to be faulty.

The answer to the question posed above, is that it depends on the knowledge of how the fault-modes affect the components, i.e. the behavioral models for the fault modes. Assume that the behavioral model for one fault mode of the inverter, says that outputs are randomly generated. Then we can of course not say that an inverter, from its input-output seems to be non faulty, is in the behavioral mode OK. On the other hand, if the only fault mode is $SA0$ (stuck at 0), and we observe $in = 0$ and $out = 1$, then we can draw the conclusion that the inverter is in the behavioral mode OK.

To assume that a fault always visibly affects the input-output relation of a component is called *exoneration*. Depending on the application, one should choose to make the exoneration assumption or not. Remember that this choice is made by choosing the appropriate behavioral models. Consider for example the inverter from Section 2.2.1. Since we have the fault modes $SA0$ and U, defined by their behavioral models, we do *not* make the exoneration assumption.

Appendix

3.A Notation used

Ω	Set of all system behavioral modes.
$\mathcal{Z}$	Set of all possible observations.
$\delta(\cdot)$, S	δ is the decision function from observations to a diagnosis statement $S \subseteq \Omega$.
H_k^0, H_k^1, M_k	H_k^0 and H_k^1 are the null hypothesis and the alternative hypothesis respectively in the k:th hypothesis test.
$T_k(z)$	Test quantity for hypothesis test k. Normally, H_k^0 is rejected when $T_k(z) > J_k$ where J_k is a threshold.
Θ, θ, Θ_{F_i}	Set of possible fault states θ is denoted Θ. The set of θ that are possible in system behavioral mode F_i is denoted Θ_{F_i}.
$\mathcal{M}(\theta)$	Model of the process as a function of fault state.
$\mathcal{M}_{F_i}(\theta)$	Model of the process as a function of fault state when $\theta \in \Theta_{F_i}$.

Chapter 4
Design of Test Quantities

This chapter describes some general principles on how test quantities can be designed. Most of the principles presented are valid for all kinds of fault models.

4.1 Test Quantities are Model Validity Measures

For each behavioral mode γ, the system corresponds to a specific model. This model may also be dependent on the fault state θ if the set Θ_γ consists of more than one fault state. The model for behavioral mode γ is denoted

$$\mathcal{M}_\gamma(\theta) \tag{4.1}$$

and we have implicitly assumed that $\theta \in \Theta_\gamma$. For example, the loose contact behavioral mode, described in Example 2.8, might be abbreviated LC, and the corresponding model is then denoted $\mathcal{M}_{LC}(c_1(t))$.

From the previous chapter, we realize that the assumption (or

conclusion) we make in each hypothesis test can be written

$$F_p \in \begin{cases} M^C & \text{if } T(z) \geq J \\ \Omega & \text{if } T(z) < J \end{cases} \tag{4.2}$$

where F_p denotes the present behavioral mode and z the observations. In (4.2), we have again assumed that $S^0 = \Omega$. As said before, the test quantity $T(z)$ should be designed such that if the data come from a behavioral mode in M^C, then $T(z)$ should be large. On the other hand, if the data x match the hypothesis H^0, i.e. a behavioral mode in M can explain the data, then $T(z)$ should be small. This can be restated by using the notation of the model (4.1):

> The test quantity $T(z)$ should be low if the data z match any of the models $\mathcal{M}_\gamma(\theta)$, $\gamma \in M$, and large otherwise.

Thus the test quantity $T(z)$ can be seen as a measure of the validity of the models $\mathcal{M}_\gamma(\theta)$ with respect to the observations z.

When constructing a diagnosis system, the main task is usually to construct model validity measures for models $\mathcal{M}_\gamma(\theta)$. Several principles for constructing such measures exists and we will in the following sections discuss four of them: using the *prediction error*, using *residuals*, using *parameter estimates*, and using the *likelihood function*. Unfortunately, there is no approach that is always the best choice and this has to be determined case by case based on experience or trial and error. Also note that although the basic idea of these principles are different, it can very well happen that, in some specific cases, the derived expressions for $T(z)$ equal each other.

How to compute the test quantity $T(z)$ is sometimes referred to as the *computational form* because it states exactly how to compute the test quantity. As will be showed in detail later on in this chapter, and also in the chapters that follows, it is also possible to derive expressions describing how faults influence the test quantity. Such expressions can be used to determine which faults that are easy to detect in the test quantity etc. and is called the *internal form* of the test quantity.

4.2 Test Quantities Based on Prediction Errors

With the view that the test quantity should be a model validity measure, it is intuitive to determine the ability of a model to describe measured data by evaluating how well the model can *predict* future data.

Given a model $\mathcal{M}_\gamma(\theta)$, with a fixed θ and measurements z, we can make a prediction $\hat{y}(t)$ of the output $y(t)$. Then we can calculate a *prediction error* $y(t) - \hat{y}(t|\theta)$. Now assume that we have collected a set of measurements over a time window $t = 1 \dots N$, and we want to calculate a measure that reflects model validity over this period. It is then natural to use some norm of the prediction errors to measure the model validity. Thus, to measure the validity of the model $\mathcal{M}_\gamma(\theta)$, with respect to the observations z, we can use a function $V(\theta, z)$:

$$V(\theta, z) = \frac{1}{N} \sum_{t=1}^{N} \|y(t) - \hat{y}(t|\theta)\| \tag{4.3}$$

The norm $\|\cdot\|$ can for example be the quadratic norm. For notational convenience, we have here assumed unit time. If the model $\mathcal{M}_\gamma(\theta)$, with fixed θ, is the correct one, then $V(\theta, z)$ will be small and otherwise, hopefully, large.

Now remember the definition of Θ^0:

$$\Theta^0 = \bigcup_{\gamma \in M} \Theta_\gamma$$

If Θ^0 consists of only one value θ_0, a test quantity can be calculated as

$$T(z) = V(\theta_0, z) \tag{4.4}$$

This is now a model validity measure for the single model $\mathcal{M}_\gamma(\theta_0)$.

Consider now the case where Θ^0 consists of several values θ, i.e we can not compute the test quantity since we do not know which θ to insert into (4.4). A sensible approach is then to use the θ that makes the model describe data as well as possible. With the function $V(\theta, z)$ as a validity measure, the test quantity can then be calculated as

$$T(z) = \min_{\theta \in \Theta^0} V(\theta, z) \tag{4.5}$$

The test quantity is now a measure of the validity of any of the models $\mathcal{M}_\gamma(\theta)$, $\gamma \in M$, with respect to the observations z. Note also that all faults belonging to behavioral modes in M have been decoupled in the test quantity T.

The following four examples illustrate how test quantities calculated in the general way described by formula (4.5), can be used for different types of fault modeling.

Example 4.1 Consider a system that can be modeled as

$$y(t) = gu(t) + b + v(t) \qquad v(t) \in N(0, \sigma) \qquad \theta = [b, g]$$

where $v(t)$ and $v(t+k)$ are independent for $k \neq 0$.

Assume that we want to consider three behavioral modes:

NF	$g = 1, b = 0$	no fault
F_b	$g = 1, b \neq 0$	bias fault
F_g	$g \neq 1, b = 0$	gain fault

Further we want to design a test quantity for the hypotheses

$$H^0 : F_p \in \{NF, F_b\}$$
$$H^1 : F_p = F_g$$

For these hypotheses, Θ^0 becomes $\Theta^0 = \{[b, g] \mid g = 1\}$. By using the formulas (4.5) and (4.3), we get

$$T(z) = \min_{\theta \in \Theta^0} \frac{1}{N} \sum_{t=1}^{N} \|y(t) - \hat{y}(t|\theta, z)\| = \min_{b} \frac{1}{N} \sum_{t=1}^{N} \big(y(t) - \hat{y}(t|b, z)\big)^2 \tag{4.6}$$

The estimate $\hat{y}(t|b)$ (we have skipped the argument z) can be obtained as

$$\hat{y}(t|b) = u(t) + b$$

Inserting this expression into (4.6) means that the test quantity becomes

$$T(z) = \min_{b} \frac{1}{N} \sum_{t=1}^{N} (y(t) - u(t) - b)^2 \tag{4.7}$$

The minimization is simple since it can be shown that the minimizing value of b is

$$\hat{b} = \frac{1}{N}\sum_{t=1}^{N}(y(t) - u(t))$$

The test quantity (4.7) will be small under H^0 and thus the bias fault is decoupled in $T(z)$.

Now follows an example showing how the principle from this section can be applied to a change detection problem, i.e. the case where also the *change time* or the *fault occurrence time* is considered in the fault model.

Example 4.2 Consider a signal $y(t)$ which can be modeled as

$$y(t) = v(t) + a(t)$$

where $v(t)$ is independent and identically distributed (iid) sequence and $v(t) \sim N(0, \sigma)$. The functions $a(t)$ and $\sigma(t)$ are equal to the known constants a_0 and σ_0 in the fault free case, but can contain an abrupt change to unknown values a_1 or σ_1 if a fault occurs.

Assume that we want to consider three behavioral modes:

NF	no fault
F_a	an abrupt change in $a(t)$ at the time t_{ch}
F_σ	an abrupt change in $\sigma(t)$ at the time t_{ch}

This means that the fault state-vector can be described as $\theta = [t_{ch}, a_1, \sigma_1]$.

Further we want to design a test quantity for the hypotheses

$$H^0 : F_p \in \{NF, F_a\}$$
$$H^1 : F_p \in \{F_\sigma\}$$

By using the general expression (4.5), the test quantity becomes

$$T(z) = \min_{\theta\in\Theta^0} \frac{1}{N} V(\theta, z) = \min_{[t_{ch},a_1]} \sum_{t=1}^{N} (y(t) - \hat{y}(t|t_{ch}, a_1))^2$$

where

$$\hat{y}(t|t_{ch}, a_1) = \begin{cases} a_0 & \text{if } t < t_{ch} \\ a_1 & \text{if } t \geq t_{ch} \end{cases}$$

This test quantity will be small under H^0 and thus all faults belonging to F_a are decoupled in $T(z)$.

The next example illustrates how test quantities can be designed in a case where we have mixed types of fault models. In this example, one fault is modeled as an arbitrary input and another fault is modeled as a constant parameter.

Example 4.3 Consider a system that can be modeled as

$$\begin{aligned} x(t+1) &= ax(t) + u(t) \\ y(t) &= x(t) + f(t) \end{aligned}$$

Assume that we want to consider three behavioral modes:

NF	$a = 0.5,\ f(t) \equiv 0$	no fault
F_a	$a \neq 0.5,\ f(t) \equiv 0$	a fault in the dynamics
F_f	$a = 0.5,\ f(t) \not\equiv 0$	an arbitrary sensor fault

We decide to design test quantities for two hypothesis tests with the hypotheses

$$\begin{aligned} H_1^0 &: F_p \in \{NF, F_a\} & H_1^1 &: F_p = F_f \\ H_2^0 &: F_p = NF & H_2^1 &: F_p \in \{F_f, F_a\} \end{aligned}$$

The test quantity for the first test, using expression (4.5), becomes

$$T_1(z) = \min_a \frac{1}{N} \sum_{t=1}^{N} (y(t) - \hat{y}(t|a))^2 = \frac{1}{N} \sum_{t=1}^{N} (y(t) - \hat{a}y(t-1) - u(t-1))^2$$

where $\hat{a}$ is the least square estimate of a. For the second test, the set Θ_2^0 contains only one element. Thus, the test quantity becomes

$$T_2(z) = \frac{1}{N} \sum_{t=1}^{N} (y(t) - \hat{y}(t))^2 = \frac{1}{N} \sum_{t=1}^{N} (y(t) - 0.5y(t-1) - u(t-1))^2$$

Now assume that the present fault mode is F_a and H_2^1 is accepted but H_1^0 is not rejected, i.e. $T_1 < J_1$ and $T_2 > J_2$. This will imply that the diagnosis, under a single fault assumption, becomes

$$S = \{NF, F_f, F_a\} \cap \{F_f, F_a\} = \{F_f, F_a\}$$

That is, both F_f and F_a can explain the process behavior. However, it is unlikely that the arbitrary fault signal $f(t)$ behaves in such a way that the process output exactly matches the model $\mathcal{M}_{F_a}(\theta)$. Therefore, we may draw the conclusion that the behavioral mode F_a is the one present in the process.

Example 4.4 The following example shows how traditional in-range monitoring can be solved using the general formula (4.5). Assume that under a no-fault situation, a state x is limited in range, $c_l \leq x \leq c_h$. Assume further that x is measured using a sensor y as $y(t) = x(t)$. If no more models are available, a prediction of $y(t)$ can in any case be written

$$\hat{y}(t|c) = c \qquad c_l \leq c \leq c_h$$

By using the general expression (4.5), the test quantity becomes

$$T(z) = \min_{c_l \leq c \leq c_h} V(c, z) = \min_{c_l \leq c \leq c_h} |y(t) - \hat{y}(t|c)| =$$
$$= \begin{cases} 0 & \text{if } c_l \leq y(t) \leq c_h \\ y(t) - c_h & \text{if } y(t) > c_h \\ c_l - y(t) & \text{if } y(t) < c_l \end{cases}$$

This shows that traditional in-range testing can be seen as a special case of test quantities based on prediction errors.

The above example is also a clear illustration on how knowledge of range limitations of θ should be incorporated into the fault model to improve diagnosis performance. More specifically, without the knowledge $c_l < c < c_h$, the sensor y can not be diagnosed.

4.2.1 Minimization of $V(\theta, z)$

The procedure to compute (4.5), i.e. to minimize $V(\theta, z)$, has not been addressed so far. The technical details are not going to be addressed in detail here, but the interested reader is referred to general literature on optimization, e.g. (Luenberger, 1989), and system identification, e.g. (Ljung, 1999).

4.3 Test Quantities Based on Residuals

To use residuals is a common way to generate test quantities used for fault detection and fault isolation. The word residual is used in many contexts but in the context of fault diagnosis it can be loosely defined as *any* signal r that is ideally zero in the fault-free case and when non-monitored faults are present. This is not a formal definition and a formal definition for deterministic models will be introduced in Chapter 6. To be able to detect the monitored faults, a good and useful residual is also non-zero when a monitored fault is present.

Example 4.5 For example, for a system described by stable transfer operators

$$y = G_u(p)u + G_f(p)f$$

where p is the differentiation operator. A residual can then be generated by the following linear filter

$$r = W(p)(y - G_u(p)u)$$

where $W(p)$ is any stable linear filter. The filter $W(p)$ can for example be a low-pass filter to filter out measurement noise.

A residual is always a model validity measure for some model. For example, assume that we have a residual $r_1(t)$ which is zero for the behavioral modes NF, $F1$, and $F2$, but non-zero for other behavioral modes. It is clear that this residual is a model validity measure for the model

$$\mathcal{M}_\gamma(\theta), \qquad \gamma \in \{NF, F1, F2\}$$

or equivalently the null hypothesis

$$H^0 : \ F_p \in \{NF, F1, F2\}$$

If we have a residual with these properties, it is direct to produce a test quantity according to Figure 4.1. The residual $r(t)$ is a signal obtained by in some way filtering the observations $z(t)$. Then the test quantity T is some measure of the size of the residual $r(t)$. While $r(t)$ is usually considered to be a *signal*, i.e. a function of time, the test quantity will here, as in hypothesis testing, primarily be seen as a constant value. However, it is also possible to calculate one new test quantity T for each sample, and thus in these cases, we can view also T as a signal.

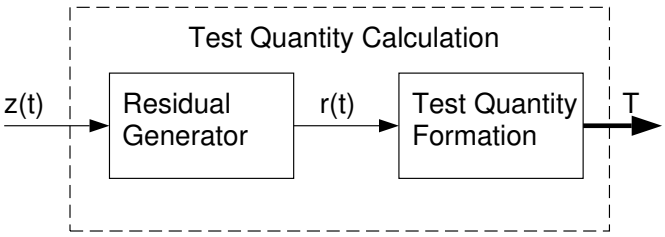


Figure 4.1: The calculation of the test quantity from a residual signal.

This implies that we also calculate one new diagnosis statement for every new sample. An example of forming a test quantity is to use the mean power of the residual in a predetermined time-window:

$$T = \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} r^2(t) dt$$

Residuals are mainly generated in one of two ways, either using so called *consistency relations* or using *observers* and this chapter only contains a general discussion about residual generation. However, more specific techniques will be given in Chapter 6, which discusses linear residual generation, and in Chapter 7, which discusses nonlinear residual generation.

Now follows an introduction to consistency relations and how they can be used to generate residuals.

4.3.1 Consistency Relations

A consistency relation[1] is any relation between known or measured variables that, in the fault free case, always holds. Because of this property, consistency relations are often the basis for residuals.

For example, consider the first order state-space model

$$\begin{aligned} \dot{x} &= -x + u \\ y &= x \end{aligned} \tag{4.8}$$

[1]Consistency relations is not the only term used in fault diagnosis literature. Other words that are common are parity relations, parity equations, parity functions, and analytical redundancy relations.

A consistency relation for this model can easily be derived by substituting y for x in the dynamic equation leading to

$$\dot{y} + y - u = 0 \tag{4.9}$$

The interpretation of this is that for any known signals y and u that are *consistent* with the model, meaning that they may have been generated by the model, will also satisfy the consistency relation. This further means that if the consistency relation does not hold, we know that model (4.8) can not explain measured data. Such reasoning can be used to detect and isolate faults.

More generally, g is a consistency relation if the following holds for all y and u that satisfies the original system equations (the model) when $f = 0$:

$$g(y, \dot{y}, \ddot{y}, \dots, u, \dot{u}, \ddot{u}, \dots) = 0 \tag{4.10}$$

For linear systems, a consistency relation can always be written as

$$Q(p)\begin{pmatrix} y \\ u \end{pmatrix} = Q_y(p)y + Q_u(p)u = 0 \tag{4.11}$$

where $Q_y(p)$ and $Q_u(p)$ are polynomial vectors (or matrices if multidimensional consistency relations are considered) in p. This is the case in (4.9) where $Q_y(p) = p + 1$ and $Q_u(p) = -1$. Consistency relations for linear systems are illustrated by the following example:

Example 4.6 Consider a linear state-space description

$$\dot{x} = -\alpha_1 x + \alpha_2 u \tag{4.12a}$$
$$y = \alpha_3 x \tag{4.12b}$$

which corresponds to the transfer operator $y = G(p)u$ with

$$G(p) = \frac{\alpha_2\alpha_3}{p + \alpha_1}$$

An equivalent description of the relation between y and u is given by the differential equation

$$\dot{y} + \alpha_1 y - \alpha_2\alpha_3 u = 0 \tag{4.13}$$

Relation (4.13) only includes known signals u and y and *not* the state x and is therefore a consistency relation for the system (4.12).

This also generalizes to the non-linear case as the next example shows.

Example 4.7 Consider a nonlinear system, described by state-space equations:

$$\begin{aligned} \dot{x} &= -x^2 + u \\ y &= x^3 \end{aligned}$$

A consistency relation for the model above can be stated as

$$\dot{y}^3 + 27\dot{y}y^2u + 27y^4 - 27y^2u^3 = 0 \tag{4.14}$$

This consistency relation was derived as follows. The measurement equation directly gives that

$$y - x^3 = 0$$

By differentiating both sides, another relation is obtained

$$\dot{y} - 3x^2\dot{x} = \dot{y} - 3x^2(-x^2 + u) = 0$$

Now we have two different relations which both include the unknown variable x. With a clever combination of these two relations the state-variable x can be eliminated. For higher order systems, it may be necessary to differentiate the measurement equation several times before obtaining a set of relations where it is possible to eliminate the state-variables. It is left as an exercise for the reader to validate the relation, e.g. by showing that a solution y, u to the original model satisfies (4.14).

Note that the elimination of the state variable x was easy in the linear case, where the relation could have been derived in the same way as in the nonlinear case by differentiating equations and eliminating the state. In the nonlinear case, the variable elimination problem is not so easy and this example used particularly easy nonlinearities (polynomials) where this eliminations is generally possible.

A problem with consistency relations for dynamic systems is that they often contain derivatives of signals, which makes them difficult to use directly as residuals. The following example illustrates this:

Example 4.8 Consider the linear model

$$y = G(p)u + f$$

where the transfer function $G(s)$ is

$$G(s) = \frac{1}{s^2 + as + b}$$

The time domain interpretation of the model is then:

$$\ddot{y} + a\dot{y} + by - u - \ddot{f} - a\dot{f} - bf = 0 \tag{4.15}$$

Equation (4.15) directly gives us a consistency relation, by examining the fault free case, i.e. by setting $f \equiv 0$ ($f = \dot{f} = \ddot{f} = 0$):

$$\ddot{y} + a\dot{y} + by - u = 0$$

and an equivalent description of the relation using the differentiation operator p:

$$(p^2 + ap + b)y - u = 0$$

It is clear that if $\ddot{y}$ and $\dot{y}$ were known, we could calculate $r = \ddot{y} + a\dot{y} + by - u$ which would be 0 in the fault free case and deviate from 0 when $f \not\equiv 0$. However, the higher order derivatives are usually not known and one way to circumvent this complication is to add, e.g. low-pass, dynamics to the consistency relation. That is, instead of computing the residual like $r = \ddot{y} + a\dot{y} + by - u$, compute the residual according to the differential equation

$$\ddot{r} + c_1\dot{r} + c_2 r = \ddot{y} + a\dot{y} + by - u$$

where constants c_1 and c_2 has been chosen to ensure a stable residual generator. In the frequency domain the residual generator transforms to

$$r = \frac{p^2 + ap + b}{p^2 + c_1 p + c_2} y - \frac{1}{p^2 + c_1 p + c_2} u$$

which can be realized on state-space form, i.e. higher order derivatives of y and u need not be used. The filter still has the property that $r = 0$

in the fault free case (after that influence from initial condition has disappeared).

In Example 4.8, it was stated that if higher order derivatives of y and u were known, the consistency relation could be used as a residual generator. However, this is normally not the case, but the consistency relation could nevertheless be used as a basis for the design of a residual generator by adding e.g. low-pass dynamics.

In fault diagnosis, we are looking for consistency relations where all unknown variables are eliminated. In all examples above, the only unknown variable was the state x. Usually also variables modeling non-monitored faults need to be eliminated. This will be discussed more in Chapter 6 and 7, where also systematic methods are given to derive linear and non-linear consistency relations.

4.3.2 Connection Between Residual Generation and Test Quantities Based on Prediction Errors

This section is a brief description of some connections between the general test quantities described in this chapter and some common techniques for residual generation.

First we show relations to linear techniques that will be described in detail in Chapter 6.

Example 4.9 Consider a system that can be modeled as

$$y_1 = \frac{1}{q^{-1}+1}(u+f_1) \tag{4.16}$$

$$y_2 = \frac{1}{q^{-1}+2}(u+f_1)+f_2 \tag{4.17}$$

Assume that we want to consider three behavioral modes:

NF	$f_1(t) \equiv 0, f_2(t) \equiv 0$	no fault
F_1	$f_1(t) \not\equiv 0, f_2(t) \equiv 0$	actuator fault
F_2	$f_1(t) \equiv 0, f_2(t) \not\equiv 0$	fault in sensor 2

Further we want to design a test quantity for a hypothesis tests with the hypotheses

$$H^0 : F_p \in \{NF, F_1\}$$
$$H^1 : F_p \in \{F_2\}$$

A linear residual generator for these hypotheses is

$$r = \frac{(q^{-1}+2)y_2 - (q^{-1}+1)y_1}{q^{-1}+3} \tag{4.18}$$

It will now be shown how the same residual can be obtained by using the general expression (4.5).

We assume here that the test quantity is recalculated for every new sample. The two-step approach is used and this means that we first have to estimate the parameter (now a signal) $f_1(t)$. From the model (4.16), the fault signal $f_1(t)$ can be estimated as

$$\hat{f}_1 = \arg\min_{f_1} \big(y_1 - \frac{1}{q^{-1}+1}(u+f_1)\big)^2 = (q^{-1}+1)y_1 - u \tag{4.19}$$

With this estimate and by using the expressions (4.3) and (4.5), the test quantity can now be calculated. We use an infinite window length, and to simplify the notation, we assume that the test quantity is calculated for $t = 0$:

$$T_1(z) = V(\hat{f}_1, z) = \sum_{t=-\infty}^{0} \|y_2(t) - \hat{y}_2(t|\hat{f}_1)\|$$

By means of the estimate (4.19), the prediction error can be expressed as

$$y_2 - \hat{y}_2(\hat{f}_1) = y_2 - \frac{1}{q^{-1}+2}(u+\hat{f}_1) = y_2 - \frac{q^{-1}+1}{q^{-1}+2}y_1$$

Then choose the measure $\|\cdot\|$ as

$$c_n q^n(\,\cdot\,)$$

where

$$\sum_{n=-\infty}^{0} c_n q^n = \frac{q^{-1}+2}{q^{-1}+3}$$

This means that

$$T_1(z) = \sum_{n=\infty}^{0} c_n q^n\big(y_2(t) - \hat{y}_2(t|\hat{f}_1)\big) = \frac{q^{-1}+2}{q^{-1}+3}\big(y_2(t) - \frac{q^{-1}+1}{q^{-1}+2}y_1(t)\big) = r$$

We have thus shown how the residual generator (4.18) can be obtained by using the general expression (4.5).

The next example shows how a residual, calculated with the help of an observer, can be expressed using the general expression (4.5).

Example 4.10 Assume that we have a non-linear model

$$\dot{x} = f(x,u) \tag{4.20}$$
$$y_1 = h_1(x,u) + f_1 \tag{4.21}$$
$$y_2 = h_2(x,u) \tag{4.22}$$

Here f_1 is a signal modeling a fault in sensor 1. Then assume that an observer for x can be constructed as

$$\dot{\hat{x}} = f(\hat{x},u) + K\big(y_2 - h_2(\hat{x},u)\big) \tag{4.23}$$

Then

$$r = y_2 - \hat{y}_2 = y_2 - h_2(\hat{x},u) \tag{4.24}$$

is a residual generator which will be insensitive to faults in sensor 1. This means that the corresponding null hypothesis is described by $M = \{NF, F_1\}$ where F_1 is the behavioral mode for $f_1 \not\equiv 0$. If we assume one new test quantity for each sample, we can see that r is similar to a test quantity constructed in accordance with formulas (4.3) and (4.5). However, according to the expression (4.5), the parameter f_1 should be implicitly estimated when calculating the test quantity. To see this, the formulas (4.3) and (4.5) as follows:

$$\begin{aligned} T(z) &= \min_{f_1} V(f_1,z) = \min_{f_1} |\mathbf{y} - \hat{\mathbf{y}}| = \min_{f_1} \left| \begin{matrix} y_1 - \hat{y}_1(f_1) \\ y_2 - \hat{y}_2 \end{matrix} \right| = \\ &= \min_{f_1} \left| \begin{matrix} y_1 - h_1(\hat{x},u) - f_1 \\ y_2 - h_2(\hat{x},u) \end{matrix} \right| = \left| \begin{matrix} y_1 - h_1(\hat{x},u) - f_1 \\ y_2 - h_2(\hat{x},u) \end{matrix} \right|_{f_1 = y_1 - h_1(\hat{x},u)} = \\ &= \left| \begin{matrix} 0 \\ y_2 - h_2(\hat{x},u) \end{matrix} \right| = |y_2 - h_2(\hat{x},u)| = |r| \end{aligned}$$

4.4 Test Quantities Based on the Likelihood Function

When the probability density functions of the noise is known, or can be assumed to be known, it is possible to use the likelihood function as model validity measure. The likelihood function is defined as

Definition 4.1 (Likelihood Function)**.** Let $f(\mathbf{z}|\theta)$ denote the probability density function of the sample $\mathbf{Z} = [Z_1,\ Z_2,\ \dots\ Z_n]$. Then, given that $\mathbf{Z} = \mathbf{z}$ is observed, the function of θ defined by

$$L(\theta|\mathbf{z}) = f(\mathbf{z}|\theta) \tag{4.25}$$

is called the *likelihood function*.

Thus, the likelihood function gives the "probability" (only formally correct for discrete distribution functions) that we observe the data $\mathbf{z}$ for a given θ. Note the important difference that the likelihood function is a function of θ while the probability distribution function is a function of $\mathbf{z}$.

Given a model, it is possible to set up a likelihood function that becomes a measure for how well the measured data matches the model. Recall from Section 4.1 that this is exactly what we want when constructing test quantities. This is also the reason why likelihood functions are a common choice for test quantities in general statistical hypothesis testing. Thus, when using the likelihood function as a model validity measure, the measure $V(\theta, z)$ in (4.5) corresponds to $L(\theta|\mathbf{z})$. In contrast to residuals, the likelihood function becomes large when measurement data matches the model and small when the data does not match the model. Thus, the null hypothesis H^0 should be rejected if $T(z) < J$. Note that $>$ has been changed to $<$, compared to previous cases.

If the set Θ^0 consists of only one element, then the likelihood function (4.25) can be used directly as a test quantity. When Θ^0 consists of several elements, we have to use optimization in accordance with (4.5). However, since the likelihood function becomes large when measurement data matches the model, the minimization must be replaced by maximization. The test quantity then becomes

$$T(\mathbf{z}) = \max_{\theta \in \Theta^0} L(\theta|\mathbf{z}) \tag{4.26}$$

This principle is usually called the *maximum likelihood*.

Often it is assumed that the data are independent and identically distributed such that

$$f(\mathbf{z}|\theta) = \prod_{i=1}^{N} f(z_i|\theta)$$

Here z_i means $z(t_i)$. This means that the likelihood function becomes

$$L(\theta|\mathbf{z}) = \prod_{i=1}^{N} f(z_i|\theta)$$

and thus, much simpler to calculate. A further simplification is obtained by using the *log-likelihood function* defined as

$$l(\theta|\mathbf{z}) = \ln\ L(\theta|\mathbf{z})$$

If the assumption about independent data is used, we get

$$l(\theta|\mathbf{z}) = \ln\ L(\theta|\mathbf{z}) = \ln \prod_{i=1}^{N} f(z_i|\theta) = \sum_{i=1}^{N} \ln\ f(z_i|\theta)$$

Note that since the logarithm function $\ln(z)$ is monotone, a hypothesis test based on the log-likelihood function $l(\theta|\mathbf{z})$ is equivalent to a test based on the basic likelihood function $L(\theta|\mathbf{z})$.

Example 4.11 Consider again Example 4.1 but instead of (4.6), we use the likelihood function to obtain the test quantity. Let z_i denote $y(i) - u(i)$ which means that $z_i \sim N(b, \sigma)$. The test quantity then becomes

$$T(z) = \max_{\theta \in \Theta^0} L(\theta|z) = \max_{b} \prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} \exp\{-\frac{(z_i - b)^2}{2\sigma^2}\}$$

Due to the independence assumption on $v(t)$, the log-likelihood version of this test quantity becomes

$$T'(z) = \max_{\theta \in \Theta^0} l(\theta|z) = \max_{b} \sum_{i=1}^{N} \ln\ \frac{1}{\sigma\sqrt{2\pi}} \exp\{-\frac{(z_i - b)^2}{2\sigma^2}\} =$$

$$= \max_{b} \left[-N\ \ln\ \sigma\sqrt{2\pi} - \frac{1}{2\sigma^2} \sum_{i=1}^{N} (z_i - b)^2 \right]$$

Note that the last expression contains a term independent of the optimization variable b. This term can be neglected and the remaining expression is then equivalent to.

$$\min_b \sum_{i=1}^{N} (z_i - b)^2 = \min_b \sum_{i=1}^{N} (y(i) - u(i) - b)^2$$

Now note that it happens, in this particular problem, that this expression is equal to the expression obtained in Example 4.1.

The drawback with using the likelihood functions, compared to prediction errors, is that to make the calculations tractable, we must usually assume that the data are independent and normally distributed. On the other hand the likelihood function is very universal. It can for example easily handle faults that are modeled as an increase in the variance of a signal.

4.5 Test Quantities Based on Parameter Estimates

A most natural way of detecting a fault is to estimate a parameter that is influenced by a fault, and then compare the estimated value with the nominal value of the parameter. Now we discuss how to use the estimated parameter itself, more directly, as a test quantity, i.e. a model validity measure.

One solution is to estimate an element θ_i of the fault state vector θ and then compare it with the nominal (i.e. no fault) value θ_i^0. We have here assumed that Θ_{NF} contains only one element, i.e. $\Theta_{NF} = \{\theta_0\}$.

Now first consider the case where the set Θ^0 consists of only one element. Then a test quantity can be constructed as

$$T(\mathbf{z}) = \|\hat{\theta}_i - \theta_i^0\| \qquad \hat{\theta}_i = \arg\min_{\theta_i} V'(\theta_i, \mathbf{z})$$

where $V'(\theta_i, \mathbf{z})$ is some model validity measure. This is a common solution used in literature, e.g. (Isermann, 1993).

When the set Θ^0 consists of more than one element, additional parameters have to be estimated. That is, in addition to estimating the parameter θ_i we also have to estimate the free parameters in Θ^0, i.e. the ones corresponding to faults that are decoupled. This means

that compared to the principles from Sections 4.2 and 4.4, i.e. using the prediction error or the likelihood function, one extra parameter must be estimated. This of course implies that decoupling might be more difficult.

Using estimates as model validity measure has both advantages and disadvantages compared to residuals and the likelihood function. Test quantities based on estimates can have very good performance for the behavioral mode corresponding to the estimated parameter. However for other behavioral modes, the performance might be quite bad and also highly dependent on the input signal.

An additional complication with parameter estimates is that the input must excite the system sufficiently to make sure that enough information about individual parameters is available in the measured data such that the estimates becomes reliable. To achieve this, it may be necessary to add perturbations to the input signal. However if other considerations are taken into account, it is not sure that for a specific application, it is desirable to choose such an input. This problem of sufficiently exciting inputs may get even worse for processes working in closed loop.

4.6 Robustness via Normalization

When constructing test quantities, a goal is that they should be insensitive to *uncontrolled effects* such as changes in inputs u and state x, disturbances d, model errors, etc. Normally, the constructed test quantities do not meet these goals perfectly. The reasons why the test quantities become sensitive to uncontrolled effects are:

- Approximate decoupling. Because of fundamental limitations it is sometimes impossible to completely decouple disturbances and effects of faults (i.e. the faults belonging to behavioral modes in the null hypothesis).
- Model errors. Most unmodeled disturbances, incorrect model structure, and unmodeled noise etc. implies that the performance of the test quantities is degraded. The most serious problem is usually that the significance level is raised.
- Modeled noise. Even though noise terms are included in the model, it is mostly impossible to avoid that the noise is going to

affect the test quantities.

The discussion above is closely related to the issue of *robustness*. More exactly, robustness can be defined as the ability of the test quantities to satisfy some specific performance goals while the uncontrolled effects are present to a certain degree. In connection with linear residual generation, methods to achieve and analyze robustness have been extensively studied, e.g. see (Chen and Patton, 1999; Frisk and Nielsen, 1999). In many of these methods, the robustness issue is a fundamental part of the design process for the test quantities. A somewhat different approach is to first design the test quantity without robustness considerations and then afterward consider robustness as an additional design step by adjusting and compensating the originally designed test quantity. In for example (Höfling and Isermann, 1996), there are experimental results showing performance of the latter robustness approach.

As a way to achieve and improve robustness by adjusting and compensating already designed test quantities, we will here consider *normalization*. Normalization is to compensate the test quantity for unmodeled effects by multiplying it with a cleverly chosen variable that is a function of the measured data x. Here we investigate normalization for the four methods of constructing test quantities described in the previous sections, namely to use prediction errors, residuals, the likelihood function, and parameter estimates.

4.6.1 Normalization When Using Parameter Estimates

The procedure and ideas are here best illustrated with an example.

Example 4.12 Consider a system which can be modeled as

$$y(t) = bu(t) + v(t)$$

where $v(t) \sim N(0, \sigma_v)$. The nominal (i.e. corresponding to the no fault case) value of b is b_0. We will use the notation U, Y, and V to denote column vectors of u, y, and v respectively.

Assume a test quantity based on a parameter estimate:

$$T_2(z) = (\hat{b} - b_0)^2 \qquad \hat{b} = \frac{1}{U^T U} U^T Y \tag{4.27}$$

where $\hat{b}$ is the least square estimate of b. Consider the fault free case, i.e. $b = b_0$, which means that

$$\hat{b} - b_0 = \frac{1}{U^T U} U^T (b_0 U + V) - b_0 = b_0 + \frac{1}{U^T U} U^T V - b_0 = \quad (4.28)$$

$$= \frac{1}{Np} U^T V \quad \sim N(0, \frac{\sigma_v}{\sqrt{Np}}) \quad (4.29)$$

where $Np = U^T U$, and p is the mean power of u. We see that $\hat{b} - b_0$ has a standard deviation that is dependent on u. If the mean power of u varies during operation, this leads to an undesirable situation where the significance level of the test will then depend on u. The solution is to use normalization and we therefore multiply (4.28) with $\sqrt{Np}$. Then we have that

$$\sqrt{Np}(\hat{b} - b_0) \ \sim N(0, \sigma_v). \quad (4.30)$$

The corresponding normalization for the test quantity (4.27) becomes

$$T_2'(z) = Np(\hat{b} - b_0)^2 \qquad \hat{b} = \frac{1}{U^T U} U^T Y \quad (4.31)$$

Thus, using (4.31) means that a fixed threshold will imply a fixed significance level independent on u.

In terms of robustness, a hypothesis test based on the normalized test quantity (4.31) and with a fixed threshold, will satisfy the performance goal that the significance level must not be above a certain level. This will hold for any u. However, there is no guarantee that other performance goals, such as the probability of $T_2'(z) > J_2$ when a fault is present, are satisfied.

It is also worth noting that the distribution of the estimate in most cases is impossible to compute exactly. One way to still, approximately, normalize the estimates is to use the asymptotic distribution of the estimate. By asymptotic it is meant the limiting distribution of the estimate when the number of data becomes large (infinite). See for example (Ljung, 1999) for detailed analysis of asymptotic distributions for many model structures and estimation methods.

4.6.2 Normalization When Using Residuals

Equivalent to multiplying the test quantity with a normalization variable, which is a function of measured data, is to let the threshold be

a function of the measured data. This is usually called an *adaptive threshold*.

The basic idea of adaptive thresholds is that since disturbances and other uncontrolled effects vary with time, also the thresholds should vary with time instead of being fixed to a constant value. An example is shown in Figure 4.2. The solid line represents the residual/test

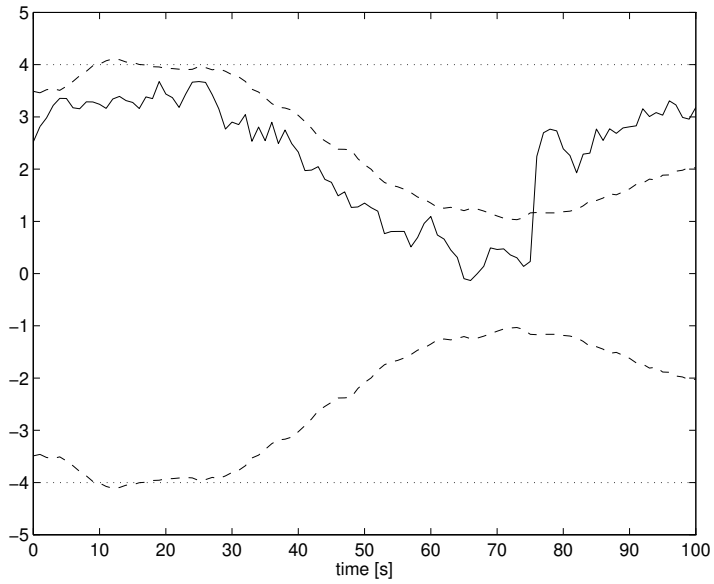


Figure 4.2: An example of the use of an adaptive threshold, with the test quantity (solid), the adaptive threshold (dashed), and as a comparison, the fixed threshold (dotted).

quantity, the dashed lines are the adaptive thresholds, and the dotted lines the fixed thresholds. There is a fault occurring at time $t = 75$ s, but because of disturbances, the test quantity is non-zero also before this time-point. To avoid false alarm, the fixed threshold has been set high. This means that the fault is missed if the fixed threshold is used. The adaptive threshold "adapts" to the disturbances and therefore follows the test quantity as long as there are no faults. When the fault occurs, the residual crosses the threshold and the fault is detected.

One technique for computing adaptive thresholds in connection with linear residual generation is presented in (Ding and Frank, 1991).

Consider a system that can be described as

$$y = \big(G(s) + \Delta G(s)\big)u + G_d(s)d + G_f(s)f + v$$

where $\Delta G(s)$ is a model error, u is the input, d is the disturbance, f is the fault, and v is measurement noise. Consider then a residual described by

$$\begin{aligned} r =& H_y(s)y + H_u(s)u = \\ &= H_y(s)\big(G(s)u + \Delta G(s)u + G_d(s)d + G_f(s)f + v\big) + H_u(s)u \end{aligned}$$

If measurement noise v is neglected and it is assumed that the input u and the disturbance d are perfectly decoupled in the model, then in the fault free case, the residual becomes

$$r = H_y(s)\Delta G(s)u$$

It is seen that the size of the residual in the fault free case depends on the absolute size of the model error $\Delta G(s)$ and the input $u(t)$. If $\delta > \|\Delta G(s)\|$ denotes a known bound of $\Delta G(s)$, the adaptive threshold can be selected as

$$J_{adp}(t) = \delta \|H_y(p)u\| \tag{4.32}$$

This approach relies on that a bound on the model uncertainty can be determined with confidence. If this is the case, it is guaranteed that no false alarm, caused by model uncertainties, will be generated.

Another approach is proposed in (Höfling and Isermann, 1996). This approach is more ad-hoc because the computation of the adaptive threshold is determined by tuning some design parameters. A generalized description of how the threshold is computed is the non-linear expression

$$J_{adp}(t) = kH_{LP}(p)\big(|H_d(p)u(t)| + c\big) \tag{4.33}$$

where $H_{LP}(s)$ and $H_d(s)$ are linear filters, and k and c constants. The filter $H_d(s)$ serves as a weighting in the frequency domain of model uncertainties. For frequency ranges where the model uncertainty is high, the filter gain should be high and vice versa. For example if the model is good for low frequencies but uncertain for higher frequencies, the filter $H_d(s)$ should be a high-pass filter. The value of the constant c is determined by the amount of other kinds of disturbances, such as measurement noise, and makes the threshold become greater than zero

even though the input is zero. Finally $H_{LP}(s)$ is a low-pass filter for smoothing of the threshold.

By using adaptive thresholds according to the principles described above, it is possible to get a nearly fixed significant level, independent of changes in the input signal. In this sense, the adaptive threshold is similar to the normalization described in the previous section for the test quantity based on the parameter estimate. Robustness is achieved in the sense that a certain significant level can be guaranteed independently of the input. Note however that if overall performance gains are desirable, these robustness techniques are never a substitute for using better models.

Both kinds of adaptive thresholds, i.e. (4.32) and (4.33), can be written on the more general form

$$J_{adp} = c_1 W(u, y) + c_2 \tag{4.34}$$

where $W(u, y)$ is some measure of the model uncertainty present for the moment.

To use an adaptive threshold is equivalent to *normalize* the test quantity. Consider the use of a test quantity $T(z)$ in combination with the threshold (4.34):

$$T(z) \geq J_{adp} \qquad \text{(reject } H_0\text{)}$$

By using normalization, this relation can instead be written as

$$T'(z) = \frac{T(z)}{c_1 W(u, y) + c_2} \geq 1 \qquad \text{(reject } H_0\text{)}$$

where $T'(z)$ is the normalized test quantity. The new threshold becomes $J = 1$.

4.6.3 Normalization When Using the Prediction Error

When the test quantity is constructed via the prediction error, it might be a good solution to estimate the overall model error $W(u, y)$ as

$$W(u, y) = \min_{\theta \in \Theta} V(\theta, z) = \min_{\theta \in \Theta} \sum_{t=1}^{N} (y(t) - \hat{y}(t|\theta))^2 \tag{4.35}$$

Note that the minimization is over *all* possible θ. The expression 4.35 might seem to be difficult to calculate but if the same $V(\theta, z)$ is used for

all hypothesis tests, as was described in Section 4.2, then the calculation of (4.35) becomes easy.

Now assume that $c_2 = 0$. Then an adaptive threshold becomes

$$J_{adp} = \min_{\theta \in \Theta} V(\theta, z)\ c_1 \tag{4.36}$$

With this adaptive threshold, the normalized version of a test quantity based on the expression (4.5) becomes

$$T'(z) = \frac{\min_{\theta \in \Theta^0} V(\theta, z)}{\min_{\theta \in \Theta} V(\theta, z)} > c_1 \qquad \text{(reject } H_0\text{)}$$

We will see that this expression has strong similarities with the likelihood ratio described in the next section.

One problem of using the expression (4.35) as an estimate of the model error is that different faults may be modeled with different degrees of freedom. For example, assume that behavioral mode $F1$ is present, and that there is a large model error present. Even though a model $\mathcal{M}_{F2}$ is not the model that corresponds to the present behavioral mode, it can have many free parameters so that a good agreement with data can be obtained. On the other hand, the model $\mathcal{M}_{F1}$ may have no free parameter and can therefore not be fitted well to the data, because of the model error. In this case, $W(y, u)$ would become small because the model $\mathcal{M}_{F2}$ can be fitted well to the data. The result is probably that the behavioral mode $F1$ is rejected in spite of that it is the correct diagnosis. One solution to this problem might be to use different data sets for the estimation of the parameter θ and the evaluation of expression (4.3).

4.6.4 Normalization When Using the Likelihood Function

Now consider a test quantity based on the likelihood function together with an adaptive threshold similar to the one defined by (4.35) and (4.36):

$$J_{adp} = \max_{\theta \in \Theta} L(\theta|z)\ c_1$$

Thus H_0 is rejected if

$$T(z) = \max_{\theta \in \Theta^0} L(\theta|z) < \max_{\theta \in \Theta} L(\theta|z)\ c_1$$

By using normalization, we get a new test quantity $T'(z)$ and H_0 is now rejected if

$$T'(z) = \frac{\max_{\theta\in\Theta^0} L(\theta|z)}{\max_{\theta\in\Theta} L(\theta|z)} < c_1 \tag{4.37}$$

The test quantity $T'(z)$ is called the *likelihood ratio* test quantity (or statistic). To emphasize that maximization is involved, the term *maximum likelihood ratio* or *generalized likelihood ratio* is also used in the literature.

A number of different variations of the maximum likelihood ratio exist. One variation is to switch the numerator and the denominator. Another is to make the maximization in the denominator of (4.37) over $\Theta^1 = \Theta^{0^C}$ instead of Θ. It can be shown that in this case, it is equivalent to make the maximization over Θ (Lehmann, 1986). Further, the maximization is often replaced by supremum. Two examples of variations are

$$T(z) = \frac{\sup_{\theta\in\Theta} L(\theta|z)}{\sup_{\theta\in\Theta^0} L(\theta|z)} \tag{4.38}$$

$$T(z) = \frac{\max_{\theta\in\Theta^1} L(\theta|z)}{\max_{\theta\in\Theta^0} L(\theta|z)} \tag{4.39}$$

The likelihood ratio test quantity is widely used in statistics. The reason is partly that it is the optimal test quantity in the case where both the null hypothesis and the alternative hypothesis are simple, i.e. Θ^0 and Θ^1 consists each of only one element (Neyman-Pearson lemma). Optimality proofs also exists for many other cases where H^0 is simple (P.H.Garthwaite, 1995). For many cases where a theoretical justification is missing, the likelihood ratio has still been shown to be very good in practice (Lehmann, 1986). However, there are also cases in which the likelihood ratio is not good (Lehmann, 1986). See also Section 4.7.1 for further optimality discussion of the likelihood ratio.

Commonly the *maximum log-likelihood ratio* is used. This together with a change detection application is illustrated in the following example:

Example 4.13 Consider a signal $z(t)$ which can be modeled as

$$z(t) = v(t) + \theta(t)$$

where the noise $v(t)$ is independent and in each sample $N(0, \sigma^2)$-distributed. Before the change-time t_{ch}, $\theta(t) = 0$ and after the change time, $\theta(t) = \mu$ where μ is unknown.

The following two hypotheses are considered:

$$\begin{aligned} H_0 &: \quad \text{no change in mean of } z(t) \\ H_1 &: \quad \text{an abrupt change in mean of } z(t) \text{ occurs} \end{aligned}$$

By using the assumption of independent data, the likelihood ratio test quantity in the form (4.38) becomes

$$T(z) = \frac{\sup_{\theta\in\Theta} L(\theta|z)}{\sup_{\theta\in\Theta^0} L(\theta|z)} = \frac{\sup_{[t_{ch},\mu]} L([t_{ch},\mu]|z)}{L([N,0]|z)} =$$
$$= \frac{\sup_{[t_{ch},\mu]} \prod_{i=0}^{t_{ch}-1} f(z(i)|0) \prod_{i=t_{ch}}^{N} f(z(i)|\mu)}{\prod_{i=0}^{N} f(z(i)|0)} = \sup_{[t_{ch},\mu]} \frac{\prod_{i=t_{ch}}^{N} f(z(i)|\mu)}{\prod_{i=t_{ch}}^{N} f(z(i)|0)}$$

Now by using the assumption of Gaussian data, and switching to the log-likelihood ratio, we get the following test quantity:

$$\begin{aligned} T'(z) &= \sup_{[t_{ch},\mu]} \ln \frac{\prod_{i=t_{ch}}^{N} f(z(i)|\mu)}{\prod_{i=t_{ch}}^{N} f(z(i)|0)} = \\ &= \sup_{[t_{ch},\mu]} \sum_{i=t_{ch}}^{N} \ln f(z(i)|\mu) - \sum_{i=t_{ch}}^{N} \ln f(z(i)|0) = \\ &= \sup_{[t_{ch},\mu]} -\frac{1}{2\sigma^2} \sum_{i=t_{ch}}^{N} (\mu - 2z(i))\mu = \\ &\overset{*}{=} \sup_{t_{ch}} \sup_{\mu} -\frac{1}{2\sigma^2} \sum_{i=t_{ch}}^{N} (\mu - 2z(i))\mu = \\ &= \frac{1}{2\sigma^2} \sup_{t_{ch}} \sup_{\mu} -(N - t_{ch} + 1)\mu^2 + 2\mu \sum_{i=t_{ch}}^{N} z(i) \end{aligned}$$

The equality marked with $\overset{*}{=}$ can be shown to hold in special cases, including this one, but is not generally valid.

Note the relation between this example and Example 4.2, where a similar problem was solved by using a residual.

4.7 Using CUSUM to Compute a Test Quantity

After generating, for example, a residual it may still be necessary to apply some post filtering of the signal before a reliable decision based on the signal can be made. For example, to attenuate noise or to obtain the desired trade-off between detection performance and detection time, a simple low-pass filter can be applied to the residual. For a generated residual $r(t)$, a new residual $r'(t)$ can be generated by

$$r'(t) = H_{LP}(p)r(t)$$

where $H_{LP}(p)$, for example a linear low-pass filter with a Butterworth structure, and an alarm is generated if $r'(t)$ exceeds its respective threshold.

This section will describe an often useful algorithm that can be used on any signal that is generated for detection. A simple, but often useful, alternative to e.g. a linear low-pass filter is the CUSUM (Cumulative SUM) algorithm. CUSUM is a simple non-linear detection algorithm first proposed by Page (1954). Now follows a brief, non-formal, description of the basic idea behind the algorithm. A formal derivation of the algorithm based on statistical fundamentals is given in Section 4.7.1. This section will show how the algorithm can be used together with residuals generated as in Section 4.3, but it is important to note that the approach is equivalently applicable to signals generated according the principles in Sections 4.2, 4.4, and 4.5. The only requirement is that the generated quantity is a *signal*, i.e. a time series that should be used for detection.

To make things simple, let the null hypothesis H_0 be the no-fault case, hypothesis H_1 the faulty case, and let $s(t)$ be a signal that has the following properties

$$E\{s(t)\} < 0, \quad H_0 \text{ is valid} \tag{4.40a}$$
$$E\{s(t)\} > 0, \quad H_1 \text{ is valid} \tag{4.40b}$$

This means that $s(t)$ changes sign, in the mean, when a fault occurs. Such a function is in (Page, 1954) referred to as a score function. Note that a basic residual $r(t)$ is not a score function since a residual ideally is 0 under H_0 and deviates from 0 under H_1.

Since the score function is subject to noise it is reasonable to average the score to be able to make a more reliable decision. Thus, given a score function $s(t)$ it is reasonable to cumulatively sum the score function according to

$$g(t+1) = g(t) + s(t) \tag{4.41}$$

The behavior of the cumulative sum $g(t)$ is then a negative drift, in the mean, as long as H_0 is valid and a positive drift when H_1 is valid. Detection of a fault is then transformed into a change in drift. To illustrate the procedure, consider the score function in Figure 4.3. In the plot it can be seen, for this particular realization, that the mean value before time $t = 500$ seem to be slightly less than zero and after $t = 500$ it seem to be slightly greater than zero, i.e. the score function seem to fulfill the requirements (4.40). The change is visible to the eye but it is not possible to set a fixed threshold to reliably detect the change. Figure 4.4 shows a plot of the cumulative sum, computed

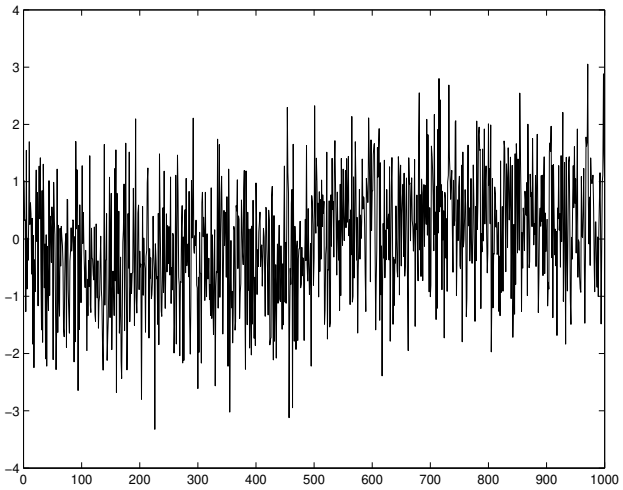


Figure 4.3: Score function $s(t)$ that has the property $E\{s(t)\} < 0$ before the change and $E\{s(t)\} > 0$ after the change.

according to (4.41) and there it is clearly visible that around $t = 500$, there is a change in the signal. Note that this cumulative sum is nothing more than a discrete time first order low-pass filter with a pole in $z = 1$. As discussed above, the change in the score function manifests as a

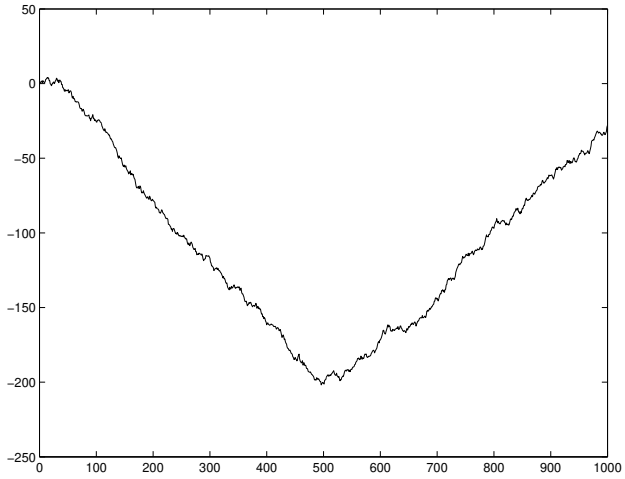


Figure 4.4: Cumulative sum of the score function in Figure 4.3.

change from a negative slope to a positive slope. An alarm can then be generated when the cumulative sum is significantly higher than the minimum value of the cumulative sum, i.e. compute the test quantity $T(t)$ according to

$$g(t) = \sum_{i=1}^{t} s(i) \tag{4.42a}$$

$$T(t) = g(t) - \min_{0 \le i < t} g(i) \tag{4.42b}$$

An alarm is generated when $T(t)$ is larger than some positive threshold J. The above algorithm is in (Page, 1954) called the CUSUM algorithm applied to the score function $s(t)$. Even though (4.42) is already in a simple form that is easy to implement in a computer, the test is commonly stated in the slightly different formulation

$$T'(t) = \max(0, T'(t-1) + s(t)), \quad T'(0) = 0 \tag{4.43}$$

where an alarm is generated when $T'(t)$ is larger than some positive threshold J. It is not the case that $T'(t) = T(t)$ but a test based on (4.42) is still equivalent to a test based on (4.43). This follows from the fact that that the threshold J is positive and the proposition below.

Proposition 4.1.

$$T'(t) = \max(0, T(t))$$

Proof. See Appendix 4.B. ■

Returning to the example and applying (4.43) to the score function in Figure 4.3 results in Figure 4.5 where the change around $t = 500$ is clearly visible and that it is easy to set a threshold to reliably detect the change.

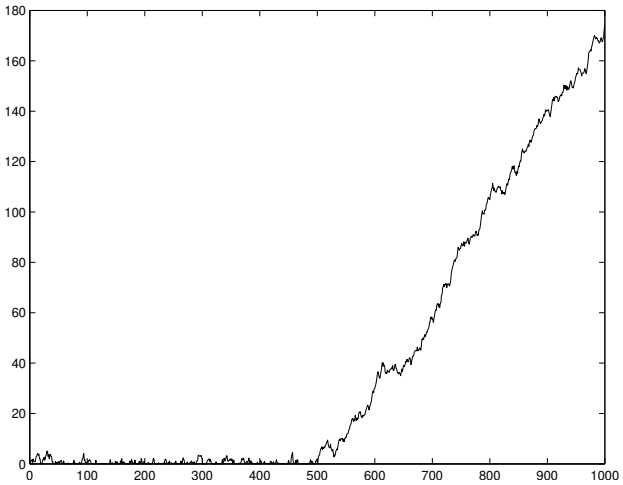


Figure 4.5: CUSUM algorithm applied to the score function in Figure 4.3.

Previously in this chapter, residuals, not score functions, have been generated and ideally a residual is zero in the fault free case and non-zero in case of a fault. Thus, the residual does not have the properties (4.40) of a score function. Typically, one then introduces a drift parameter ν such that $s(t) = |r(t)| - \nu$ has the desired properties (4.40). The CUSUM algorithm applied to a residual is then

$$T'(t) = \max(0, T'(t-1) + |r(t)| - \nu), \quad T'(0) = 0$$

and the drift parameter ν is a design parameter for the algorithm designer to choose. A rule of thumb is that ν is of the same order of magnitude as the size of the residual in the fault free case. From this it is also clear that ν can be used as an adaptive threshold that varies with the operating point, i.e. a large ν for operating points where the model uncertainty is large and a small ν for operating points where the model uncertainty is small.

4.7.1 Analytical Derivation of the CUSUM Algorithm

The derivation of the CUSUM algorithm above is based on intuition only and no formal motivation is given. This section will formally derive the CUSUM algorithm based on fundamental assumptions. First, consider a detection test with the simple hypotheses

$$H_0 : \theta(t) = \theta_0$$
$$H_1 : \theta(t) = \theta_1$$

and we know that observed data z, given the value of θ is distributed as $f(z|\theta)$. A classical result from statistical theory, the Neyman-Pearson lemma (Basseville and Nikiforov, 1993), states that an optimal test for the hypothesis is to alarm if

$$\log \frac{f(z|\theta_1)}{f(z|\theta_0)} > J$$

for some threshold J. This means that hypothesis H_0 is rejected when the probability of H_1 is significantly larger than the probability of H_0. The logarithm function is introduced for reasons that will become apparent in a moment, but note that since the logarithm function is monotone it does not change the performance of the detector, it only results in another choice of threshold J.

Now, return to our formal derivation of the CUSUM algorithm and consider the hypotheses

$$H_0 : \theta(k) = \theta_0,\ 0 \le k \le t$$
$$H_1 : \theta(k) = \begin{cases} \theta_0,\ 0 \le k < t_{ch} \\ \theta_1,\ k \ge t_{ch} \end{cases}$$

where the change time t_{ch} is considered unknown. Let data $z(k)$, $k = 1, \ldots, t$ be distributed as $f(z|\theta)$ and assume that data $z(i)$ and $z(j)$ for $i \neq j$ are independent. Then, similar to the simple hypothesis test discussed above, a test quantity is formed using the log-likelihood ratio. Utilizing the independence assumption we obtain

$$\log \frac{f(z(1), \ldots, z(t)|H_1)}{f(z(1), \ldots, z(t)|H_0)} = \log \frac{\prod_{k=1}^{t} f(z(k)|H_1)}{\prod_{k=1}^{t} f(z(k)|H_0)} =$$
$$\log \frac{\prod_{k=1}^{t_{ch}-1} f(z(k)|\theta_0) \prod_{k=t_{ch}}^{t} f(z(k)|\theta_1)}{\prod_{k=1}^{t} f(z(k)|\theta_0)} = \sum_{k=t_{ch}}^{t} \log \frac{f(z(k)|\theta_1)}{f(z(k)|\theta_0)}$$

If t_{ch} were known we could use this as a test quantity but now that t_{ch} is unknown we use the standard statistical approach, replace t_{ch} with the maximum likelihood estimation of t_{ch}. This results in

$$\max_{1\leq t_{ch}\leq t} \sum_{k=t_{ch}}^{t} s(k)$$

where $s(k)$ denotes the likelihood ratio at each time instant. This expression can be rewritten as

$$\max_{1\leq t_{ch}\leq t} \sum_{k=t_{ch}}^{t} s(t) = \sum_{k=1}^{t} s(t) - \min_{0\leq i<t} \sum_{k=1}^{i} s(k) = g(t) - \min_{0\leq i<t} g(i)$$

which is identical to (4.42). This means that if the score function is defined as the likelihood ratio and we have independent data, a log-likelihood ratio test together with maximum likelihood estimation of the unknown change time results in the CUSUM algorithm.

Likelihood ratio is a standard tool in detection theory. An in depth treatment of likelihood ratios, CUSUM algorithm, and also further development of more general algorithms can be found in for example (Basseville and Nikiforov, 1993).

Example 4.14 Assume that a residual $r(t)$ has been generated and that the residual $r(t)$ is subject to additive white Gaussian noise. Further, assume that a fault results in a change in mean of the residual from 0 to θ_1. The residual $r(t)$ is then $N(\theta, \sigma^2)$ distributed with $\theta = 0$ in the no fault case and $\theta = \theta_1$ in the faulty case. Under the assumption that θ_1 and the standard deviation σ can be considered known, the score function is then

$$s(t) = \log \frac{f(r(t)|\theta = \theta_1)}{f(r(t)|\theta = 0)} = \log \frac{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r(t)-\theta_1)^2}{2\sigma^2}}}{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{r^2(t)}{2\sigma^2}}} = \frac{\theta_1}{\sigma^2}(r(t) - \frac{\theta_1}{2})$$

and the test quantity according to (4.43) is

$$T'(t) = \max(0, T'(t-1) + \frac{\theta_1}{\sigma^2}(r(t) - \frac{\theta_1}{2})), \quad T'(0) = 0$$

This is a quite natural expression where $T'(t)$ increases when $r(t)$ is closer to θ_1 than 0.

Consider another case where, instead of change in mean, a fault causes a change in variance from $\sigma = \sigma_0$ to $\sigma = \sigma_1$. The score function then becomes

$$s(t) = \log \frac{f(r|\sigma = \sigma_1)}{f(r|\sigma = \sigma_0)} = \log \frac{\frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{r^2(t)}{2\sigma_1^2}}}{\frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{r^2(t)}{2\sigma_0^2}}} = \frac{1}{2} \log \frac{\sigma_0^2}{\sigma_1^2} + \frac{(1 - \frac{\sigma_0^2}{\sigma_1^2})}{2\sigma_0^2} r^2(t)$$

and the test quantity

$$T'(t) = \max(0, T'(t-1) + \frac{1}{2} \log \frac{\sigma_0^2}{\sigma_1^2} + \frac{(1 - \frac{\sigma_0^2}{\sigma_1^2})}{2\sigma_0^2} r^2(t)), \quad T'(0) = 0$$

The test quantity thus cumulatively sums $r^2(t)$ which is quite natural when one wants to monitor the variance of a signal.

Appendix

4.A Parameter Estimation

Estimation of constant parameters is involved in many of the techniques to calculate test quantities described above. This is especially common when the fault model is *deviation in constant parameters*. A thorough treatment of parameter estimation for many model structures, both linear and non-linear, can be found in e.g. (Ljung, 1999). Here a short description of how constant parameters can be estimated is given in a special case where the analysis is particularly simple, *linear regression*. The objective is to give some basic insight into the parameter estimation problem.

Thus, consider the case where the model can be stated in the form

$$y(t) = \varphi(t)\theta + v(t) \tag{4.44}$$

where $v(t)$ is independent noise with variance σ^2. Note that both $y(t)$ and θ can be vector valued. A model on the form (4.44) is called a linear regression.

When the model is in the form (4.44), θ can be estimated by using the *least squares* technique. The estimate can be written as

$$\hat{\theta} = (\Phi^T\Phi)^{-1}\Phi^T Y \tag{4.45}$$

where

$$\Phi = \begin{bmatrix} \varphi(1) \\ \vdots \\ \varphi(N) \end{bmatrix} \quad \text{and} \quad Y = \begin{bmatrix} y(1) \\ \vdots \\ y(N) \end{bmatrix}$$

It can be shown that the estimate (4.45) is optimal in several senses. Let θ_0 be the true value of the parameter θ, then the covariance matrix of the estimate can be computed as

$$E\{(\hat{\theta} - \theta_0)(\hat{\theta} - \theta_0)^T\} = \sigma^2(\Phi^T\Phi)^{-1}$$

Thus, the larger the regressor Φ, the smaller the variance of the estimate becomes which is intuitive.

Note that expression (4.45) is generally not a numerically good way of computing the estimate. However, numerical issues are beyond the

scope of this text and are left to interested readers to explore on their own.

Although not discussed here, it is also possible to use a recursive estimation approach, e.g. the RLS (Recursive Least Square) algorithm (Ljung, 1999). One common form of (4.44) is the case where we have a linear system that can be written as:

$$y(t) = a_1 y(t-1) + a_2 y(t-2) + \cdots + a_n y(t-n) + \\ + b_0 u(t) + \cdots + b_m u(t-m) + v(t)$$

This linear model is usually called an ARX (Auto Regressive eXogeneous) model. If the ARX model is written on the form (4.44), $\varphi(t)$ and θ becomes

$$\varphi(t) = [y(t-1)\ y(t-2)\ \ldots\ y(t-n)\ u(t)\ \ldots\ u(t-m)]$$
$$\theta = [a_1\ \ldots\ a_n\ b_0\ \ldots\ b_m]$$

4.B Proof of Proposition 4.1

Proposition 4.1.

$$T'(t) = \max(0, T(t))$$

Proof. The result will be proven by induction. Since $T(0) = T'(0) = 0$, the result holds for $t = 0$. Now, the induction assumption is

$$T'(j) = \max(0, T(j)), \quad j = 0, \ldots, t-1$$

The proof of the induction step will be separated into two cases.
Case 1: $T(t-1) \geq 0$
In this case, using the definition of $T(t)$, it is immediate that

$$T(t-1) = g(t-1) - \min_{0 \leq i < t-1} g(i) \geq 0 \Rightarrow \min_{0 \leq i < t} g(i) = \min_{0 \leq i < t-1} g(i) \quad (4.46)$$

Now, write

$$\begin{aligned} T'(t) = \max(0, T'(t-1) + s(t)) &\overset{(a)}{=} \max(0, T(t-1) + s(t)) = \\ = \max(0, g(t-1) - \min_{0 \leq i < t-1} g(i) + s(t)) &\overset{(b)}{=} \max(0, g(t) - \min_{0 \leq i < t} g(i)) = \\ &= \max(0, T(t)) \end{aligned}$$

where the equality marked (a) is due to the induction assumption and the equality marked (b) is due to (4.42a) and (4.46). This ends the proof for case 1.

Case 2: $T(t-1) < 0$
The second case is proved in a similar way as case 1. Using the definition of $T(t-1)$, we find that

$$T(t-1) = g(t-1) - \min_{0 \leq i < t-1} g(i) < 0 \Rightarrow \min_{0 \leq i < t} g(i) = g(t-1) \quad (4.47)$$

Expand $T'(t)$ as

$$\begin{aligned} T'(t) = \max(0, T'(t-1) + s(t)) &\overset{(c)}{=} \max(0, \max(0, T(t-1)) + s(t)) = \\ &\overset{(d)}{=} \max(0, s(t)) = \max(0, g(t) - g(t-1)) = \\ &\overset{(e)}{=} \max(0, g(t) - \min_{0 \leq i < t} g(i)) = \max(0, T(t)) \end{aligned}$$

where the equality marked (c) is due to the induction assumption, equality marked (d) due to the case $T(t-1) < 0$, and the equality marked (e) is due to (4.47). This ends the proof for case 2 and also the proof of the proposition. ■

Chapter 5
Evaluation of Test Quantity Performance

The previous chapter illustrated a number of principles one can follow when designing test quantities to be used in a diagnosis system. A natural question is then, which method is best in a particular situation? Not surprisingly there is no method that is always the best choice and the choice of method typically depends on the circumstances. This chapter deals with how to use statistics to evaluate performance of a test quantity and develop tools to compare performance of different test quantities.

5.1 The Power Function

There are many possible measures to evaluate the performance of a test quantity, here we will focus on the *power function*, a performance measure often used in statistics when evaluating performance in hypothesis testing. At the end of this section alternatives to the power function will be mentioned.

To introduce the power function, consider a hypothesis test with

two simple hypotheses

$$H_0 : \theta \in \Theta_0, \quad \text{fault free}$$
$$H_1 : \theta \in \Theta_1, \quad \text{fault}$$

Let a test quantity for the hypothesis test be denoted $T(z)$ and a corresponding threshold J. Naturally, important performance measures for a test are the probability to detect a fault and the probability for false alarm. Both these properties can be described by the *power function* $\beta(\theta)$ which is defined as the probability of rejecting the null hypothesis for a given value of θ. Formally, the function can be defined as

$$\beta(\theta) = P(\text{reject } H_0 \mid \theta) = P(T(z) > J \mid \theta)$$

Figure 5.1 shows a typical power function for a test where $\theta = 0$ corresponds to the null hypothesis. The function describes the probability

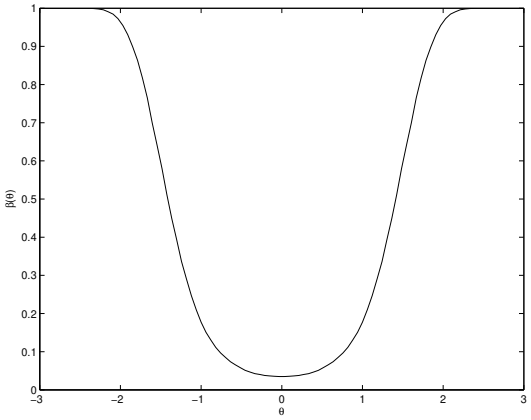


Figure 5.1: Example of a power function.

to detect a change in θ, thus for $\theta \notin \Theta_0$ $\beta(\theta)$ should be large and for $\theta \in \Theta_0$, $\beta(\theta)$ should be small. The value of $\beta(\theta)$ for $\theta \in \Theta_0$ gives the probability of false alarm. Thus, the function $\beta(\theta)$ captures both the probability to detect a change and also the probability of false alarm. This property makes the power function suitable as a performance indicator of a test quantity.

Generally, the probability to reject the null hypothesis when the null hypothesis is true is called the *significance level* of the test and is

formally defined as

$$\alpha = \sup_{\theta \in \Theta_0} \beta(\theta)$$

In statistical literature, the mistake to reject H_0 when H_0 is true is called a TYPE I error. Similarly, to not reject H_0 when the alternative hypothesis H_1 is true is called a TYPE II error. For the hypothesis test above, the TYPE I error is equal to the probability of false alarm and the TYPE II error is equal to the probability of missed detection.

In a general fault diagnosis system that consists of several hypothesis tests and more complex hypotheses, there is a connection between these errors and the probability of false alarm, missed detection, and missed isolation but not a direct one to one relation. We will not go into details here but it is clear that to achieve low probabilities of false alarm, missed detection, and missed isolation for the complete diagnosis system, we need to keep the probabilities of the TYPE I and II errors for each test quantity low.

The remainder of the text in this chapter will be devoted to the power function but it is worth mentioning that there are alternatives that are commonly used in the literature. Two examples are the mean detection delay (Basseville and Nikiforov, 1993) and the ROC-curve. Let t_{alarm} be the time the test quantity exceeds its threshold and t_{fault} the time the fault is introduced in the system. The mean detection delay is then formally

$$E\{t_{\text{alarm}} - t_{\text{fault}}\}$$

where t_{fault} is a deterministic fault time. The ROC-curve, or Receiver Operating Characteristics curve, is primarily from the detection theory in communication (Kay, 1998) and is a plot of probability of detection $P(D)$ as a function of the false alarm probability $P(FA)$. The curve is typically obtained by varying the threshold J. Examples of ROC-curves are shown in Figure 5.2 for three different fault sizes. Note that, although only the power function is considered in the text that now follows, most of the text is applicable and relevant also for these other performance measures.

5.2 Deriving the Power Function Analytically

We will here describe how the power function can be derived analytically. This is only possible in some special cases. One of these cases is studied

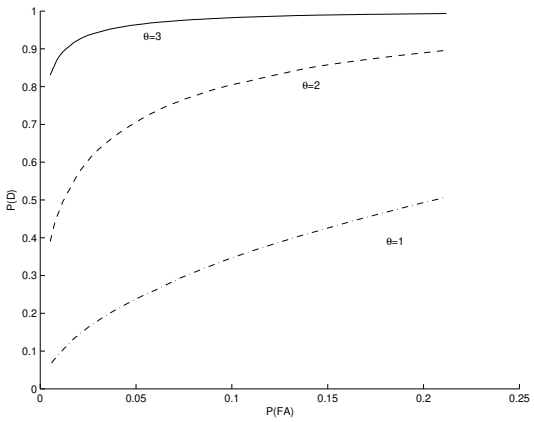


Figure 5.2: Examples of ROC-curves for three different values of θ.

here, namely the case when we have independent data which is also Gaussian distributed.

Consider a system which can be modeled as

$$y(t) = bu(t) + a + v(t) \tag{5.1}$$

where $v(t) \sim N(0, \sigma_v)$. The nominal (i.e. corresponding to the no fault case) value of b is $b_0 = 1$ and the nominal value of a is $a_0 = 0$. Now we will discuss how the power function, for two different test quantities, can be derived analytically. First for a test quantity based on a residual, and then for a test quantity based on a parameter estimate.

5.2.1 A Test Quantity Based on the Prediction Error

Assume that we have a test quantity $T_1(z)$ based on the prediction error, i.e.

$$T_1(z) = \sum_{t=1}^{N} (y(t) - \hat{y}(t))^2 = \sum_{t=1}^{N} (y(t) - u(t))^2 \tag{5.2}$$

Consider first the fault free case. Then

$$\frac{y-u}{\sigma_v} = \frac{b_0 u + v - u}{\sigma_v} = \frac{v}{\sigma_v} \quad \sim N(0,1) \tag{5.3}$$

This further implies that

$$\frac{T_1(z)}{\sigma_v^2} \sim \chi^2(N)$$

Now given a threshold J_1, we can calculate

$$\beta([a_0\ b_0]) = \beta([0\ 1]) = P(T_1(z) \geq J_1 \mid a = 0, b = 1)$$

This is equivalent to calculating

$$P(\frac{T_1(z)}{\sigma_v^2} \geq \frac{J_1}{\sigma_v^2} \mid a = 0, b = 1)$$

which can be done by using any standard table for the χ^2 distribution (or by using the built-in function in Matlab). That is, we have illustrated how to analytically derive the probability of a TYPE I error (false alarm) for a hypothesis test using the test quantity (5.2). When trying to do the same derivation for the faulty case, i.e. $b \neq 1$ or $a \neq 0$, it is seen that the quantity (5.3) does not have mean 0 anymore. This further implies that $T_1(z)/\sigma_v^2$ will not have a χ^2 distribution and the analytical derivation becomes more complicated. Instead of using analytical derivations, it is then more suitable to use simulations as described in Section 5.3 below.

5.2.2 A Test Quantity Based on an Estimate

Still with the model (5.1) in mind, assume that we want to derive the power function for a test quantity based on an estimate, namely the test quantity (4.31) from Example 4.12:

$$T_2'(z) = Np(\hat{b} - 1)^2 \qquad \hat{b} = \frac{1}{U^T U} U^T Y$$

Now given a threshold J_2, we want to calculate

$$\beta(b) = P(T_2'(z) \geq J_2 \mid b, a = a_0)$$

This is equivalent to computing

$$P\big(\sqrt{Np}(\hat{b} - 1) \geq \sqrt{J_2}\big) + P\big(\sqrt{Np}(\hat{b} - 1) \leq -\sqrt{J_2}\big) \tag{5.4}$$

From (4.30) it are then easy to use tables for the normal distribution to compute (5.4).

5.3 Estimating the Power Function Using Simulations

As was seen in the previous section, it is possible to derive the power function only in some (very) limited cases. Another method is to numerically compute, or at least estimate the power function. This can be done using so called Monte Carlo simulations.

The method can be described as follows:

1. Assume a distribution of noise in the observation data z. This noise does not have to be independent or Gaussian.

2. Fix the parameter θ for which we will calculate $\beta(\theta)$.

3. In a computer, generate a large amount of data series z_i, $i = 1, \dots N$, where N may be more than 1000, maybe 100000.

4. For each data series z_i, calculate the value t_i of the test quantity, i.e. $t_i = T(z_i)$.

5. Collect all the N values t_i in a histogram. This histogram is now an estimation of $f(t|\theta)$.

6. By using a fixed threshold J, $\beta(\theta)$ can now be estimated.

7. Go back to step 2 and fix a new θ.

Example 5.1 Consider the test quantities $T_1(z)$ in (5.2) and $T_2'(z)$ in (4.31). Assume that we want to compute power functions related to these test quantities. As was said above, the power function $\beta_1(b)$ for a test based on $T_1(z)$ can not be derived analytically. This means that we have to use simulations. On the other hand, the power function $\beta_2(b)$ for a test based on $T_2'(z)$ is possible to derive analytically. The thresholds for both tests are chosen such the significance levels equal each other, i.e. $\alpha_1 = \alpha_2 = 0.003$. Both the power functions $\beta_1(b)$ and $\beta_2(b)$ are plotted in Figure 5.3. From this plot we can conclude that the test based on $\beta_2(b)$ is better than the test based on $\beta_1(b)$. This is because $\beta_2(b) \geq \beta_1(b)$ for all b except $b = b_0 = 1$.

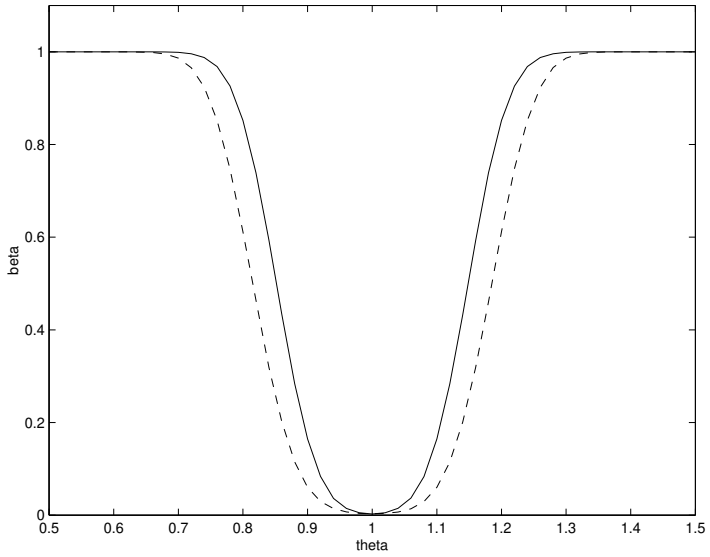


Figure 5.3: The power functions $\beta_1(b)$ (dotted) and $\beta_2(b)$ (solid) for two tests based on $T_1(z)$ and $T_2'(z)$.

5.4 Estimating the Power Function Using Measurement Data

We will here show how to estimate the power function $\beta(\theta)$ by using measurement data. The method is similar to simulation and can be described as follows:

1. To calculate $\beta(\theta)$ for a specific θ, we manipulate the process such that the parameter θ is fixed.

2. Collect a number of measurement series z_i, $i = 1, \ldots N$.

3. For each data series z_i, calculate the value t_i of the test quantity, i.e. $t_i = T(z_i)$.

4. Collect all the N values t_i in a histogram. This histogram is now an estimation of $f(t|\theta)$.

5. By using a fixed threshold J, $\beta(\theta)$ can now be estimated.

This procedure can be repeated for a number of different θ:s, and thereby the power function $\beta(\theta)$ can be obtained as a sampled function of θ.

Chapter 6
Linear Residual Generation

It is evident that the class of models that is considered greatly influences the difficulty of the residual generator design problem and that the more complicated the model, the more difficult it is to find general design and analysis methods. This chapter is about designing residual generators based on a simple class of models, linear process models where all faults and disturbances affecting the system are modeled as additive input signals.

Although linear models often have difficulties to accurately model physical processes, it is still one of the largest classes that can be treated in a simple enough and conclusive manner. Also, the analysis of linear system reveals and highlights important characteristics that are useful when performing designs based on more complicated models. These two facts are the motivation for the rather thorough treatment of linear systems that is included in this chapter. This is also the motivation for assuming that all inputs are considered deterministic, i.e no stochastic noise descriptions are included. Analysis of stochastic properties of the model and the residual generator are important topics, but not considered here since the main objective is to illustrate general principles and properties of the residual generation problem.

Linear systems were, probably because of their simplicity, one of the first areas to be investigated when researchers began to explore the residual generation problem. A pioneering work was done in (Chow and Willsky, 1984) where linear state-space descriptions was considered. Then, a great number of research papers and application studies followed and even today researchers continue to publish papers dealing with linear systems. A main focus of recent research is often design for uncertain models, optimal designs, and how to extend the class of linear models considered.

The method described in this chapter does not address uncertain systems or optimality conditions. It does however consider a class of models that is more general than commonly considered in the literature. The method relies on polynomial algebra and to not hide the rather straightforward principles of the design method behind polynomial algebra, the design method is first outlined for the static case in Section 6.4. The design procedure can then be well understood using only basic linear algebra. The dynamic case is then in Section 6.5 shown to be a straightforward generalization of the static case using polynomial algebra and rational vector spaces instead of vector spaces over real numbers. To be complete, all proofs are included in the text. However some of the more technical proofs and lemmas are placed in an appendix at the end of the chapter to not disturb the flow of the text. Also note that the appendix includes brief descriptions of null-spaces and the polynomial algebra that is needed to fully understand the presentation.

6.1 Basic Principles

Before describing general methodology for linear residual generation in the sections to follow, this section gives a brief outline of the basic principles behind the theory.

A linear residual generator can be said to be a linear filter that takes known signals z as input and generates a residual r that can be used to detect faults. A residual generator is thus a filter $R(p)$

$$r = R(p)z$$

For example, let the model be given by the transfer operators $G_u(p)$, $G_d(p)$ and $G_f(p)$, i.e. the model describing the influence of control

signals u, disturbances d and faults f on the measurement y is given by

$$y = G_u(p)u + G_d(p)d + G_f(p)f$$

Assume also that the model is stable, for such a model a residual generator can for example be written as

$$r = W(p)(y - G_u(p)u) = W(p)[I \ -G_u(p)] \begin{pmatrix} y \\ u \end{pmatrix} = R(p)z \qquad (6.1)$$

where $W(p)$ is a linear filter for the designer to choose. Note that (6.1) is the residual expressed in known signals only. Because the computation of the residual is based on such an expression it is called the *computational form*.

By substituting the model (6.6) for y in the residual generator we obtain

$$r = W(p)\left(G_d(p)d + G_f(p)f\right) \qquad (6.2)$$

A requirement on the residual is that it must be (ideally) 0 in the fault free case. This means that the residual must be zero for any disturbance d. This requirement imposes that it must hold that

$$W(p)G_d(p) = 0 \qquad (6.3)$$

Because of (6.3), it is seen in (6.2) that it is possible to express the residual in only the faults. Thus, the residual can be written as

$$r = W(p)G_f(p)f \qquad (6.4)$$

which means that the residual is able to detect the fault f if $W(p)G_f(p) \neq 0$. An expression like (6.4), i.e. an expression describing how the residual responds to faults, is called the *internal form*.

The goal of the sections that now follow is to describe a systematic way of designing the linear filter $R(p)$ but first a small example.

Example 6.1 Consider the linear model

$$y = \begin{bmatrix} \frac{1}{p+1} \\ \frac{1}{p+2} \end{bmatrix} u + \begin{bmatrix} \frac{1}{p+3} \\ \frac{1}{p+4} \end{bmatrix} d + \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$$

which has 2 sensors, one control input u, one disturbance d and two sensor faults f_1 and f_2. A residual generator can then, following the

procedure outlined above, be written as

$$r = W(p)\left(y - \begin{bmatrix} \frac{1}{p+1} \\ \frac{1}{p+2} \end{bmatrix} u\right)$$

The matrix $W(p)$ must fulfill (6.3) and one choice is

$$W(p) = \frac{1}{p+\alpha}\begin{bmatrix}(p+3) & -(p+4)\end{bmatrix}$$

with $\alpha > 0$ to ensure stability of the residual generator. This means that the complete residual generator can be written as

$$r = \begin{bmatrix} \frac{p+3}{p+\alpha} & -\frac{p+4}{p+\alpha} & -\frac{2}{(p+\alpha)(p+1)(p+2)} \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \\ u \end{pmatrix} = R(p)z$$

6.2 Model Description

Now that the general principle has been established, we are ready to become more formal and before we formally define the linear residual generation problem, we need to define the class of models that are considered. This is the topic of this section and the linear model description that will be used in this chapter is the following

$$H(p)x + L(p)z + F(p)f = 0 \tag{6.5}$$

where $x \in \mathbb{R}^{n_x}$ is a vector of *unknown* signals that should not influence the residual, $z \in \mathbb{R}^{n_z}$ is a vector of *known* signals, and $f \in \mathbb{R}^{n_f}$ is the vector of faults we wish to detect. Note that any signals we wish to decouple in the residual are collected in the vector x. Matrices $H(p)$, $L(p)$, and $F(p)$ are polynomial matrices of suitable dimensions in the differentiation operator p. The meaning of the operator differentiation operator p is illustrated by the small example

$$(p^2 + 2p + 3)x = \ddot{x} + 2\dot{x} + 3x$$

Models in form (6.5) is attractive from a modeling perspective since modern modeling techniques, such as object oriented modeling (Mattson et al., 1998) uses equation based modeling languages, e.g. Modelica.

The resulting model will then be in the general form (6.5) and not in state-space form. Typically, different components are modeled by specifying physical laws that connect internal variables to each other. The separate models are then connected together via algebraic constraints that result in a model that is not immediately in state-space form and may require considerable effort to translate to a state-space form.

This chapter will only consider time-continuous models. However, corresponding results for the time-discrete case is easily obtained by simply replacing the differentiation operator p for the time shift operator q and the Laplace variable s for the Z-transform variable z. To not make the presentation unnecessary technical we will impose a technical assumption in Section 6.2.2 on the model (6.5). This is done to avoid some uncommon special cases that require a rather technical treatment.

The model structure (6.5) looks a little bit different than what is found in most texts on automatic control where commonly state-space or transfer function models are used to model the process. Actually, there is a complete branch of automatic control that considers models in form (6.5). See for example the book by (Polderman and Willems, 1997) for an extensive review. The model structure (6.5) is attractive from a diagnosis perspective since for *passive* diagnosis systems, i.e. the diagnosis system does not influence the process, there is no need to distinguish between process inputs and outputs. It is only interesting to distinguish between known signals, unknown signals, and faults and this is only possible with formulations like (6.5). However, as will be shown below, the standard model descriptions fits directly into the more general form (6.5). For example, if the model is given by a transfer operator $y = G(p)u$ where

$$G(s) = \frac{b(s)}{a(s)}$$

the model does not fit exactly the form (6.5) since $G(p)$ is a rational expression in p, not polynomial. But, what the transfer operator really means is that y and u satisfies the differential equation

$$a(p)y - b(p)u = 0$$

which is a polynomial description which fits directly into the model description (6.5). In the text that now follows we will use both the

complex Laplace-variable s and the differentiation operator p. In most cases one may switch between s and p without further thought, but one should remember that $a(p)$ is an *operator* and $a(s)$ a polynomial in a complex variable. Thus, one can, strictly speaking, for example not talk about zeros to the polynomial $a(p)$. Also, note that

$$y = \frac{p}{p}u$$

does *not* imply that $y = u$, only that $\dot{y} = \dot{u}$.

As noted in the introduction of this chapter, first *static* models will be treated. Static models, i.e. models that does not contain any dynamics, does not contain any time-differentiated variables. Thus, for static models the matrices $H(p)$, $L(p)$, and $F(p)$ are constant matrices that does not contain the differentiation operator p.

To make the reader a little more familiar with the model structure, two examples conclude this section. A first that considers a small physical example and a second example where other common linear model descriptions and their relations to (6.5) are investigated.

Example 6.2 Consider two rotating inertias connected via a spring according to Figure 6.1.

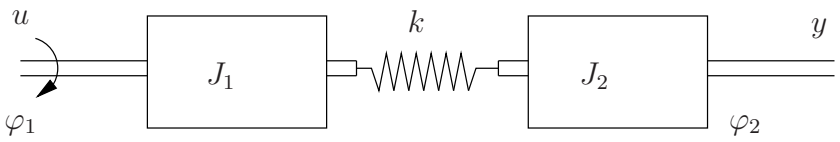


Figure 6.1: A small physical example. Two inertias connected with a spring.

Signals φ_1 and φ_2 are the crank angles of the two inertias, J_1 and J_2 are the moments of inertia, k the spring constant, and u a known driving torque. The equations describing the dynamics can then be written as

$$\begin{aligned} J_1\ddot{\varphi}_1 &= u - \tau_1 \\ \tau_1 &= k(\varphi_1 - \varphi_2) \\ J_2\ddot{\varphi}_2 &= \tau_1 \end{aligned}$$

where τ_1 is the torque developed by an ideal spring. Assume that we measure the angular velocity of the second inertia and that we model an additive sensor fault f. The measurement equation is then

$$y = \dot{\varphi}_2 + f$$

Now, the unknown variables, denoted with x in (6.5), are the variables φ_1, φ_2, and τ_1

$$x = \begin{pmatrix} \varphi_1 \\ \varphi_2 \\ \tau_1 \end{pmatrix}$$

The known variables, denoted z in (6.5), are the torque u and the measurement y

$$z = \begin{pmatrix} y \\ u \end{pmatrix}$$

Collecting the 4 model equations into the form (6.5) then gives the model matrices

$$H(p) = \begin{bmatrix} J_1 p^2 & 0 & 1 \\ k & -k & -1 \\ 0 & J_2 p^2 & -1 \\ 0 & -p & 0 \end{bmatrix}, \quad L(p) = \begin{bmatrix} 0 & -1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}$$

$$F(p) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

The reader is encouraged to verify that the matrices $H(p)$, $L(p)$ and $F(p)$ accurately reflects the original model.

Common model structures used in control theory is state-space descriptions and transfer functions. The example below will show that these common model structures directly fit into the more general model description (6.5).

Example 6.3 First, consider the common state-space model structure that is heavily used in control systems literature. Let the system

be described by the state-space equations[1]

$$\dot{w} = Aw + B_u u + B_d d + B_f f$$
$$y = Cw + D_u u + D_d d + D_f f$$

where w is the state vector, u known control inputs, d unknown disturbances, f the faults, and y the measurement.

Rewriting the model in matrix form with $x = (w,\ d)$ and $z = (y,\ u)$ gives

$$\begin{bmatrix} C & D_d \\ -(pI - A) & B_d \end{bmatrix} \begin{pmatrix} w \\ d \end{pmatrix} + \begin{bmatrix} -I & D_u \\ 0 & B_u \end{bmatrix} \begin{pmatrix} y \\ u \end{pmatrix} + \begin{bmatrix} D_f \\ B_f \end{bmatrix} f = H(p)x + L(p)z + F(p)f$$

Another common model description is transfer functions. Also these model descriptions fit directly into the general model form (6.5). To see this, let the model be described by the transfer operators

$$y = G_u(p)u + G_d(p)d + G_f(p)f \tag{6.6}$$

Also, let the numerators and denominators of the transfer functions be defined according to

$$D^{-1}(s)[N_u(s)\ N_d(s)\ N_f(s)] = [G_u(s)\ G_d(s)\ G_f(s)] \tag{6.7}$$

This means that the $N_u(s)$, $N_d(s)$, and $N_f(s)$ matrices describe the numerators and $D(s)$ the (common) denominator of the transfer matrices. Now, with $x = d$ and $z = (y,\ u)$ we obtain

$$N_d(p)x + [-D(p)\ N_u(p)]z + N_f(p)f = 0$$

which is in the form (6.5). In general, a decomposition of a multi-input multi-output transfer matrix into numerators and denominators as in (6.7) is called an MFD (Matrix Fraction Description).

The conclusion of this example is that commonly used linear model descriptions directly fit into the general linear model structure (6.5) and that any results derived for (6.5) also apply for state-space or transfer function models in a single framework. Thus, we do not need to develop two separate design algorithms and it will prove that using (6.5) is no more difficult than to only consider state-space or transfer function models.

[1] The variable w is used as a state-variable instead of the customary x to avoid any unnecessary mix up with x in (6.5).

6.2.1 Decoupling in Linear Systems

As stated, the model is assumed to be in the form

$$H(p)x + L(p)z + F(p)f = 0$$

Typically, one wants to generate residuals that corresponds to a fixed decision structure as described in Chapter 3. This section describes how to reformulate the model such that the designed residual generators fit into such a framework.

To illustrate, assume that the system is influenced by three faults, f_1, f_2, and f_3 and that we are to design two residual generators r_1 and r_2 corresponding to the decision structure.

	f_1	f_2	f_3
r_1	0	X	X
r_2	X	0	X

Design of Residual Generator r_1

Since residual r_1, according to the table above, should *not* be influenced by the fault f_1, the signal f_1 should be *decoupled* in the residual. A way to ensure this is to include the signal f_1 in the vector of unknown signals x in the model equations.

Let $F_1(p)$ be the first column in $F(p)$ in (6.5), i.e. the column corresponding to fault f_1. Then, when designing residual r_1, rewrite the model as

$$\begin{bmatrix} H(p) & F_1(p) \end{bmatrix} \begin{pmatrix} x \\ f_1 \end{pmatrix} + L(p)z + \begin{bmatrix} F_2(p) & F_3(p) \end{bmatrix} \begin{pmatrix} f_2 \\ f_3 \end{pmatrix} = 0$$

This means that when designing r_1, the x vector and the $H(p)$ matrix is redefined as

$$x := \begin{pmatrix} x \\ f_1 \end{pmatrix}, \quad H(p) := \begin{bmatrix} H(p) & F_1(p) \end{bmatrix}$$

After the rewrite, the model is in the form

$$H(p)x + L(p)z + \begin{bmatrix} F_2(p) & F_3(p) \end{bmatrix} \begin{pmatrix} f_2 \\ f_3 \end{pmatrix} = 0$$

If a residual generator is designed that is not influenced by the unknown signals in x, it is guaranteed that the fault signal f_1 is decoupled in the residual r_1 and thereby that r_1 is consistent with the desired decision structure.

Design of Residual Generator r_2

When, in a second step, the residual generator r_2 is designed, it is instead of f_1 fault signal f_2 that should not influence the residual. The vector x and the matrix $H(p)$ is then, similar as in the case of r_1, redefined as

$$x := \begin{pmatrix} x \\ f_2 \end{pmatrix}, \quad H(p) := \begin{bmatrix} H(p) & F_2(p) \end{bmatrix}$$

and the model is in the form

$$H(p)x + L(p)z + \begin{bmatrix} F_1(p) & F_3(p) \end{bmatrix} \begin{pmatrix} f_1 \\ f_3 \end{pmatrix} = 0$$

Again, designing a residual that is not influenced by the unknown signals in x, guarantees that the residual r_2 is consistent with the desired decision structure.

Final Comment on Decoupling

The above discussion illustrates how the model matrices in the model (6.5) are redefined for each residual design. How the variables and matrices are redefined depends on the desired decision structure, i.e. which signals should be decoupled in each residual.

6.2.2 A Technical Assumption

As noted above, a technical assumption will be imposed on the model equations to avoid some uncommon special cases that would make the presentation much more technical. Generally, for well posed model of physical processes, the assumption will hold as is discussed briefly below.

The assumption is the following rank condition

$$\text{rank}\,[H(s)\ L(s)] = m, \text{ for all } s \in \mathbb{C} \tag{6.8}$$

where m is the number of equations in the model, i.e. rows in (6.5). That $[H(s)L(s)]$ has full normal rank, i.e. full rank for some s, is a reasonable assumption since it means that there are no linear dependencies in the fault-free model equations. In non-mathematical terms one can say that this means that no two equations say the same thing. However, (6.8) not

only states that $[H(s)L(s)]$ has full normal-rank, it states that the model equations have full row-rank *for all* s. This is a technical assumption and means that the model is *controllable*. This is a generalization (Polderman and Willems, 1997) of the notion of controllability and for a state-space model the condition directly corresponds to the basic controllability condition that is used in any basic course on automatic control. A complete investigation of the residual generator problem for the case where the assumption above is not fulfilled can be found in (Nyberg and Frisk, 2006).

6.3 Linear Residual Generators

In Section 6.1, a residual generator was loosely described as a filter that in the fault free case is zero. For the residual to be any good for fault detection, we also require that the residual is non-zero in case of a fault. To formalize these notions, we start by describing the fault free behavior with the set $\mathcal{O}$

$$\mathcal{O} = \{z | \exists x; H(p)x + L(p)z = 0\} \tag{6.9}$$

The meaning of this set is that $\mathcal{O}$ is the set of *all* known signals z that is consistent with the fault-free model, i.e. could have originated from a fault-free process. The fault detection problem is then to decide if the measured signals $z \in \mathcal{O}$ or not. Note that z coming from a faulty process might very well also be included in $\mathcal{O}$. This topic will be studied further in Section 6.7 where fault detectability is discussed. Now, with the definition of $\mathcal{O}$, the fault free behavior of the system, we can formally define a residual generator as

Definition 6.1 (Linear residual generator)**.** A proper linear filter $R(p)$ is a residual generator and $r = R(p)z$ is a residual if

$$z \in \mathcal{O} \Rightarrow \lim_{t\to\infty} r = 0$$

The definition is quite natural, the residual should go to zero if the observed data is consistent with a fault free model. The reason why the residual r only is required to go to zero and not be identical to zero will be explained in Section 6.5 where residual generation for dynamic models are discussed. The properness requirement on the residual

generator $R(p)$ assures that the residual generator can be written in state-space form.

Now, the aim of the remainder of this chapter is to first find a design procedure that gives all residual generators $R(p)$ for a given model and also to analyze the fault detectability properties of both the residual generator and the system.

6.4 Residual Generators for Static Models

Now that the class of models and residual generators has been defined, it is time to develop a design algorithm. However, to not hide the simple and rather basic principles of the design algorithm behind polynomial algebra we will first only consider *static* linear systems. A design procedure is developed for this simpler class of models, and it will in Section 6.5 be shown how residual generators can be designed also for dynamic models using an identical design procedure, only utilizing polynomial matrix algebra.

A general static linear model is in the form

$$Hx + Lz + Ff = 0 \tag{6.10}$$

Compare this formulation with its dynamic counterpart (6.5), the only difference is that (6.10) does not include any differentiation operator p and matrices H, L and F are therefore *constant* matrices.

Now, to illustrate a design, let the rows of matrix N_H span the left null-space[2] of matrix H. This means that N_H has the maximum number of linearly independent rows that solves the equation

$$N_H H = 0$$

Now, multiply the model equation (6.10) from the left with N_H to obtain

$$N_H(Hx + Lz + Ff) = N_H Lz + N_H Ff = 0 \tag{6.11}$$

Here one sees that in the fault-free case, i.e. when $f = 0$, the expression $N_H Lz = 0$. By recalling Definition 6.1, one can then see that $r = Rz$ is a residual generator if we let

$$R = \gamma N_H L$$

[2]For those who need a recapitulation of null-spaces, see any basic book in linear algebra. A brief description is included in Appendix 6.B.

where γ is any row-vector of suitable dimensions. The row-vector γ thus parameterizes a set of residual generators. This simple outline showed a sufficient procedure to find a residual generator. The theorem below also states that it is in fact also a necessary procedure, i.e. all residual generators can be found by selecting different values of the parameter γ. The parameter γ should be selected to achieve best possible detection performance. How to select γ is further discussed in Section 6.8.

Theorem 6.1 (Static residual generators)**.** The constant matrix R is a residual generator for model (6.10) if and only if R can be written

$$R = \gamma N_H L$$

Proof. The if-part is already proven above and we only need to prove that if R is a residual generator, there exist a γ such that $R = \gamma N_H L$. Now, since R, according to assumption, is a residual generator we know that $z \in \mathcal{O} \Rightarrow Rz = 0$, which together with Lemma 6.7 from Section 6.D implies that

$$N_H L z = 0 \Rightarrow Rz = 0$$

This implication is equivalent to Im $R \subseteq$ Im $N_H L$, where Im R is the space spanned by the rows of R. Furthermore, this implies that there exist a row-vector γ such that

$$R = \gamma N_H L \tag{6.12}$$

which ends the proof. ■

Based on the theorem above we can now state a design procedure for residual generators for static linear models (6.10). The procedure can be summarized as

1. Form the model matrices H, L, and F.
2. Compute a basis N_H for the left null-space of matrix H.
3. A residual generator $r = Rz$ for the model can then be formed by $R = \gamma N_H L$ where γ is a free design parameter.

The parameter γ is the only parameter free for the designer to choose and the aim of this choice is to achieve desirable fault detectability properties in the residual. The fault influence on the residual is given by the expression

$$r_{int} = -\gamma N_H F f$$

which is obtained directly from (6.11). The design procedure is illustrated on a small idealized static example.

Example 6.4 Consider a simple static linear model with two known signals z_1 and z_2, two unknown signals x_1 and x_2 and one fault f to detect:

$$Hx + Lz + Ff = \begin{bmatrix} 8 & -11 \\ 6 & -22 \\ -8 & 10 \\ -2 & -5 \end{bmatrix} x + \begin{bmatrix} 4 & -9 \\ 3 & -3 \\ 1 & -11 \\ -10 & -10 \end{bmatrix} z + \begin{bmatrix} 15 \\ 1 \\ -12 \\ 0 \end{bmatrix} f = 0$$

Computing the left null-space to matrix H in MATLAB gives

$$N_H = \begin{bmatrix} 0.6815 & 0.0144 & 0.7220 & -0.1186 \\ 0.3466 & -0.4411 & -0.1859 & 0.8067 \end{bmatrix}$$

Here, choose the parameter $\gamma = [-1 \ -1]$ (an ad-hoc choice). Theorem 6.1 then gives that

$$r = \gamma N_H L z = 3.5119 z_1 + 20.7501 z_2$$

is the computational form of a residual generator. The internal form, i.e. the expression that shows the fault influence on the residual, is given by

$$r_{int} = -\gamma N_H F f = 8.5628 f$$

which also shows that the fault is detectable in the residual since in case of a fault ($f \neq 0$), also the residual will be non-zero.

6.5 Residual Generators for Dynamic Models

The goal of this section is to extend the approach outlined in the previous section for static models to cover also dynamic models. This will be done in a, more or less, identical way as for static models but using vector spaces over rational functions rather than vector spaces

over real numbers. For the purpose of this text, linear algebra over polynomials is in many ways no more difficult than regular linear algebra, only a few new notions need to be introduced. For those not familiar with such algebra, a brief summary of notions used in the text are described in Section 6.C.

The presentation will now closely follow the presentation for the static case. Thus, consider again the dynamic model (6.5), where the matrices $H(p)$, $L(p)$, and $F(p)$ again includes the differentiation operator p. Now, similar to the static procedure, let the rows of the polynomial matrix $N_H(s)$ span the left null-space of matrix $H(s)$. Now, multiply the model equation (6.5) from the left with $N_H(p)$ to obtain

$$N_H(p)(H(p)x + L(p)z + F(p)f) = N_H(p)L(p)z + N_H(p)F(p)f = 0$$

Here one sees that in the fault-free case, i.e. when $f = 0$, the expression $N_H(p)L(p)z = 0$. If we for the moment assume that not only z is known but also $\dot{z}$, $\ddot{z}$, and so on. Then we could compute a residual as $r = R(p)z$ where

$$R(p) = \gamma(p)N_H(p)L(p) \tag{6.13}$$

which should be compared to the, almost identical, expression (6.12) that was derived for the static case.

Example 6.5 Lets continue with the model from Example 6.2. This model is simple enough to compute a basis for the left null-space of matrix $H(s)$ by hand. For a general and more complicated model, efficient numerical computational tools, e.g. MATLAB, can compute this basis fast and in a reliable manner. In this case, the basis $N_H(s)$ is given by the expression

$$N_H(s) = \begin{bmatrix} k & -J_1 s^2 & k + J_1 s^2 & k(J_1 + J_2)s + J_1 J_2 s^3 \end{bmatrix}$$

The reader is encouraged to verify that this really is a basis for the left null-space, both that $N_H(s)H(s) = 0$ and that the dimension of the null-space really is one. Multiplying (6.5) from the left with $N_H(p)$

gives, for this example, in the fault free case

$$0 = \begin{bmatrix} k & -J_1p^2 & k + J_1p^2 & k(J_1 + J_2)p + J_1J_2p^3 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{pmatrix} y \\ u \end{pmatrix} =$$

$$\begin{bmatrix} k(J_1 + J_2)p + J_1J_2p^3 & -k \end{bmatrix} \begin{pmatrix} y \\ u \end{pmatrix} = J_1J_2y^{(3)} + k(J_1 + J_2)\dot{y} - ku$$

Again with the assumptions that time derivatives of z, i.e. u and y, are known we could set $\gamma(p) = 1$ and compute a residual using the expression

$$r = J_1J_2y^{(3)} + k(J_1 + J_2)\dot{y} - ku \tag{6.14}$$

That this expression is reasonable can, in this simple case with no disturbances d, be easily verified by computing the transfer function from u to y

$$G(s) = \frac{k}{J_1J_2s^3 + k(J_1 + J_2)s}$$

From $y = G(p)u$ the residual (6.14) can be directly identified.

Similar as in the static case, the fault influence on this residual is given by the expression

$$r_{int} = -\gamma(p)N_H(p)F(p)f = (k(J_1 + J_2)p + J_1J_2p^2)f = \\ = k(J_1 + J_2)\dot{f} + J_1J_2\ddot{f} \tag{6.15}$$

Unfortunately, assuming that the time derivative of z is known is a rather unrealistic assumption. It is often difficult to estimate derivatives, especially in a noisy environment. For example, the residual generator (6.14) in Example 6.5 requires that the third order derivative of the measurement signal y is known or can be reliably estimated. It is not hard to realize the difficulty in estimating the third order derivative in a noisy environment. Thus, equation (6.13) is generally not useful as a residual generator. Fortunately, we can still utilize this expression to derive a way to compute a residual following the same line of reasoning as in Section 4.3.1. Instead of computing the residual according to (6.13), compute it according to the differential equation

$$d(p)r = \gamma(p)N_H(p)L(p)z \tag{6.16}$$

where $d(p) = 1 + d_1 p + d_2 p^2 + \cdots + d_q p^q$ is a *stable* polynomial in p of order q. It is important that $d(s)$ is stable, i.e. all zeros in the open left half plane. This can be seen by observing (6.16) in the fault free case. In the fault free case we know that the right hand side equals 0, i.e., the residual r obeys the differential equation

$$d(p)r = 0$$

If then $d(s)$ has only strictly stable roots, r will go to 0 asymptotically. Now, using the transfer operator notation $R(p)$, the residual generator (6.16) can be written as $r = R(p)z$ with

$$R(p) = \frac{1}{d(p)}\gamma(p)N_H(p)L(p) \tag{6.17}$$

The row-vector $\gamma(p)N_H(p)L(p)$ is said to have *row-degree* k if k is the highest order polynomial of all elements in the vector. Now, if the order q of the polynomial $d(p)$ is greater or equal to the row-degree of $\gamma(p)N_H(p)L(p)$, the filter $R(p)$ can be written in state-space form and r can be computed without any need to estimate time derivatives of z. One way to realize the residual generator in state-space form is to use the observer canonical form which is described in Appendix 6.A. For general descriptions on realization theory, see any basic book on automatic control, for example (Glad and Ljung, 1997) (in Swedish) or (Kailath, 1980).

Example 6.6 Consider again Example 6.5 where

$$\gamma(p)N_H(p)L(p) = \begin{bmatrix} J_1 J_2 p^3 + k(J_1 + J_2)p & -k \end{bmatrix}$$

This means that the polynomial $d(p)$ must *at least* be of order 3 for the filter $R(p)$ in (6.17) to be realizable on state-space form. Let us place all poles of the residual generator in $s = -\alpha$ with $\alpha > 0$, i.e.

$$d(s) = (s + \alpha)^4$$

The transfer function $R(s)$ of the residual generator is then

$$R(s) = \frac{1}{(s+\alpha)^4}\begin{bmatrix} J_1 J_2 s^3 + k(J_1 + J_2)s & -k \end{bmatrix}$$

which can easily be stated on state-space form, e.g. in observer canonical form. This means that the residual can be computed without estimating any derivatives.

The approach outlined above and in the examples shows a *sufficient* way to design residual generators. But, as in the static case, it is in fact also a necessary procedure and this is summarized in the theorem below.

Theorem 6.2 (Dynamic residual generators)**.** A filter with transfer operator $R(p)$ is a residual generator if and only if there exists a stable polynomial $d(s)$ and a row-vector $\gamma(s)$ such that

$$R(p) = \frac{1}{d(p)}\gamma(p)N_H(p)L(p) \tag{6.18}$$

where $\deg d(p) \geq \deg \gamma(p)N_H(p)L(p)$.

Proof. The same proof given for Theorem 6.1 holds also for this theorem with only minor adjustments. It is left as an exercise for the reader to complete this proof. ■

Based on the theorem above we can now state a design procedure that can be summarized as

1. Form the model matrices $H(p)$, $L(p)$, and $F(p)$.
2. Compute a basis $N_H(s)$ for the left null-space of matrix $H(s)$.
3. A residual generator $r = R(p)z$ for the model can be formed where
$$R(p) = \frac{1}{d(p)}\gamma(p)N_H(p)L(p)$$
The row-vector $\gamma(p)$ and the polynomial $d(p)$ are free design parameters with the constraints
   - the polynomial $d(s)$ must have all its zeros in the open left half-plane.
   - the degree of $d(s)$ must be greater or equal to the row-degree of $\gamma(s)N_H(s)L(s)$ to be able to realize the residual generator on state-space form.

In the design method above, there is no discussion on how to choose the available design freedom that is concentrated in the polynomial $d(s)$ and row-vector $\gamma(s)$. This topic will be readdressed in Section 6.8 after fault detectability is discussed in Section 6.7.

A natural question to ask is, how many linearly independent residual generators does there exist for a given model. By looking at the theorem formulation above, one can see the set of linear residual generators can be seen as a linear vector space over rational vectors. Thus, the dimension of this space, which equals the number of rows of matrix $N_H(s)$, gives how many linearly independent residual generators that exists. Let n_r denote the dimension of this space. The number of rows of $N_H(s)$ is given by

$$n_r = m - \operatorname{rank} H(s) \tag{6.19}$$

where m is the number of equations in the model (i.e. rows of $H(s)$). For models where the unknown variables are independent, i.e., $H(s)$ has full column rank, this means that

$$n_r = m - n_x$$

where n_x is the number of unknown variables. Thus, the quantity $m-n_x$ can be said to be the *degree of redundancy* available in the model. This means that the more redundancy available, the more available design freedom we have when designing our residual generators $R(s)$. This also means that we need more model equations than unknown variables to be able to find any residual generator.

Example 6.7 Consider a state-space model as in Example 6.3. Matrix $H(s)$ is then given by

$$H(s) = \begin{bmatrix} C & D_d \\ -(sI - A) & B_d \end{bmatrix}$$

The number of rows in $H(s)$ is then $n_y + n_w$ where n_y is the number of measurements and n_w the number of states. Given that the input d represents independent inputs, i.e. $[D_d^T B_d^T]^T$ has full column rank, we have $\operatorname{rank} H(s) = n_w + n_d$ where n_d is the number of disturbances. The degree of redundancy is then

$$n_r = \underbrace{n_y + n_w}_{m} - \underbrace{(n_w + n_d)}_{n_x} = n_y - n_d$$

Thus, the degree of redundancy for a state-space model is then quite naturally the number of measurements minus the number of disturbances we need to decouple. Thus, a necessary condition for the existence of any residual generator is that the process is equipped with more sensors than there are unknown disturbances to be decoupled.

6.6 Residual Generators in State-Space Form

To implement a linear residual generator in a electronic control system, typically the filter is written in state-space form. As described above, if the numerators has a degree that is less or equal to the degree of the denominator in

$$R(p) = \frac{1}{d(p)}\gamma(p)N_H(p)L(p) \tag{6.20}$$

the filter can be written in state-space form.

Example 6.8 Consider the first order model

$$\begin{aligned}\dot{x} &= -x + u_1\\ y &= x + u_2\end{aligned}$$

with one output y and two inputs u_1 and u_2. A corresponding consistency relation is then

$$\dot{y} + y - (\dot{u}_2 + u_2) - u_1 = 0 \tag{6.21}$$

Adding residual generator dynamics as in (6.16), with a pole in $-\alpha$, then gives a residual generator in transfer function form as

$$R(p) = \frac{1}{p+\alpha}\begin{pmatrix} p+1 & -1 & -(p+1)\end{pmatrix}$$

In this first order system, it is straightforward to obtain a state-space form. By observing that (6.16) becomes

$$\dot{r} + \alpha r = \dot{y} + y - (\dot{u}_2 + u_2) - u_1$$

and with the state variable $w = r - y + u_2$ we obtain

$$\begin{aligned}\dot{w} = \dot{r} - \dot{y} + \dot{u}_2 = -\alpha r + y - u_2 - u_1 = -\alpha(w + y - u_2) + y - u_2 - u_1 =\\ = -\alpha w + \begin{pmatrix}(1-\alpha) & -1 & -(\alpha+1)\end{pmatrix}\begin{pmatrix} y\\ u_1\\ u_2\end{pmatrix}\end{aligned}$$

Thus, a state-space formulation of the residual generator is then

$$\dot{w} = -\alpha w + \begin{pmatrix}(1-\alpha) & -1 & -(\alpha+1)\end{pmatrix}\begin{pmatrix} y \\ u_1 \\ u_2 \end{pmatrix}$$

$$r = w + \begin{pmatrix}1 & 0 & -1\end{pmatrix}\begin{pmatrix} y \\ u_1 \\ u_2 \end{pmatrix}$$

In the above example, it was straightforward to find the state-space variable due to that it was only first order dynamics. It is however true that for any transfer function where the numerator degree is not higher than the denominator, one can always find a state-space form, see (Kailath, 1980) for a detailed description. There are many ways to find state-variables, and a simple alternative is to directly utilize the observer canonical form Appendix 6.A. The observer canonical form is not numerically suitable for large order residual generators, but is here used to illustrate the principle.

Example 6.9 Consider again the residual generator from Example 6.6 with the residual generator in transfer function form

$$R(s) = \frac{1}{(s+\alpha)^4}\begin{bmatrix} J_1 J_2 s^3 + k(J_1+J_2)s & -k \end{bmatrix}$$

To directly use the observer canonical form, expand the denominator as

$$(s+\alpha)^4 = s^4 + 4\alpha s^3 + 6\alpha^2 s^2 + 4\alpha^3 s + \alpha^4$$

and identify the polynomial coefficients. Then, the residual generator can be stated as

$$\dot{w} = \begin{pmatrix} -4\alpha & 1 & 0 & 0 \\ -6\alpha^2 & 0 & 1 & 0 \\ -4\alpha^3 & 0 & 0 & 1 \\ -\alpha^4 & 0 & 0 & 0 \end{pmatrix} w + \begin{pmatrix} J_1 J_2 & 0 \\ 0 & 0 \\ k(J_1+J_2) & 0 \\ 0 & -k \end{pmatrix}\begin{pmatrix} y \\ u \end{pmatrix}$$

$$r = \begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix} w$$

Note that this simple procedure is directly possible when the denominator degree is *strictly* higher than the numerator degrees. For the case where they are equal, perform a polynomial division first.

6.7 Fault Detectability

Definition 6.1 does not state anything on what happens with the residual in case of a fault, i.e. a residual generator is not required to be sensitive to a fault. This means that, according to the definition, $r \equiv 0$ is a residual generator although a rather useless one. The question then arises, is it at all possible to find a residual generator that is sensitive to the faults in f? This question is the topic of *detectability analysis*.

Before going further, there is a need to distinguish between *fault detectability* and *fault sensitivity* of a fault in a specific residual. A fault f_i is detectable in a model (6.5) if when $f_i \neq 0$, the measurements is distinguishable from measurements from a fault free process. Thus, fault detectability is a *system property* and is not related to a specific residual generator. More formally, fault detectability can be defined as

Definition 6.2 (Fault detectability)**.** Fault i is detectable in (6.5) if there exist signals f_i, x, and z, consistent with (6.5) with $f_j = 0$ for $j \neq i$, such that $z \notin \mathcal{O}$.

When talking about a specific residual generator and which faults that can be detected, we talk about *fault sensitivity* of a residual. Let $G_{rf_i}(s)$ be the transfer function from fault f_i to the residual r, then fault sensitivity can be defined as

Definition 6.3 (Fault sensitivity)**.** A residual from a residual generator $R(p)$ for (6.5) is sensitive to fault i if $G_{rf_i}(s) \neq 0$.

Let the residual generator $R(p)$ be given by (6.18), it is then straightforward to determine the transfer function $G_{rf_i}(s)$. First, multiplication of (6.5) from the left by $d^{-1}(p)\gamma(p)N_H(p)$ and trivial reordering of the terms gives that

$$d^{-1}(p)\gamma(p)N_H(p)(H(p)x + L(p)z) = -d^{-1}(p)\gamma(p)N_H(p)F(p)f$$

Since $N_H(p)H(p)$ is the null operator and that $R(p)$ is given by (6.18) we have obtained the transfer function from fault to residual $G_{rf}(s)$ as

$$G_{rf}(s) = -d^{-1}(s)\gamma(s)N_H(s)F(s) \tag{6.22}$$

The following small example illustrates the two above definitions

---

Example 6.10 Consider the model

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{bmatrix} \frac{1}{p+1} \\ \frac{1}{p+2} \end{bmatrix} u + \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$$

Both faults f_1 and f_2 are clearly detectable since they both directly affect the measurement signal y and there are no unknown signals. However, for the residual generator

$$r = y_1 - \frac{1}{p+1}u = \begin{bmatrix} 1 & 0 & -\frac{1}{p+1} \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \\ u \end{pmatrix} = R(p)z$$

the transfer functions from faults to the residual is given by

$$G_{rf_1} = 1, \quad G_{rf_2} = 0$$

which means that the residual is only sensitive to fault f_1 and not f_2.

6.7.1 Fault Detectability Conditions

Now that the basic definitions have been introduced, we can derive conditions to test if a fault is detectable or not. For a general model, it is not as easy as in Example 6.10 to determine the detectability properties of the model and residual generator. Before deriving a detectability condition, we state a rather expected result that supports Definition 6.2; a fault is detectable if and only if there exists a residual generator sensitive to that particular fault.

Lemma 6.3. A fault f_i is detectable in (6.5) if and only if there exists a residual generator that is sensitive to fault f_i.

Proof. See Section 6.D.1. ■

Before the main detectability condition is stated, a small example is included that illustrates the basic principles behind the detectability condition.

Example 6.11 Consider the state-space model

$$\begin{aligned} \dot{w} &= Aw + Bu + \begin{bmatrix} 1 \\ 2 \end{bmatrix} d + \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} f \\ y &= Cw \end{aligned}$$

where d is an unknown disturbance that should be decoupled in the residual. In this case, the fault f will not be detectable if any influence on the system from f equally well could be the cause of a disturbance

d, i.e. d and f have similar influence on the system. Thus, for the fault to be detectable the influence from f on the system must not lie in the same space as the influence from d. This means that f is detectable if

$$\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \notin \text{Im} \left\{ \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right\}$$

Thus, if $c_2 \neq 2c_1$ the fault f will be detectable, otherwise not. The set membership test above can also be written as the rank test

$$\text{rank} \begin{bmatrix} 1 & c_1 \\ 2 & c_2 \end{bmatrix} > \text{rank} \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Following the same reasoning as in the example above, we can now state the main fault detectability condition.

Theorem 6.4. Fault i is detectable in (6.5) if and only if

$$\text{rank}\,[H(s)\ F_i(s)] > \text{rank}\,H(s) \tag{6.23}$$

Proof. Multiply (6.5) by $N_H(p)$ and rearrange the terms to obtain

$$N_H(p)L(p)z = -N_H(p)F_i(p)f_i$$

Due to condition (6.23) it holds that $N_H(p)F_i(p) \neq 0$. Thus, there exists a row-vector $\gamma(p)$ such that $\gamma(p)N_H(p)F_i(p) \neq 0$ and a $d(p)$ of high enough order such that $R(p) = d^{-1}(p)\gamma(p)N_H(p)L(p)$ is a residual generator with transfer function $G_{rf_i}(s) \neq 0$. Then, according to Lemma 6.3 the fault f_i is detectable. The converse is proved in a similar way and is left for the interested reader to prove. ■

A procedure to test detectability can thus be summarized as

1. Form the model matrices $H(p)$, $L(p)$, and $F(p)$.
2. For each fault i, verify that

$$\text{rank}\,[H(s)\ F_i(s)] > \text{rank}\,H(s)$$

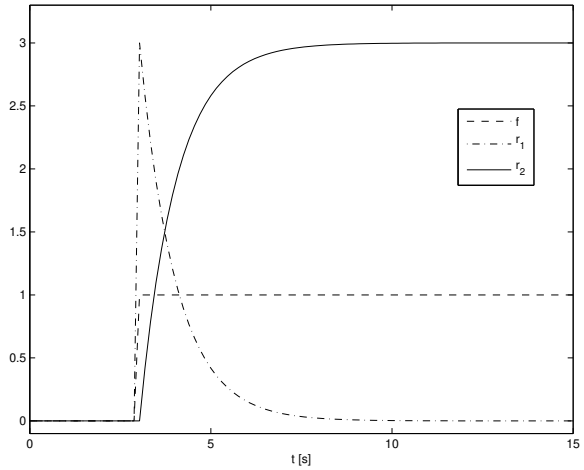


Figure 6.2: The dashed line is the fault signal and the solid (r_1) and dash-dotted (r_2) signals are two residuals.

6.7.2 Strong Fault Detectability

Detectability of a fault is however sometimes not enough in a common practical situation; constant faults. To illustrate this, assume two residual generators that, when excited by a constant fault, behave as in Figure 6.2. The residuals r_1 and r_2 have fundamentally different behaviors. Residual r_1 only reflect *changes* in the fault signal and r_2 has approximately the same shape as the fault signal. It is evident that it is more difficult to use r_1 than r_2 in a reliable diagnosis system to detect the fault even though it is clear that $G_{r_i f}(s) \neq 0$ for both $i = 1, 2$.

The difference between the two residuals is the stationary property, i.e. the value of $G_{rf}(0)$. It is clear that residual r_1 has $G_{r_1 f}(0) = 0$ while residual r_2 have $G_{r_2 f}(0) \neq 0$. This leads to the definition of *strong fault detectability*.

Definition 6.4 (Strong fault detectability)**.** Fault i is strongly detectable in (6.5) if there exist a *constant* signal f_i, and signals x, and z, consistent with (6.5) with $f_j = 0$ for $j \neq i$, such that $z \notin \mathcal{O}$.

Note that the word *constant* in the definition above means that $f(t)$ constant for all $t \in (-\infty, \infty)$. Note that the fault in Figure 6.2

is only constant *after* $t = 3$. Quite naturally, there is a corresponding result to Lemma 6.3 also for the strong detectability case.

Lemma 6.5. A fault f_i is strongly detectable in (6.5) if and only if there exists a residual generator that is sensitive to fault f_i and $G_{rf_i}(0) \neq 0$.

Proof. The proof is given by a slight modification of the proof of Lemma 6.3. ■

The main condition for testing *strong* fault detectability is then given by

Theorem 6.6. A fault f_i is strongly detectable in (6.5) if and only if $N_H(0)F_i(0) \neq 0$.

Proof. The only-if part is immediate since according to Lemma 6.5 there exists a residual generator (6.18) such that

$$G_{rf_i}(0) = d^{-1}(0)\gamma(0)N_H(0)F_i(0) \neq 0$$

which implies that $N_H(0)F_i(0) \neq 0$. The if-part is proven in the same way as the if-part of the proof of Theorem 6.4. ■

Faults that are detectable but not strongly detectable are here called *weakly detectable* faults.

Example 6.12 Consider again Example 6.2 with the two rotating inertias. In Example 6.5 $N_H(s)F(s)$ was computed

$$N_H(s)F(s) = -k(J_1 + J_2)s - J_1J_2s^2$$

from which it is clear that $N_H(0)F(0) = 0$ and thus the fault f is only weakly detectable. This can also be seen in (6.15) where only derivatives of the fault signal appears in the internal form of the residual generator.

A procedure to test for strong detectability can thus be summarized as

1. Form the model matrices $H(p)$, $L(p)$, and $F(p)$.
2. Compute a basis $N_H(s)$ for the left null-space of matrix $H(s)$.
3. For each fault i, verify that

$$N_H(0)F_i(0) \neq 0$$

6.8 Choice of parameters $\gamma(s)$ and $d(s)$

The design procedure in Section 6.5 includes two design choices that has not yet been thoroughly discussed; how to in step 3 of the residual generator design method, choose the polynomial $d(s)$ and the parameter $\gamma(s)$. No optimal procedure how to make these choices will be presented here, rather a short and basic guide on how a designer may think while making these choices. In short one can say that the choice of $\gamma(s)$ influences *fault sensitivity* properties of the residual and $d(s)$ ensures that the filter $R(p)$ is proper and can be written on *state-space form*. The polynomial $d(s)$ also gives the *dynamics* of the residual generator and can impose for example low-pass characteristics of the residual generator to filter out measurement noise and other uncertainties.

For the selection of $\gamma(s)$, consider the transfer function from f to r in (6.22)

$$G_{rf}(s) = -d^{-1}(s)\gamma(s)N_H(s)F(s)$$

Since $d(s) \neq 0$ it is clear that it is the term $\gamma(s)N_H(s)F(s)$ that determines if the transfer function from fault to residual is non-zero, i.e. if the residual is sensitive to the fault or not. And, as proven in Lemma 6.3, if a fault is detectable there exists a γ such that $\gamma(s)N_H(s)F(s) \neq 0$. It is generally rather straightforward to choose $\gamma(s)$ and there is often no reason to choose a dynamic $\gamma(s)$, i.e. for most cases $\gamma(s)$ will be a constant row-vector. The example below illustrates the choice of $\gamma(s)$. See also the proof of Lemma 6.3 for some further details on how to find a feasible $\gamma(s)$.

Example 6.13 Consider again Example 6.10 and that we want to detect fault f_1. Deriving a state-space realization of the model and

putting the equation in form (6.5) results in e.g.

$$\begin{bmatrix} p+1 & 0 \\ 0 & p+2 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} x + \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \\ u \end{pmatrix} + \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \end{bmatrix} f_1 = 0$$

Computation by hand gives that

$$N_H(p)F(p) = \begin{bmatrix} 1 & 0 & p+1 & 0 \\ 0 & 1 & 0 & p+2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \end{bmatrix} = \begin{bmatrix} p+1 \\ 0 \end{bmatrix}$$

from which it is obvious that $\gamma(s) = [1\ 0]$ is a feasible choice to ensure that the residual is sensitive to f_1.

Now that $\gamma(s)$ has been chosen, we need to choose the denominator $d(s)$. First of all, the degree of the stable polynomial $d(s)$ must be *at least* as high as the degree of the numerator $\gamma(p)N_H(p)L(p)$. This is because of the requirement that the residual generator should be realizable on state-space form. Generally, $d(s)$ is chosen such that the residual generator gets satisfactorily filtering properties, e.g. low-pass characteristics. Consider the residual generator

$$r = R(p)z = \frac{a(p)y + b(p)u}{d(p)}$$

and let the measurement signal y be subjected to measurement noise, i.e.

$$y = y_0 + \epsilon$$

where y_0 is the real physical entity that is measured and ϵ is measurement noise. In case we had no measurement noise ($\epsilon = 0$), the residual would be identically zero (after the initial decay due to unknown initial conditions) but with measurement noise the residual will not be identically zero. In the fault free case, the residual will respond to noise according to

$$r = \frac{a(p)}{d(p)}\epsilon$$

and here it is clear how $d(p)$ shapes the transfer function from the noise ϵ to the residual r. Thus, a suitable choice of low-pass poles, e.g.

in a Butterworth structure, will provide suitable choice of filtering of measurement noise. Note that, $d(p)$ also shapes the response of faults in the residual. Thus, the more low-pass effect that is imposed in the residual, the slower the response to a fault.

6.9 Consistency Relations

Consistency relations was introduced in Section 4.3.1 and is a relation among known signals that is valid in case of $f = 0$. This chapter has not yet touched upon this concept, but there exists a close relation between such consistency relations and the design procedure outlined in this chapter. This will be explored a bit further in this section. Linear consistency relations can efficiently be written using a row-vector $Q(p)$ as $Q(p)z = 0$. Thus, for $Q(p)z = 0$ to be a consistency relation it must hold that

$$z \in \mathcal{O} \Rightarrow Q(p)z = 0$$

Based on Lemma 6.7 it is clear that the design procedure from Section 6.5 also provides a set of consistency relations. In fact, each row in the expression $N_H(p)L(p)z = 0$ is a consistency relation. It can also be proven that the expression $N_H(p)L(p)z$ not only gives a set of consistency relations but all possible consistency relations for a model (6.5) can be written as a linear combination of the rows in $N_H(p)L(p)z = 0$. Thus, one can say that $N_H(p)L(p)$ spans the space of consistency relations.

Example 6.14 Consider again the model in Example 6.2. Computing a basis for the left null-space of matrix $H(s)$ gives that

$$N_H(p) = \begin{bmatrix} k & -J_1p^2 & k + J_1p^2 & J_1J_2p^3 + k(J_1 + J_2)p \end{bmatrix}$$

Since $z \in \mathcal{O} \Rightarrow 0 = N_H(p)L(p)z$ we obtain that $z \in \mathcal{O}$ implies that

$$\begin{aligned} 0 = J_1J_2y^{(3)} + k(J_1 + J_2)\dot{y} - ku = \\ = \begin{bmatrix} J_1J_2p^3 + k(J_1 + J_2)p & -k \end{bmatrix} \begin{pmatrix} y \\ u \end{pmatrix} = Q(p)z \end{aligned}$$

where $y^{(i)}$ is the i:th derivative of y.

6.10 Concluding Design Example

To conclude this section, a design is made based on a slightly larger model where computation by hand is unattractive. The model is taken from (Maciejowski, 1989) and represents a linearized model of vertical-plane dynamics of an aircraft. The inputs and outputs of the model are

Inputs		Outputs	
u_1:	spoiler angle [tenth of a degree]	y_1:	relative altitude [m]
u_2:	forward acceleration [ms^{-2}]	y_2:	forward speed [ms^{-1}]
u_3:	elevator angle [degrees]	y_3:	Pitch angle [degrees]

The model is provided by a 5:th order state-space model with state-space matrices:

$$A = \begin{bmatrix} 0 & 0 & 1.132 & 0 & -1 \\ 0 & -0.0538 & -0.1712 & 0 & 0.0705 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0.0485 & 0 & -0.8556 & -1.013 \\ 0 & -0.2909 & 0 & 1.0532 & -0.6859 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ -0.12 & 1 & 0 \\ 0 & 0 & 0 \\ 4.419 & 0 & -1.665 \\ 1.575 & 0 & -0.0732 \end{bmatrix}$$

$$C = [I_3 \; 0_{3\times 2}]$$

$$D = 0_{3\times 3}$$

Suppose the faults of interest are three sensor-faults (denoted f_1, f_2, and f_3), and three actuator-faults (denoted f_4, f_5, and f_6). Also, assume that the faults are modeled with additive fault models. The total model, including faults, then becomes:

$$\begin{bmatrix} C \\ -(pI_5 - A) \end{bmatrix} w + \begin{bmatrix} -I_3 & D \\ 0_{5\times 3} & B \end{bmatrix} \begin{pmatrix} y \\ u \end{pmatrix} + \begin{bmatrix} I_3 & 0_{3\times 3} \\ 0_{5\times 3} & B \end{bmatrix} \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \end{pmatrix} = 0$$

where w is the state-vector.

Design Problem

The design example is intended to illustrate the design procedure and also illustrate how available design freedom can be utilized, e.g. when selecting the residual structure but also when selecting dynamics of the residual generator.

The objective is to design a residual generator $R(p)$ that decouples the fault in the elevator angle actuator f_6. The result will be a residual that reacts to any of the faults $f_1, \ldots, f_5$, but not to f_6. In this case, the residual will correspond to a row in the influence structure of the form

	f_1	f_2	f_3	f_4	f_5	f_6
r	x	x	x	x	x	0

To place this design problem into the problem formulated in previous sections, define vectors f, x, and z as

$$f = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{pmatrix}, \qquad x = \begin{pmatrix} w \\ f_6 \end{pmatrix}, \qquad z = \begin{pmatrix} y \\ u \end{pmatrix}$$

Note that the fault f_6 that is to be decoupled in the residual is included in the vector of unknown signals x. The first step in the design procedure is to form matrices $H(p)$, $L(p)$ and $F(p)$ and they are here given by

$$H(p) = \begin{bmatrix} C & 0_{3\times 1} \\ -(pI_5 - A) & B_3 \end{bmatrix}, \quad L(p) = \begin{bmatrix} -I_3 & D \\ 0_{5\times 3} & B \end{bmatrix}$$
$$F(p) = \begin{bmatrix} I_3 & 0_{3\times 1} & 0_{3\times 1} \\ 0_{5\times 3} & B_1 & B_2 \end{bmatrix}$$

where B_i is the i:th column of matrix B.

The second step is to compute a basis $N_H(s)$ for the left null-space of $H(s)$. Computations in Matlab gives that $N_H(s)L(s)$ is

$$N_H(s)L(s) = \left[\begin{array}{cccccc} 0.0705s & s + 0.0538 & \ldots & & & \\ 22.7459s^2 + 14.5884s & -6.6653 & \ldots & & & \\ \ldots & 0.091394 & 0.12 & -1 & 0 \\ \ldots & s^2 - 0.93678s - 16.5141 & 31.4058 & 0 & 0 \end{array} \right]$$

As was described in Section 6.9, this gives that there exists exactly two linearly independent consistency relations that decouples d. This is consequent with expression (6.19) since we have 8 measurements and 6 signals/disturbances to be decoupled. The two consistency relations are in the time domain given by

$$\begin{aligned} &0.0705\dot{y}_1 + \dot{y}_2 + 0.0538y_2 + 0.091394y_3 + 0.12u_1 - u_2 = 0 \\ &22.7459\ddot{y}_1 + 14.5884\dot{y}_1 - 6.6653y_2 + \\ &\qquad + \ddot{y}_3 - 0.93678\dot{y}_3 - 16.5141y_3 + 31.4058u_1 = 0 \end{aligned}$$

The interpretation of these expressions is that they are valid for all y and u that satisfies the model equations in both the fault free case and when there is a fault on the third actuator. It is evident that f_6, i.e. the fault on actuator u_3, is decoupled in these expressions since u_3 is not included in the consistency relations. Thus, the relations hold *regardless* of whether the value of u_3 is correct or not. The row-degrees of the expression $N_H(s)L(s)$ is 1 and 2. From this it is clear that the filter of least degree, which decouples d, is a first order filter corresponding to row 1 in the basis.

Design choices

Following the discussion in Section 6.8, select $\gamma(s)$ to $\gamma(s) = [1\ 0]$. Thus, the numerator of the residual generator is given by

$$K(p) = \gamma(p)N_H(p)L(p) = [0.0705p \quad p{+}0.0538 \quad 0.091394 \quad 0.12 \quad -1 \quad 0]$$

To form a realizable residual generator, the polynomial $d(s)$ need to have degree ≥ 1 since the row-degree of the $K(p)$ is 1. A first order polynomial is chosen to get minimal order. Further, we choose to detect faults with energy in frequency ranges up to 1 rad/s, so therefore the polynomial $d(p)$ is chosen to $d(p) = 1 + p$. The realizable residual generator is then given by the following 1:st order filter:

$$R(p) = \frac{1}{1+p}\begin{bmatrix} 0.0705p & p + 0.0538 & 0.091394 & 0.12 & -1 & 0 \end{bmatrix} \quad (6.24)$$

This can be realized on state-space form as

$$\begin{aligned} \dot{w} &= -w + \begin{bmatrix} -0.0705 & -0.9462 & 0.091394 & 0.12 & -1 & 0 \end{bmatrix} z \\ r &= w + \begin{bmatrix} 0.0705 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} z \end{aligned}$$

Result

Figure 6.3 shows the singular value (maximum gain in any direction) for the transfer function from x to r

$$G_{rx}(s) = -d^{-1}(p)\gamma(p)N_H(p)H(p)$$

This plot should theoretically be exactly 0, but because of finite word length in MATLAB it doesn't become exactly 0. The plot shows that the control signals and the decoupled fault have no significant influence on the residual. Figure 6.4 shows how the monitored faults influence the

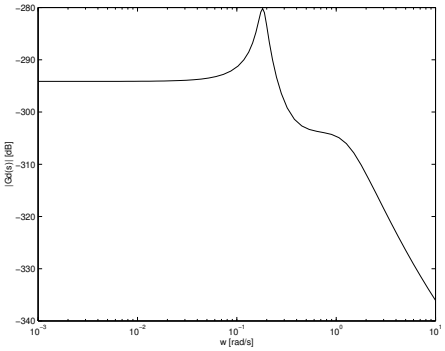


Figure 6.3: Singular value of the transfer function from x to r.

residual which clearly shows that fault influence is significantly larger than influence from the decoupled fault and control signals plotted in Figure 6.3. The leftmost plot in Figure 6.4 also shows that DC-gain

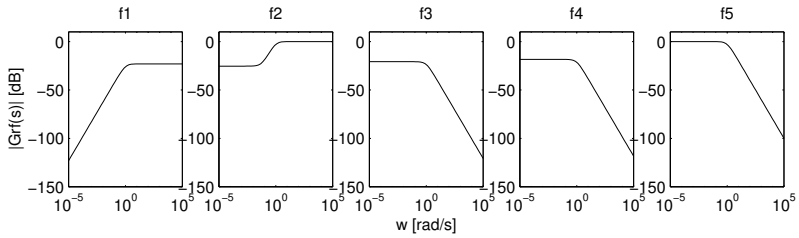


Figure 6.4: Magnitude bode plots for the monitored faults to the residual.

from fault 1 to the residual is 0, and it is therefore difficult to detect since the effect in the residual of a constant fault f_1 disappears. This is of course unfortunate but it can be proven, using the results from Section 6.7, that this is inevitable.

6.11 Alternative methods

This section will briefly describe two common alternative solutions to the residual generation problem: *diagnostic observers* and the *parity space approach*. These methods are included partly to give general knowledge on existing methods, and partly because they can be used to highlight connections to non-linear methods that are described in Chapter 7.

6.11.1 Diagnostic Observers

A common way of generating residuals is by using observers. This section gives a brief overview of observer design methods for fault diagnosis, so called diagnostic observers. In the linear case, which is studied here, strong equivalence relations exist between observer based design methods and the methods described previously in this chapter. See for example (Ding et al., 1999) for further details on these relations. Because of this strong relation between consistency relation based approaches and observer based approaches, linear observers are addressed only briefly here. The non-linear case is addressed in more detail in Chapter 7.

The basic idea is simple. Consider a linear model with no disturbances

$$\begin{aligned} \dot{x} &= Ax + Bu + Mf \\ y &= Cx \end{aligned}$$

A residual generator for this system, designed to detect the fault f can then quite naturally by designed by estimating the state x and then computing a residual according to $r = y - C\hat{x}$. The residual generator will then be in the form

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + Bu + K(y - C\hat{x}) \\ y &= C\hat{x} \end{aligned}$$

where K is the observer gain chosen such that $A - KC$ has all stable eigenvalues. To generalize the above approach to systems also with disturbances is not immediate. There are many such approaches, of varying complexity, and below a common approach is outlined.

A general linear diagnostic observer can be written as

$$\begin{aligned} \dot{w} &= Fw + Ky + Ju \\ r &= L_1 w + L_2 y + L_3 u \end{aligned}$$

where w is an estimate of the transformed state vector Tx. If T is of full rang, i.e. invertible, the observer is called a full state observer. When using observers for control, often the whole state vector needs to be estimated to be used for state-feedback. But for diagnosis, the state itself is not of importance unless a state is directly connected to a fault. So, a diagnostic observer need not be of the same order as the number of states in the supervised system.

Assume that the system under supervision is described by the realization

$$\begin{aligned} \dot{x} &= Ax + Bu + Mf + Ed \\ y &= Cx + Du + Nf + Hd \end{aligned}$$

where f is the fault signal and d the disturbances. Let the estimation error be $e = Tx - w$. The error dynamics of the observer then becomes

$$\begin{aligned} \dot{e} = T\dot{x} - \dot{w} &= T(Ax + Bu + Mf + Ed) - (Fw + Ky + Ju) = \\ &= Fe + (TA - FT - KC)x + (TB - KD - J)u + TMf + \\ &\quad + (TE - KH)d + (TM - KN)f \end{aligned} \tag{6.25}$$

The output equation becomes

$$\begin{aligned} r &= L_1(Tx - e) + L_2(Cx + Du + Nf + Hd) + L_3 u = \\ &= -L_1 e + (L_1 T + L_2 C)x + (L_2 D + L_3)u + L_2 Hd + L_2 Nf \end{aligned} \tag{6.26}$$

In the fault free case we require $r = 0$, which can be achieved if $e = 0$ is a globally stable point of equilibrium, regardless of process state, inputs or disturbances. Also, process state, inputs and disturbances should not influence the output equation. Imposing those restrictions

on (6.25) and (6.26) imposes the following conditions on the observer matrices:

$$\begin{aligned} TA - FT - KC &= 0 & TB - KD - J &= 0 \\ TE - KH &= 0 & L_1T + L_2C &= 0 \\ L_2D + L_3 &= 0 & L_2H &= 0 \end{aligned}$$

In addition, for the observer/residual generator to be stable, matrix F must have all its eigenvalues in the open left half plane.

In a design step, matrices T, F, K, J, L_1, L_2, and L_3 have to be found that satisfies the constraints above. The design techniques used for solving the design problem based on the constraints is often quite involved. Here only references to the most common approaches are given. The geometric approach to fault detection (Massoumnia et al., 1989) is one of the earliest observer based approaches. Another popular technique, described in (Patton, 1994), is the *eigenstructure assignment approach.*

Finally, we conclude with an example illustrating a particular property of observers for diagnosis that is used also in Chapter 7 where non-linear observers are used to generate residuals.

Example 6.15 Consider a linear system with two outputs and corresponding additive sensor faults f_1 and f_2:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y_1 &= C_1x + f_1 \\ y_2 &= C_2x + f_2 \end{aligned}$$

A diagnostic observer for this system can be constructed as

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + Bu + K(y_1 - C_1\hat{x}) \\ r &= y_1 - C_1\hat{x} \end{aligned}$$

As seen, the observer does only use one of the process outputs and therefore the fault f_2 is decoupled. By noting that the feedback term, that equals Kr, contains the residual, one might suspect that the feedback would force the estimation error, and the residual, to become zero. This is however not the case as can be seen if the estimation error $e = x - \hat{x}$ is studied. The dynamics of the estimation error is

$$\dot{e} = (A - KC)e - Kf_1$$

If $f_1 \neq 0$, $e = 0$ is not a stationary solution to this equation. Therefore the feedback can generally not force the residual to become zero.

There is however one case in which the stationary response of the residual will become zero. Consider the residual expressed in the estimation error and the fault:

$$r = Cx + f_1 - C\hat{x} = Ce + f_1 = \big(- C(sI - A + KC)^{-1}K + 1\big) f_1$$

In this expression it can be seen that the steady state gain of the transfer function from f_1 to r is $C(KC - A)^{-1}K + 1$. That is, the stationary response of the residual differs from zero (the fault is strongly detectable) if and only if

$$C(KC - A)^{-1}K + 1 \neq 0$$

6.11.2 The Parity Space Approach

One of the first approaches to linear residual generation was proposed in (Chow and Willsky, 1984). This approach to design linear consistency relations for linear systems is sometimes called the *parity space approach*, or the Chow-Willsky scheme. The approach is based on a state-space model:

$$\dot{x} = Ax + B_u u + B_d d + B_f f \tag{6.27a}$$
$$y = Cx + D_u u + D_d d + D_f f \tag{6.27b}$$

where $x \in \mathbb{R}^n$ is the state, $y \in \mathbb{R}^m$ the measurement vector, $u \in \mathbb{R}^{n_u}$ the control input, $f \in \mathbb{R}^{n_f}$ the faults, and $d \in \mathbb{R}^{n_d}$ the disturbance. Now by repeated substitution of (6.27a) into (6.27b), we obtain

$$Y(t) = Rx(t) + QU(t) + HV(t) + PF(t) \tag{6.28}$$

where

$$Y(t) = \begin{bmatrix} y(t) \\ py(t) \\ \vdots \\ p^{\rho}y(t) \end{bmatrix} \qquad R = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{\rho} \end{bmatrix}$$

$$Q = \begin{bmatrix} D_u & 0 & 0 & \dots \\ CB_u & D_u & 0 & \dots \\ \vdots & & \ddots & \\ CA^{\rho-1}B_u & \dots & CB_u & D_u \end{bmatrix} \qquad U(t) = \begin{bmatrix} u(t) \\ pu(t) \\ \vdots \\ p^{\rho}u(t) \end{bmatrix}$$

$$H = \begin{bmatrix} D_d & 0 & 0 & \dots \\ CB_d & D_d & 0 & \dots \\ \vdots & & \ddots & \\ CA^{\rho-1}B_d & \dots & CB_d & D_d \end{bmatrix} \qquad V(t) = \begin{bmatrix} d(t) \\ pd(t) \\ \vdots \\ p^{\rho}d(t) \end{bmatrix}$$

$$P = \begin{bmatrix} D_f & 0 & 0 & \dots \\ CB_f & D_f & 0 & \dots \\ \vdots & & \ddots & \\ CA^{\rho-1}B_f & \dots & CB_f & D_f \end{bmatrix} \qquad F(t) = \begin{bmatrix} f(t) \\ pf(t) \\ \vdots \\ p^{\rho}f(t) \end{bmatrix}$$

The size of Y is $(\rho+1)m \times 1$, R is $(\rho+1)m \times n$, Q is $(\rho+1)m \times (\rho+1)n_u$, U is $(\rho+1)n_u \times 1$, H is $(\rho+1)m \times (\rho+1)n_d$, F is $(\rho+1)n_f \times 1$, P is $(\rho+1)m \times (\rho+1)n_f$, and V is $(\rho+1)n_d \times 1$. The constant ρ determines the maximum order of the consistency relation. This can be seen by studying the definitions of the vectors Y and U.

A linear residual generator is a linear combination of measurements and control signals. This can be represented by a row vector w of length $(\rho+1)m$, a consistency relation $K(p)z = 0$ can be formed as

$$K(p)z = w(Y - QU) = 0 \tag{6.29}$$

The function $K(p)z$ can also be written as

$$K(p)z = w\begin{bmatrix} \Psi_y(p) & -Q\Psi_u(p) \end{bmatrix}\begin{bmatrix} y \\ u \end{bmatrix} \tag{6.30}$$

where

$$\Psi_y(p) = \begin{bmatrix} I_m \\ pI_m \\ \vdots \\ p^{\rho}I_m \end{bmatrix} \qquad \Psi_u(p) = \begin{bmatrix} I_{n_u} \\ pI_{n_u} \\ \vdots \\ p^{\rho}I_{n_u} \end{bmatrix}$$

Equation (6.28) implies that the following equality will hold:

$$K(p)z = w(Rx + HV + PF) \tag{6.31}$$

Since the consistency relations must hold in the fault free case and the disturbances must be decoupled, (6.31) implies that w must satisfy

$$w\,[R\,H] = 0 \tag{6.32}$$

For use in fault detection, it is also required $K(p)z$ in (6.31) is *not* zero in the case of faults. This is assured by verifying that

$$wP \neq 0 \tag{6.33}$$

Approach Summary

In conclusion, using the parity space approach, a consistency relation is constructed by following the procedure

1. Compute all the matrices in (6.28) and find a w such that (6.32) and (6.33) are fulfilled.
2. Form a consistency relation according to (6.30).
3. Form a residual generator according to

   $$R(p) = d^{-1}(p)K(p)$$

   where $d(s)$ is a stable polynomial with degree at least as high as the row-degree of $K(s)$.

When setting up all the matrices in (6.28), the constant ρ need to be determined. It can be proven that no ρ greater than n need to be considered since any consistency relation can be written as a linear combination of consistency relations of order $\leq n$. Thus, there is no reason to use a ρ larger than system order. Can there be a reason why it might be suitable to choose a ρ less than the system order? This question is left as an exercise for the interested reader.

Appendix

6.A State-space realization

A single input, single output system with a transfer function

$$G(s) = \frac{b_1 s^{n-1} + \cdots + b_{n-1}s + b_n}{s^n + a_1 s^{n-1} + \cdots + a_{n-1}s + a_n}$$

can be realized in state-space form in many ways. One form is the observer canonical form

$$\begin{aligned} \dot{w} &= \begin{bmatrix} -a_1 & 1 & 0 & \dots & 0 \\ -a_2 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & & \\ -a_{n-1} & 0 & 0 & \dots & 1 \\ -a_n & 0 & 0 & \dots & 0 \end{bmatrix} w + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix} u \\ y &= \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \end{bmatrix} w \end{aligned}$$

It is suitable to use for realizing residual generators in state-space form since it is straightforward to generalize the above form to cover also multiple input, single output systems.

6.B Null-Spaces

Let $A \in \mathbb{R}^{n \times m}$ be a matrix with rank r. The *right* null-space $\mathcal{N}_R$ of matrix A is then given by the space of all solutions to the equation $Ax = 0$, i.e.

$$\mathcal{N}_R = \{x : Ax = 0\}$$

For example, if

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 5 \\ 1 & 0 & 1 \end{bmatrix}$$

the null-space is given by

$$\mathcal{N}_R = \text{span} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

where span M here refers to the space spanned by the *columns* of matrix M. The dimension of $\mathcal{N}_A$ is directly obtained from the laws of dimensionality to be the number of columns minus the matrix rank, i.e. $m - r$. Thus, if A has full column rank, the dimension of the null-space is 0, i.e. $x = 0$ is the only solution to $Ax = 0$. In the same way as we can speak of the right null-space, we can also consider the *left* null-space which is naturally defined as

$$\mathcal{N}_L = \{v : vA = 0\}$$

Similar to the right null-space, the dimension of the left null-space is given by the number of rows minus the rank, i.e. $n - r$. For the matrix A above, the left null-space is given by

$$\mathcal{N}_L = \text{span} \begin{bmatrix} 3 & -2 & 1 \end{bmatrix}$$

where span M here refers to the space spanned by the *rows* of matrix M.

6.C Polynomial algebra

This section will provide basic knowledge of the concepts needed in the main text. For a thorough treatment of polynomial algebra in connection with linear systems, see e.g. (Kailath, 1980).

A polynomial matrix in a variable s is a matrix where each element is a scalar polynomial in s and a rational matrix is when the elements are rational functions in s. For example

$$A(s) = \begin{bmatrix} s+1 & 2 \\ 0 & as^2 \end{bmatrix}$$

is a polynomial matrix. Normal rank for a polynomial matrix is then defined as

Definition 6.5 (Normal rank)**.** Let $A(s) \in \mathbb{R}(s)^{m \times n}$ be a polynomial matrix, then the normal rank of $A(s)$ is

$$\max_{s \in \mathbb{C}} \text{rank}\, A(s)$$

The word *normal* is often not said explicitly and one often says rank of a polynomial matrix when meaning normal rank.

This means that, for example, a square matrix is invertible if and only if it has full normal rank. Matrix $A(s)$ has rank 2 if $a \neq 0$ and rank 1 if $a = 0$.

Left and right null-spaces for a matrix is defined exactly as for constant vector spaces, the left null-space of a matrix $M(s)$ is then defined as all rational row vectors $v(s)$ such that

$$v(s)M(s) = 0$$

For example, let

$$M(s) = \begin{bmatrix} s+1 \\ s+2 \\ 1 \end{bmatrix}$$

Since $M(s)$ has three rows and rank 1, the left null-space has dimension $3 - 1 = 2$. A basis $N_M(s)$ for the left null-space is then a matrix with two linearly independent rows. It is immediate that for example

$$N_M(s) = \begin{bmatrix} \frac{1}{s+1} & 0 & -1 \\ 0 & \frac{1}{s+2} & -1 \end{bmatrix}$$

is a basis for the left null-space of $M(s)$. However, this basis includes rational elements. It is easy to prove that there always exists a *polynomial basis* for the same space. For example, multiply the first row with $s + 1$ and the second with $s + 2$ to obtain the polynomial basis

$$N_M(s) = \begin{bmatrix} 1 & 0 & -(s+1) \\ 0 & 1 & -(s+2) \end{bmatrix}$$

The polynomial basis above is however not the simplest one available. An example of an even simpler basis for the same null-space is obtained by replacing the first row with the first minus the second row

$$N_M(s) = \begin{bmatrix} 1 & -1 & 1 \\ 0 & 1 & -(s+2) \end{bmatrix}$$

This is a simpler basis in the sense that the degrees of the polynomials are lower. This last basis is called a *minimal polynomial basis* for the left null-space of $M(s)$. When, in the text in this chapter, referring to bases for null-spaces, such minimal bases are normally what is considered. For the understanding of the residual generator design methods in this chapter, no detailed knowledge on how to compute the basis is necessary. It is sufficient to know their meaning and that there exists efficient tools, e.g. in Matlab, to perform the computations.

6.D Some complementary lemmas and proofs

This section includes some technical lemmas, along with complete proofs, needed to prove main results in the main text. Also, some of the proofs omitted from the main text is included.

Lemma 6.7. The behavior $\mathcal{O}$, defined by (6.9), can for a static model (6.10) be written as

$$\mathcal{O} = \{z | N_H L z = 0\}$$

Proof. Let $\mathcal{O}_1 = \{z | N_H L z = 0\}$. It is then immediate that $\mathcal{O} \subseteq \mathcal{O}_1$ since for any element $z \in \mathcal{O}$ it holds that there exists an x such that

$$Hx + Lz = 0$$

Pre-multiplication of the above from the left by N_H gives that $N_H L z = 0$ which implies that $z \in \mathcal{O}_1$. The converse is also true. First note that the rows of N_H span the orthogonal space $(\operatorname{Im} H)^\perp$ of $\operatorname{Im} H$. A vector $v \in \operatorname{Im} H$ if and only if v is orthogonal to all vectors in its orthogonal complement. This means that if $z \in \mathcal{O}_1$ it holds that

$$Lz \in \operatorname{Im} H$$

which means that there exists a vector x such that

$$Lz = Hx$$

which implies that $z \in \mathcal{O}$. We have then proven that

$$\mathcal{O} \subseteq \mathcal{O}_1 \subseteq \mathcal{O}$$

which implies that $\mathcal{O} = \mathcal{O}_1$ which ends the proof. ■

6.D.1 Proof of Lemma 6.3

Proof.
If part: Assume there exists a residual generator $R(p)$ sensitive to fault f_i. According to Theorem 6.2 and Definition 6.3, this residual generator can be written as (6.18) and $G_{rf_i}(s)$ in (6.22) is non-zero. Then it holds that

$$r = d^{-1}(p)\gamma(p)N_H(p)L(p)z = \underbrace{-d^{-1}(p)\gamma(p)N_H(p)F_i(p)}_{\neq 0} f_i \qquad (6.34)$$

Further, due to the rank assumption (6.8), for *any* f there exists x and z such that they all are consistent with the model equation (6.5). Thus, there exists an f_i such that the right hand side of (6.34) is non-zero. This implies that $N_H(p)L(p)z \neq 0$ which, according to Lemma 6.7, implies that $z \notin \mathcal{O}$ which ends the proof of the if-part.

Only-if part: Assume that f_i is detectable, then there exist signals x, f_i, and z consistent with model (6.5) such that $z \notin \mathcal{O}$ which, using Lemma 6.7, then implies that $N_H(p)L(p)z \neq 0$. Since $N_H(p)L(p)z = -N_H(p)F_i(p)f_i$ it holds that

$$-N_H(p)F_i(p) \neq 0$$

It then follows immediately that there exists $\gamma(p)$ and $d(p)$ such that

$$-d^{-1}(p)\gamma(p)N_H(p)F_i(p) \neq 0$$

Thus, there exists a residual generator $R(p) = d^{-1}(p)\gamma(p)N_H(p)L(p)$ such that $G_{rf_i}(s) \neq 0$ which ends the proof. ■

Chapter 7
Nonlinear Residual Generation

This chapter gives an introduction to nonlinear residual generation and deals both with approaches based on nonlinear consistency relations and methods based on state observation.

In general, the nonlinear problem has no general and complete solution like what was presented for the linear case in Chapter 6. Therefore, a goal of this chapter is to introduce basic concepts and provide some basic tools on possible approaches to the non-linear problem in addition to the general principles that were discussed in Chapter 4.

The chapter is divided into two main parts, the first considers using consistency relations for the non-linear problem and the second part deals with non-linear state-observation techniques and how they can be successfully used for fault diagnosis.

7.1 Nonlinear Consistency Relations

A nice property of the linear methods presented in Chapter 6 was that they could generate complete solutions, i.e. all residual generators with desired fault sensitivity properties could be obtained. As noted above, for general non-linear systems, no such complete result is possible

that provide a simple parameterization of all residual generators or consistency relations. In spite of this, the rest of this section will be spent on illustrating principles on how to derive residual generators based on non-linear consistency relations and also devote some time on an interesting class of non-linear systems for which quite strong results is possible to derive.

7.1.1 Deriving Non-Linear Consistency Relations

Example 4.7 in Chapter 4 showed how a consistency relation could be derived by differentiating the measurement equation and then applying a clever transformation to the obtained set of equations. A similar method of repeated differentiation of the measurement equation can be used to derive consistency relations that decouple not only the internal states but also unknown disturbances. This approach can be seen as a non-linear counterpart to the linear approach in Section 6.11.2, the Chow-Willsky scheme, where the measurement equation was differentiated n times and the state-variables and disturbances were eliminated by applying a suitable linear transformation to the equations.

The procedure is best illustrated by a small example. The example below illustrates decoupling in nonlinear systems:

Example 7.1 Consider a nonlinear state-space description

$$\dot{x}_1 = -x_1 x_2 + \gamma u \tag{7.1a}$$
$$\dot{x}_2 = x_3 \tag{7.1b}$$
$$\dot{x}_3 = -\gamma x_1 \tag{7.1c}$$
$$y_1 = x_1 \tag{7.1d}$$
$$y_2 = x_2 \tag{7.1e}$$

where γ models a fault we wish to decouple, i.e. a consistency relation independent of γ is wanted.

The following relations are immediate by differentiating both measurement equations and making obvious substitutions:

$$\dot{y}_1 + y_1 y_2 - \gamma u = 0$$
$$\ddot{y}_2 + \gamma y_1 = 0$$

But none of these is a consistency relation where γ is decoupled since both relations include the unknown, possibly time-varying, variable

γ. This is however not a problem, since γ easily can be eliminated by multiplying the first relation with y_1, the second with u, and adding the two. The resulting consistency relation is then

$$y_1\dot{y}_1 + y_1^2 y_2 + u\ddot{y}_2 = 0$$

which is a nonlinear consistency relation that holds for all y_1, y_2, γ that satisfies model equations (7.1).

Now, lets summarize the procedure illustrated in the example above in a more general model setting. Let the model be given by a set of (differential-)equations:

$$g_i(y, u, x, d, f) = 0, \quad i = 1, \ldots, m \tag{7.2}$$

Thus, the model consists of a set of functions g_i which are functions in y, u, x, d, f (and their derivatives). Without loss of generality it can be assumed that $f = 0$ corresponds to a fault-free system. A procedure to find consistency relations can then be stated as

1. Differentiate the model equations a suitable number of times. This is of course only necessary if g_i are differential equations.

2. Collect all equations and make clever substitutions such that a function $h(y, u, f)$ is derived for which it holds that

$$h(y, u, f) = 0 \tag{7.3}$$

3. A consistency relation for (7.2) is then

$$h(y, u, 0) = 0 \tag{7.4}$$

   This, because relation (7.3) holds for all y, u, and f. In the fault-free case, $f = 0$, the relation becomes (7.4).

   In general, the function h is a function of derivatives of y and u, which are normally not known. If these are known (or estimated), the computational form of a residual generator could be stated:

$$r^{comp}(y, u) = h(y, u, 0)$$

This procedure leaves three open questions:

1. In the first step, what is a suitable number of times that the model equations should be differentiated? For linear models there is no need to differentiate the model equations more times than the order of the model. In the non-linear case this is not generally true. However, the order of the model is a good indicator of the maximum number of times to differentiate the model equations.

2. Is there a systematic way to find the clever manipulations necessary to eliminate the unknown variables (and also a suitable subset of the fault variables)? Unfortunately, but not surprising, the answer to this question is also no. However, manipulations by hand and the use of computer-algebraic tools can get you far. Also, as will be illustrated in Section 7.1.2, for specific classes of systems more systematic methods are available.

3. In step 3 of the procedure, the consistency relation is used to form a realizable residual generator. In the linear case, this step was straightforward and was illustrated in Example 4.8 and in general in Chapter 6. A corresponding procedure for the nonlinear case is *not* direct, i.e. to go from a nonlinear consistency relation to an implementable filter is in many cases difficult. The following example illustrates this difficulty.

It should be noted that the procedure outlined above utilizes only non-differential tools that can not handle derivatives in an efficient manner. Looking into the theory of differential-algebra (Ritt, 1950) there are tools available that handles step 1 and 2 automatically. One such tool is the `diffalg`-package (Hubert, 2008), based on theory presented in e.g. (Hubert, 2005), which is included in the computer algebraic tool Maple.

Example 7.2 Consider the consistency relation from Example 7.1

$$y_1\dot{y}_1 + y_1^2 y_2 + u\ddot{y}_2 = 0$$

If all derivatives of y and u were known, a residual could be computed as

$$r = y_1\dot{y}_1 + y_1^2 y_2 + u\ddot{y}_2$$

which would be zero in the fault free case and non-zero when the original model no longer describes system dynamics.

As before, derivatives of known signals are not normally known which means that it is not possible to directly compute the residual like this. In the linear case, it was shown in Example 4.8 and in Chapter 6 how this could easily be circumvented by introducing stable residual generator dynamics and a residual could be computed without the need to know differentiated signals. For this non-linear case, the simple linear approach is unfortunately not directly applicable.

To circumvent the problem with unknown derivatives, there are a number of possible approaches. The first is the most straightforward, to estimate the derivatives and then compute the residual. A simple, but not always sufficiently good solution is to use an approximately differentiating filter

$$\dot{y} \approx \frac{s}{1 + sT_d} y$$

This typically only works when estimating low order derivatives in a relatively noise-free environment. A more robust way to estimate the derivatives is to utilize for example spline interpolation techniques. In such an approach, analytical functions are used to approximate the measured signal. The derivatives of the signal are then computed using the analytical expressions rather than the measured signal. To get some smoothing, i.e. to filter out the measurement noise, the parameters in the spline functions are often selected to minimize a criterion looking like:

$$\min_{\theta} \; P \sum_{i=1}^{N} (y_i - f(t_i; \theta))^2 + (1 - P) \int_{t=t_{\min}}^{t=t_{\max}} \ddot{f}(t; \theta)^2 dt$$

where θ are the parameters in the spline functions $f(t)$, y_i the measured signal, and $P > 0$ a design variable. Letting $P = 1$ we get no smoothing of the splines, while letting P approach 0 we get more and more smoothing. Smoothing in this case means minimizing the curvature of the resulting splines.

A third solution is to transform the original, time-continuous model, into a non-linear time-discrete model. Then, an analogous design can be made using the time-discrete model, which results in time-discrete consistency relations. These time-discrete consistency relations can be directly used as residual generators (or arbitrary low-pass dynamics can be added) since no time-differentiated signals occur, only time-shifted known signals. Note that time discretization of non-linear models is in

general difficult. Approximations are however not difficult to produce, for example using Euler forward or higher order approximations of the derivative.

A fourth solution, would be to avoid the above approximations and try to generalize the linear solution to the non-linear case by adding low-pass dynamics and writing the residual generator on state-space form, similar to what was done in Example 4.8. Thus, it would be desirable to be able to add stable (possibly linear) low-pass dynamics

$$r + \alpha_1 \dot{r} + \alpha_2 \ddot{r} = r^{comp}(y, u)$$

and finding an explicit state-space realization of the residual generator. Unfortunately, this approach is more difficult than the approaches outlined above. The mathematical conditions for this to be possible are rather strict and are only fulfilled in special cases. Below, a small example is constructed such that the realization step is immediate. The example is included to demonstrate the idea on how to use realization theory.

Example 7.3 Consider a system described by the differential equation

$$\begin{aligned} \dot{x} &= -\sin^3(x)(u+f)^2 \\ y &= x + (u+f) \end{aligned}$$

where f is an actuator fault that has to be supervised. A consistency relation for the system above can easily be derived by differentiating the measurement equation

$$\dot{y} = \dot{x} + (\dot{u} + \dot{f})$$

and eliminating the state-variable x by substitution of the model equation

$$\dot{y} = \dot{x} + \dot{u} + \dot{f} = -\sin^3(x)(u+f)^2 + \dot{u} + \dot{f} = \\ -\sin^3(y-u-f)(u+f)^2 + \dot{u} + \dot{f} \quad (7.5)$$

Now, observe that $-\sin^3(y-u)u^2 + \dot{u}$ is the right hand side with $f = 0$, and by subtraction on both sides in (7.5) we obtain

$$\dot{y} + \sin^3(y-u)u^2 - \dot{u} = \dot{f} - \sin^3(y-u-f)(u+f)^2 + \sin^3(y-u)u^2 \quad (7.6)$$

Denote the right hand side with the expression $h(y, u, f)$ and note that $h(y, u, 0)$ is zero, i.e. the left hand side is a consistency relation. Further, this means that if $\dot{y}$ and $\dot{u}$ were known, the left hand side of (7.6) could be used to compute a residual that could be used to detect the actuator fault according to:

$$\dot{y} + \sin^3(y-u)u^2 - \dot{u} = \begin{cases} 0 & f \equiv 0 \\ c(t) \not\equiv 0 & f \not\equiv 0 \end{cases}$$

Here, the time derivatives are assumed to be unknown. In the light of the previous discussion, add stable first-order linear dynamics to the left hand side of (7.6), i.e.

$$r + \alpha\dot{r} = \dot{y} + \sin^3(y-u)u^2 - \dot{u} \tag{7.7}$$

with $\alpha > 0$ and try to find an explicit state-space representation of (7.7) with y and u as inputs and the residual r as output. The choice of α corresponds to $d(s) = 1 + \alpha s$ in (6.17) and for any $\alpha > 0$, the residual generator will be stable. In this particular case, use the state variable $z = \alpha r - (y - u)$. Straightforward manipulations then give

$$\begin{aligned} \dot{z} &= \alpha\dot{r} - (\dot{y} - \dot{u}) = -r + \sin^3(y-u)u^2 = \\ &= -\frac{1}{\alpha}z - \frac{1}{\alpha}(y-u) + \sin^3(y-u)u^2 \\ r &= \frac{1}{\alpha}z + \frac{1}{\alpha}(y-u) \end{aligned}$$

The internal form of this filter is

$$r + \alpha\dot{r} = h(y, u, f)$$

which will be 0 in the fault-free case and non-zero when a fault occurs.

In general, linear dynamics is not sufficient to be able to form a state-space description of the residual generator. The added dynamics of the residual generator is free for the designer to choose (as long as it is stable). In the example, linear dynamics $r + \alpha\dot{r}$ was added. For the general problem, this can be any dynamics $g(r, \dot{r}, \dots)$ such that $r = 0$ is a globally stable operating point of the differential equation

$$g(r, \dot{r}, \dots) = 0$$

7.1.2 Consistency Relations for Polynomial Systems

A possible solution to get more general results is to reduce the class of systems considered, i.e. the non-linearities in the model equations are assumed to be of a certain class or shape. An example of such a class where more general and systematic methods are possible are polynomial systems. A system of polynomial differential equations means that the functions g_i in (7.2) are polynomial in their variables. A polynomial in variables $x_1, \ldots, x_n$ with coefficients from $\mathbb{R}$ is defined as

$$p(x_1, \ldots, x_n) = \sum_j c_j \prod_i x_i^{m_{ij}} \quad c_j \in \mathbb{R}, m_{ij} \in \mathbb{N} = \{0, 1, 2, \ldots\}$$

and is denoted $p \in \mathbb{R}[x_1, \ldots, x_n]$. Examples of polynomials in $\mathbb{R}[y_1, y_2, u, \dot{u}, \ddot{y}_2]$ are

$$g_1(y, u) = y_1^2 \dot{u} y_2 - 3uy_2 \qquad g_1(y, u) = y_1 \ddot{y}_2 + 7uy_2$$

For example, the non-linearities in Example 7.1 is examples of polynomial non-linearities.

When considering systems described by such polynomial differential equations, powerful tools from commutative and differential algebra, such as Gröbner bases and characteristic sets can be used. They provide systematic methods to perform the second step in the procedure in a similar fashion to what was presented for the linear problem.

This section is not intended to give more than a brief glimpse on what these mathematical tools can provide. Readers who are interested in these subjects are referred to the excellent book (Cox et al., 1996) on basic commutative algebra and the advanced book on basic differential algebra (Ritt, 1950).

The second step in the procedure from the last section was to eliminate unknown variables, and possibly a subset of the faults, from a set of model equations to form a consistency relation. For a system of linear equations, this could be performed by Gaussian elimination where the model equations is transferred into an equivalent, triangular, representation. In the Gaussian elimination, the variables are ordered such that the variables with the highest ordering are eliminated first. Consistency relations where unknown variables were eliminated can then directly be seen in the triangular system of equations. For polynomial non-linearities, a similar procedure is available using so called

Gröbner bases. Computing a Gröbner basis can, loosely, be said to be a non-linear version of Gaussian elimination.

Without providing any proof or theoretical background it is here stated that: Computing a Gröbner basis for a set of polynomial equations[1], given a suitable variable ordering and lexicographic monomial ordering, gives an equivalent[2] triangular system of polynomial equations where consistency relations can directly be extracted. The notion *monomial ordering* is not developed further here, but since it may be necessary for a user of a computer algebraic tool to state the monomial order to use, it is stated here which ordering to use. In case of variable elimination, lexicographic ordering (sometimes called elimination ordering) must be used.

The procedure is illustrated in Example 7.4 which also include a sample Mathematica session.

Example 7.4

Consider the following set of equations

$$\begin{aligned} x^2 + y + z - 1 &= 0 \\ x + y^2 + z - 1 &= 0 \\ x + y + z^2 - 1 &= 0 \end{aligned}$$

Suppose we wish to eliminate all variables but z. Then, compute a Gröbner basis with lexicographic monomial ordering and variable order $x \succ y \succ z$. In a Mathematica session, this is done by:

```
In[1]:= F={-1+x^2+y+z, -1+x+y^2+z, -1+x+y+z^2};
        GroebnerBasis[F,{x,y,z}]

Out[2]= {-1 + x + y + z^2, y^2 + z-z^2-y,
         2yz^2 + z^4-z^2, z^2 -4z^3 + 4z^4-z^6}
```

The Mathematica command `GroebnerBasis` has lexicographic monomial ordering as default ordering. Note the triangular structure of the Gröbner basis, the last polynomial is a function of z only, the second and third depends on y and z while the first depends on all three x, y, z. All this according to the chosen variable ordering.

[1] The language used here is intentionally sloppy. A correct formulation would need theoretical concepts not introduced in this text for the sake of brevity.

[2] In the sense that it has equal solution set.

Also, from the Gröbner basis it is immediate to extract relations where x and y has been eliminated, in this case $z^2 - 4z^3 + 4z^4 - z^6 = 0$.

A few words of caution is warranted

1. Computing Gröbner bases is a computationally very complex in both time and space(memory) and is only really feasible on sets of equations of limited size. Although, computational complexity is strongly dependent on variable ordering, shape of equation system etc., experience has shown that ≈ 20 equations in 10 to 15 variables is close to what an ordinary personal computer is capable of handling. There are ways to try to lower computational complexity, e.g. by structural analysis of the system of equations, see e.g. (Frisk, 2001) for more details.

2. Gröbner basis methodology is a non-differential tool, i.e. x and $\dot{x}$ are considered to be two completely unrelated variables. Thus, the model equations need to be differentiated "by-hand" to take into account dynamic relations between variables. This is described in the first step of the procedure outlined previously. Assume the model description consist of N equations. After differentiating the model equations m times, we are stuck with mN equations. This means that the number of equations increases rapidly when differentiating and this is a problem due to the complexity issues described above.

When considering a class of models it is of course relevant to see how large this class is and what types of systems that can be modeled. It is interesting to see that the class of systems described by polynomial differential algebraic equations is larger than one might first think. This is because all elementary functions, e.g. sin, arcsin, log, and exp can all be written as a solution to a polynomial differential equation. For example the nonlinear system

$$\dot{y} + e^{-ay} - u = 0$$

can be rewritten on polynomial state-space form as:

$$\begin{aligned} \dot{z} &= -az\dot{y} = -az(u - z) \\ \dot{y} &= -z + u \end{aligned}$$

This is made by the variable substitution $z = e^{-ay}$ which is the solution to the polynomial differential equation $\dot{z} + az\dot{y} = 0$. Another example is the relation $x = \sin y$ that can be written, in polynomial form, as $\dot{x}^2 = \dot{y}^2(1 - x^2)$.

7.2 Nonlinear Diagnostic Observers

As was described in Chapter 6, observers is a popular way of generating residuals, perhaps even more popular in the nonlinear case than in the linear case. Here nonlinear observers are explored to expand our toolbox for attacking nonlinear diagnosis problems.

There are two main obstacles when using observers for fault diagnosis:

1. Decoupling of faults/disturbances to make fault isolation possible.

2. How to choose observer form and ensure observer stability.

The second issue is a well studied problem in non-linear control literature. Although a difficult problem, many results exist and this will be addressed briefly in Section 7.2.1 where a few basic results from nonlinear observer theory is presented.

The first issue is however difficult and no general solution exists. Compare with the linear case described in Section 6.11.1, where also the decoupling problem was described to be rather involved. This is even more true in the non-linear case. However, for some restricted classes of systems, e.g. control-affine systems (Persis and Isidori, 2000, 2001), or bi-linear systems (Kinnaert, 1999), some quite general methods have been presented. The general approach of most methods is to, given the model equations

$$\begin{aligned} \dot{x} &= f(x, u, f, d) \\ y &= h(x, u, f, d) \end{aligned}$$

find transformations such that the model equations are transformed into two parts where one subsystem is not affected by the disturbance.

For example, if the system can be transformed into a form

$$\begin{aligned}\dot{x}_1 &= f_1(x_1, y_1, y_2, u, f)\\ \dot{x}_2 &= f_2(x_1, x_2, y_1, y_2, u, f, d)\\ y_1 &= h_1(x_1, u, f)\\ y_2 &= h_2(x, u, f, d)\end{aligned}$$

an observer for x_1 using measurement y_1 can be used to form a residual $r = y_1 - \hat{y}_1$ to detect the fault f. This partitioning of the system is in general difficult and beyond the scope of this text.

Because of this, we will only describe decoupling of faults in two simple but common cases:

1. Faults modeled as constant parameters, Section 7.2.2.
2. Sensor faults, Section 7.2.3.

But first, a few words on non-linear state observation.

7.2.1 Nonlinear Observers

Consider a general state-space formulation of a nonlinear system:

$$\dot{x} = f(x, u) \tag{7.8a}$$
$$y = h(x, u) \tag{7.8b}$$

where f and h are nonlinear functions describing system dynamics and the measurement equation. The observer problem is then to, from measurements y and control signals u, estimate the state-vector x.

The simplest form of an observer would be to just simulate the system, i.e. the observer equations can be stated as:

$$\dot{\hat{x}} = f(\hat{x}, u)$$

However, such an approach is highly sensitive towards disturbances, errors in initial conditions, and modeling errors. Since a measure of state-estimation quality is available in $y - \hat{y}$, it is natural to extend the observer equations with some feedback from this output-estimation error, e.g.:

$$\dot{\hat{x}} = f(\hat{x}, u) + l(y - h(\hat{x}, u)) \tag{7.9}$$

where $l(\cdot)$ is some nonlinear function. But how should the feedback term $l(\cdot)$ in (7.9) be chosen to ensure observer stability? In general, this question is unanswered but a simple approach is to mimic the linear case and make an additive correction based on the output estimation error, i.e. choose observer form

$$\dot{\hat{x}} = f(\hat{x}, u) + K(y - h(\hat{x}, u)) \tag{7.10}$$

where K is a matrix, the observer gain. Matrix K can be a constant, but also depend on e.g. estimated state and time, i.e. $K = K(\hat{x}, t)$.

The simplest approach is to use a constant observer gain and the remaining question is then how to select K to ensure stability. A natural way to choose the observer gain is to linearize the system equations around a working point (x_0, u_0), make a linear observer design e.g. by pole-placement or using a Kalman filter design, and use the obtained K in the observer structure (7.10). The procedure is then: compute the model matrices for the linearized model behavior

$$A = f_x(x_0, u_0), \quad C = h_x(x_0, u_0)$$

where f_x and h_x are the partial derivatives of f and h with respect to x. Then find K such that $A - KC$ has desired properties. The obtained observer can be suspected to work satisfactory in a neighborhood surrounding the linearization point. Note that, to be able to find a K such that $A - KC$ has desirable properties, i.e. is stable with suitable eigenvalues, it is necessary that the pair (A, C) is observable, i.e. the nonlinear system (7.8) is locally observable in operating point (x_0, u_0). To elaborate on this issue is beyond the scope of this text, but a brief discussion is included in Section 7.2.4.

Next section will briefly describe a common choice of non-linear state-estimator, the *extended Kalman filter*, which has proven useful in many applications for a long period of time. But before that, a brief outlook and list of references to some other well known state-observation techniques not mentioned further in this text is given. Not surprisingly, in literature, there is a plethora of advanced, classical, state-observation techniques. Good sources of descriptions of classical non-linear observers are the surveys (Walcott et al., 1987; Misawa and Hedrick, 1989). A well studied type of observer is the sliding mode observer (Slotine et al., 1987) that has been used frequently in connection with fault diagnosis. More recent advances is described in

(Moral and Grizzle, 1995) where state observation is treated more like an optimization problem and an interesting extension is also described in (Nikoukhah, 1995, 1998) where the class of observers is extended to differential-algebraic systems. A Bayesian approach to state observation are so called particle filters (Gordon et al., 1993) that has been used for fault diagnosis in (Li and Kadirkamanathan, 2001).

Extended Kalman Filter

In equation (7.10), an observer design was made using a linear approximation of system dynamics when computing the observer gain, for example using a steady-state Kalman filter design, and then use the full non-linear model with that observer gain. The most common way to extend the linear Kalman Filter to non-linear systems is the *Schmidt Extended Kalman Filter*, often called only Extended Kalman Filter or just EKF. Instead of linearizing around a single stationary operating point, it would be better to linearize around the true trajectory of the system. But of course, this trajectory is unknown and we do not know the current state. We do however have an estimate of the current state, which is our best current guess. Continuously linearizing around the current state-estimate is the basic idea of the (Schmidt) Extended Kalman Filter (Kailath et al., 2000).

The equations that define an EKF are different depending on if the model is given in continuous time or discrete time and if the observer runs in continuous time or discrete time. Here, only the equations for a discrete time observer for a discrete time system is presented. This is because stochastic noise is formally simpler to treat in discrete time. This basic setup illustrates the main idea and interested readers are referred to e.g. (Kailath et al., 2000) for more details on Kalman filtering. Consider the time discrete model

$$\begin{aligned} x_{t+1} &= f(x_t, u_t, w_t) \\ y_t &= h(x_t, u_t) + v(t) \end{aligned}$$

where w_t and v_t are zero-mean, white, processes with covariances R_t and Q_t respectively. Let $\hat{x}_{t_1|t_2}$ and $P_{t_1|t_2}$ denote the state estimate and estimation covariance at time point $t = t_1$ with observations up to time point $t = t_2$. Then the EKF estimate of the state, started with

$\hat{x}_{0|-1} = x_0$ and initial covariance $P_{0|-1} = \Pi_0$, is obtained by iterating

$$\begin{aligned}
\hat{x}_{t+1|t} &= f(\hat{x}_{t|t}, u_t, 0) \\
\hat{x}_{t|t} &= \hat{x}_{t|t-1} + K_t(y_t - h(\hat{x}_{t|t-1}, u_t)) \\
K_t &= P_{t|t-1}H_t^T(H_t P_{t|t-1} H_t^T + R_t)^{-1} \\
P_{t|t} &= (I - K_t H_t)P_{t|t-1} \\
P_{t+1|t} &= F_t P_{t|t} F_t^T + G_t Q_t G_t^T
\end{aligned}$$

where

$$F_t = f_x(\hat{x}_t, u_t, 0),\ H_t = h_x(\hat{x}_t, u_t),\ G_t = f_w(\hat{x}_t, u_t, 0)$$

and

$$f_x(x_0, u_0, w_0) = \left.\frac{\partial f(x,u,w)}{\partial x}\right|_{x=x_0, u=u_0, w=w_0}$$

and corresponding for the other partial derivatives.

This has been a brief look into a specific non-linear state-observation technique that is in common use. It is, by no means, an exhaustive description and there are many topics related to this subject that has not been touched here, e.g. stability of the EKF, implementation issues, non-linear observability, discretization and so on. Interested readers are referred to works cited above for more details on such subjects.

7.2.2 Decoupling of Fault Modeled as Constant Parameters

The previous section introduced nonlinear state-observers and some basic analysis/synthesis tools. Now it will be shown how such observers can be used for fault diagnosis. Since the general, nonlinear fault decoupling problem is generally still unsolved, smaller problem has to be addressed. This section focuses on the important special case where faults are modeled by a constant parameter.

First it is shown how constant parameters in linear and nonlinear dynamical systems can be estimated by nonlinear state-observers and then how such observers can be used to generate residuals where faults modeled by constant parameters are decoupled.

Example 7.5 Consider a linear system, described by equations:

$$\begin{aligned}
\dot{x} &= -\beta x + u \\
y &= x
\end{aligned}$$

where β is an *unknown* and *constant* parameter. Introduce a new state-vector $z = \binom{x}{\beta}$ and since β is constant, dynamics for state β is given by $\dot{\beta} = 0$, i.e. the extended state-space equations can be written as:

$$\dot{z} = \begin{pmatrix} -z_2 z_1 + u \\ 0 \end{pmatrix} \tag{7.11a}$$

$$y = z_1 \tag{7.11b}$$

where $z_1 = x$ and $z_2 = \beta$. Note that even though the original system was described by a linear model, the extended state-space description is nonlinear.

By designing a stable, nonlinear state observer for system (7.11) not only the state x from the original system is estimated, but also the unknown parameter β.

The example illustrates an approach to estimate parameters in dynamic systems. Note that the approach illustrated by the example applies equally well to nonlinear systems and the main difficulty is of course finding a stable state-observer for the extended state-space description.

Now, when it is clear how observers can be used to estimate constant parameters the following example will illustrate how such observers can be used to generate residuals where constant faults are decoupled.

Example 7.6 Reconsider Example 7.5 in a diagnosis application. Restate the model equations

$$\dot{x} = -\beta x + \gamma u$$
$$y = x$$

where β models a fault affecting system dynamics and γ a fault affecting the actuator. Assume that β and γ are constant parameters, both in the fault free case and in the faulty case. Assume three fault-modes:

$$\begin{aligned} \Theta &= \{[\beta\ \gamma];\ \beta \geq 0, \gamma \geq 0\} \\ \Theta_{NF} &= \{[1\ 1]\} \\ \Theta_{F1} &= \{[\beta\ 1];\ \beta \neq 1\} \\ \Theta_{F2} &= \{[1\ \gamma];\ \gamma \neq 1\} \end{aligned}$$

Suppose a diagnosis system, capable of separating the three fault-modes are to be designed and that the diagnosis system includes a hypothesis

test with the hypotheses

$$H^0 : \mathbf{F_p} \in \{\mathbf{NF}, \mathbf{F_1}\} \quad H^1 : \mathbf{F_p} \in \{\mathbf{F_2}\}$$

i.e. deviations in parameter β need to be decoupled in the test quantity for the hypothesis test. An example of such a test quantity, based on the methods described in this section, is to extend the state vector as in Example 7.5 resulting in an extended state-space description:

$$\dot{z} = \begin{pmatrix} -z_2 z_1 + \gamma u \\ 0 \end{pmatrix} = f(z, u) \tag{7.12a}$$

$$y = z_1 \tag{7.12b}$$

Assume that a K is found such that a nonlinear state-observer on the form

$$\dot{\hat{z}} = \begin{pmatrix} -\hat{z}_2 \hat{z}_1 + u \\ 0 \end{pmatrix} + K(y - \hat{y})$$
$$\hat{y} = \hat{z}_1$$
$$r = y - \hat{y}$$

is stable. Then the residual r goes to 0 when the process operates in fault mode **NF** or **F1**.

The main point of the example is that the nonlinear decoupling problem was first transformed into an estimation problem, as was done in e.g. Section 4.5, and the estimation problem was finally transformed into an observer design problem.

7.2.3 Decoupling of Sensor Faults

Fault isolation needs fault decoupling and in the previous section, decoupling of constant faults in general nonlinear systems was considered. Another important special case where fault decoupling is often possible is sensor faults. More precisely, a sensor fault is a fault that is modeled to *only* influence the measurement equation of one sensor and no other

parts of the model.

Example 7.7 Examples of sensor fault models are given by signals (or constants) f_1, f_2, and f_3 in the measurement equations:

$$y_1 = h_1(x, u) + f_1 \tag{7.13a}$$
$$y_2 = (1 - f_2)h_2(x, u) \tag{7.13b}$$
$$y_3 = h_3(x, u, f_3) \tag{7.13c}$$

Note that it is important that f_1, f_2, and f_3 does *not* occur elsewhere in the system model.

A sufficient condition for a sensor fault to be decoupled in a residual is that the sensor signal is not used when calculating the residual. In the observer framework, to decouple faults in sensor i, produce the residual by using the observer to estimate $\hat{y}_j$ not using sensor i. Then, calculate the residual as $r = y_j - \hat{y}_j$ where $i \neq j$.

Remark: If the fault is modeled as in (7.13a), it is a sufficient *and* necessary condition that the fault signal is not used in the calculation of the residual. Therefore, it is fault models of type (7.13a) that are decoupled with this methodology. But of course, since fault models (7.13b) and (7.13c) can be rewritten as (7.13a), i.e. the faults modeled by (7.13b) and (7.13c) are a subset of the faults modeled by (7.13a), these types of faults are also decoupled.

In this way, provided that it is possible to find a stable observer, decoupling of sensor faults in general nonlinear plants is achieved. If needed, the method is easily adopted to provide multiple fault decoupling by excluding more than one sensor from the observer feedback. The approach is illustrated by a small example.

Example 7.8 Assume a nonlinear, 2-output plant, where the two sensors will be monitored. The two sensor faults f_{s1} and f_{s2} are modeled as additive faults. A state-space description of the plant can be stated as

$$\begin{aligned} \dot{x} &= f(x, u) \\ \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} &= \begin{bmatrix} h_1(x, u) \\ h_2(x, u) \end{bmatrix} + \begin{bmatrix} f_{s1} \\ f_{s2} \end{bmatrix} \end{aligned}$$

Let an estimate $\hat{y}_1$ of y_1 be produced without using sensor 2 and the residual is calculated as $r_1 = y_1 - \hat{y}_1$. Then the residual will be sensitive

towards faults in sensor 1, f_{s1}, and insensitive towards faults in sensor 2, f_{s2}. This estimation can be performed via a nonlinear observer as

$$\begin{aligned}\dot{\hat{x}} &= f(\hat{x}, u) + K_1(y_1 - h_1(\hat{x}, u)) \\ r_1 &= y_1 - \hat{y}_1 = y_1 - h_1(\hat{x}, u)\end{aligned}$$

The matrix K_1 must be such that the observer is stable in the whole operating range. A similar observer estimating y_2 without using sensor 1 can be stated as

$$\begin{aligned}\dot{\hat{x}} &= f(\hat{x}, u) + K_2(y_2 - h_2(\hat{x}, u)) \\ r_2 &= y_2 - \hat{y}_2 = y_2 - h_2(\hat{x}, u)\end{aligned}$$

Provided we can find K_1 and K_2 that stabilizes the observers, we have shown how two residual generators can be designed for isolation of the sensor faults f_{s1} and f_{s2}.

7.2.4 Choice of observer feedback signals

As seen in the discussions above, when designing diagnostic observers there is quite a bit of freedom available for the designer when choosing which sensor signals to feed back into the observer. Motives for *not* feeding back a sensor signal might be that we wish to make the state estimation insensitive to faults in that particular sensor. However, note that this feedback selection must be done with some care, in particular it is necessary that, with the selected set of feedback signals, the system becomes observable. General non-linear observability conditions will not be discussed further here, but an intuitive sufficient condition to detect non-observable models will be discussed. For this, consider the model

$$\dot{x}_1 = f_1(x_1, u) \tag{7.14a}$$
$$\dot{x}_2 = f_2(x_1, x_2, u) \tag{7.14b}$$
$$y_1 = h_1(x_1, u) \tag{7.14c}$$
$$y_2 = h_2(x_2, u) \tag{7.14d}$$

From this one can see that it is not possible to use y_1 to estimate x_2 and use $r = y_2 - h_2(\hat{x}_2, u)$ as a residual. This is because there is no information about x_2 in the measurement signal y_1, i.e. the system is

not observable. This is illustrated in Figure 7.1. The arrows indicate how different variables influence each other and since there is no path from x_2 to y_1 there is no possibility to estimate x_2 from only y_1. Also

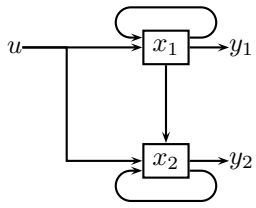


Figure 7.1: Schematic figure of the model (7.14).

note that the reverse is possible, i.e. to use y_2 to estimate x_1 and compute a residual as $r = y_1 - h_1(\hat{x}_1, u)$. Thus, it is generally a good idea to verify that there is a path from the state variables to the observer feedback signals. Note that this is only a necessary condition for observability. Consider the linear system

$$\begin{aligned} \dot{x} &= \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & 0 & 0 \\ a_{31} & 0 & 0 \end{bmatrix} x \\ y &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} x \end{aligned}$$

It is clear that there exists a path from all state variables to y but a quick analysis of the system reveals that the system is unobservable regardless of the values of parameters a_{ij}.

Chapter 8
Probabilistic Diagnosis

Chapter 3 described techniques for fault isolation in consistency based diagnosis systems where the main objective were to compute all diagnoses, according to Definition 2.1, given the observations and test results. Recall that an assignment $\mathcal{D}$ of behavioral modes to each component is a diagnosis if and only if the set of equations

$$\mathcal{D} \cup \mathcal{O} \cup \mathcal{M} \tag{8.1}$$

are *consistent*, where $\mathcal{O}$ are the observations and $\mathcal{M}$ the model. If the model is uncertain, for example if there is measurement noise, then there will never be consistency. Consider for example a situation where two sensors measure the same variable x and both sensors are subjected to Gaussian sensor noise ϵ_i as

$$y_1 = x + \epsilon_1$$
$$y_2 = x + \epsilon_2$$

Consistency is then naturally checked by computing the residual

$$r = y_1 - y_2$$

and check for zero. However, the residual is only exactly 0 if $\epsilon_1 = \epsilon_2$ and since this happens with probability 0 this is not a feasible approach.

In previous chapters we dealt with this situation by setting a suitable threshold such that inconsistency was indicated if the residual exceeded its threshold. The value of the threshold is based on, for example, a desired probability of false alarm or other performance specifications. A consequence of this approach to handling uncertainties is for example that the fault isolation procedure does not take into consideration whether a residual is far above its' corresponding threshold or only marginally above, the final decision will be the same. It is clear that the model and measurement uncertainty, inherent in practically any diagnosis problem, need to be handled with care. The above technique using thresholds is one way but another is to introduce probabilities in the diagnosis procedure and this is the topic of this chapter.

Making decision under uncertainty is an important topic in many disciplines, not only fault diagnosis, and a common way to approach this problem is to use probabilities. Then, instead of determining consistency of (8.1), an alternative would be to compute the probability of a candidate $\mathcal{D}$ given the observations $\mathcal{O}$ and the model $\mathcal{M}$

$$P(\mathcal{D}|\mathcal{O}, \mathcal{M}).$$

Since the number of candidates $\mathcal{D}$ is exponential in the number of components, an alternative could be the probability of the individual behavioral modes $\mathcal{B}_i$ as

$$P(\mathcal{B}_i|\mathcal{O}, \mathcal{M}). \tag{8.2}$$

where $\mathcal{B}_i$ may be a fault mode or the no-fault case for component i.

To illustrate this, consider a simple case with three residuals, three faults, and the corresponding decision structure

	F_1	F_2	F_3
r_1	0	X	X
r_2	X	0	X
r_3	X	X	0

(8.3)

where fault F_i corresponds to that component C_i has a malfunction. Say that residuals r_1 and r_2 exceed their thresholds and raise their corresponding alarms. In a consistency based approach this would mean that there are two conflicts

$$\pi_1 = OK(C_2) \wedge OK(C_3)$$
$$\pi_2 = OK(C_1) \wedge OK(C_3)$$

which corresponds to the minimal diagnoses

$$\mathcal{D}_1 = OK(C_1) \wedge OK(C_2) \wedge \neg OK(C_3),$$
$$\mathcal{D}_2 = \neg OK(C_1) \wedge \neg OK(C_2) \wedge OK(C_3)$$

Here, without any additional information, the single fault diagnosis $\mathcal{D}_1$ could be assumed more probable than the double fault diagnosis $\mathcal{D}_2$. On the other hand, with fictive numbers, the result from a probability based approach could look something like

$$P(\mathcal{B}_i|\mathcal{O}, \mathcal{M}) = \begin{cases} 0.01 & \text{if } \mathcal{B}_i = NF \\ 0.85 & \text{if } \mathcal{B}_i = F_1 \\ 0.93 & \text{if } \mathcal{B}_i = F_2 \\ 0.22 & \text{if } \mathcal{B}_i = F_3 \end{cases}$$

where the observations $\mathcal{O}$ could be results of diagnostic tests or measurements. Here it is clear that, with a high probability, there is a fault and that the double fault F_1 and F_2 is perhaps a prime candidate for a workshop engineer to investigate.

It is clear that obtaining such detailed, probabilistic, information about possible diagnoses is very attractive when making decisions in uncertain environments. But, as attractive the approach may seem, it is unfortunately often intractable since analytic expressions for these probability distributions are not available in non-linear and/or dynamic contexts. For example, it is generally not possible to analytically compute the probability distribution $p(x_t)$ of the state vector x_t in a non-linear difference equation

$$x_{t+1} = f(x_t) + \varepsilon_t$$

where ε_t is a random process. It is possible to use approximative techniques, e.g., Extended/Unscented Kalman Filters or Monte-Carlo methods, to obtain approximations of the probability distributions, but this may be computationally very heavy.

The main objective of this text is to introduce concepts for probability based diagnosis and also introduce some basic computation tools. In this text, only models with *discrete* random variables will be used which will allow us to use Bayesian networks where efficient techniques to perform the computations are available. This is not a

severe limitation since many of the probabilistic models built use only discrete variables, quite likely due to the lack of modeling tools and algorithms for the general case (Dìez and Druzdzel, 2006). In this text, focus is on principles and only a brief introduction to Bayesian networks will be included.

8.1 Introductory example

The presentation from here on will use the notation reviewed in Appendix 8.A. First, consider a situation with only two faults, f_1 and f_2, i.e., there are four possible system behavioral modes $\mathcal{B}_i$

$$FM \in \{NF, f_1, f_2, f_1 \& f_2\}.$$

Assume that faults f_1 and f_2 are equally probable and occurs independently. Now, consider a residual generator that is designed to detect fault f_1 but that the residual r is not only sensitive to fault f_1 but also to the second fault f_2. However, the residual is designed to detect f_1 in particular and is therefore significantly more sensitive to fault f_1 than f_2. An alarm is generated when the residual r exceeds a given threshold J. The probabilities that represent fault sensitivities in the residual and the a priori probabilities of faults are summarized in Table 8.1.

Table 8.1: A priori and sensitivity probabilities.

Probability	Formula	Value
False alarm	$P(r > J \mid FM = NF)$	0.01
A priori probability of fault i	$P(f_i),\ i = 1, 2$	0.02
Sensitivity to single fault f_1	$P(r > J \mid FM = f_1)$	0.99
Sensitivity to single fault f_2	$P(r > J \mid FM = f_2)$	0.30
Sensitivity to double fault $f_1 \& f_2$	$P(r > J \mid FM = f_1 \& f_2)$	0.99

Now, assume that the residual exceeds its threshold J. The conclusion in a deterministic, consistency based, setting as in Chapter 3 would then be that it is either fault f_1 or fault f_2 (or both) that caused the residual to exceed the threshold J. However, the information that both faults are equally probable and that the residual r is much more sensitive to fault f_1 than fault f_2 has not been taken into account. It is clear that, with this information it is reasonable to conclude that it is more probable that f_1 caused the alarm than fault f_2. Basic probability

theory allows us to compute exactly how much more probable it is that fault f_1 is the real fault than fault f_2.

Utilizing independence between the fault variables F_1 and F_2 are utilized above, the a priori probability for a fault mode can be written, for example, as

$$P(FM = f_1) = P(f_1, \neg f_2) = P(f_1)P(\neg f_2)$$

Using this fact and basic formulas for conditional probabilities we obtain the expressions

$$\begin{aligned}
P(FM = NF|r > J) &= \frac{P(r > J|FM = NF)P(FM = NF)}{P(r > J)} = \\
&= \frac{P(r > J|FM = NF)P(\neg f_1)P(\neg f_2)}{P(r > J)} \\
P(FM = f_1|r > J) &= \frac{P(r > J|FM = f_1)P(FM = f_1)}{P(r > J)} = \\
&= \frac{P(r > J|FM = f_1)P(f_1)P(\neg f_2)}{P(r > J)} \\
P(FM = f_2|r > J) &= \frac{P(r > J|FM = f_2)P(FM = f_2)}{P(r > J)} = \\
&= \frac{P(r > J|FM = f_2)P(\neg f_1)P(f_2)}{P(r > J)} \\
P(FM = f_1\&f_2|r > J) &= \frac{P(r > J|FM = f_1\&f_2)P(f_1)P(f_2)}{P(r > J)}
\end{aligned} \tag{8.4}$$

where $P(\neg f_i) = 1 - P(f_i)$. With the probability values from Table 8.1, the a posteriori probabilities for all behavioral modes can then be computed as

$$P(FM = f|r > J) = \begin{cases} 27.2\% & \text{if } f = NF \\ 55.0\% & \text{if } f = f_1 \\ 16.7\% & \text{if } f = f_2 \\ 1.1\% & \text{if } f = f_1\&f_2 \end{cases} \tag{8.5}$$

Note that it is not necessary to compute the denominator $P(r > J)$ since all 4 cases have the same denominator and that they all sum to 1. Thus, the probabilities can be determined by computing the

numerators, with numerical values from Table 8.1, and then normalize such that the probabilities sum to 1. Thus, it has been computed that the probability is 0.55 for single fault f_1 and 0.167 for single fault f_2.

One conclusion of this introductory example is that using only basic probability formulas it is direct to extend the consistency based diagnoses with a probability measure, i.e., associate each diagnosis with a probability. In the next section, we will introduce more general probabilistic models and also discuss computational techniques that make efficient computing with these models possible.

8.2 Probabilistic Models and Inference

This presentation will be limited to models with discrete random variables and static systems but much of the material presented can be extended in different ways to cover also continuous valued variables and dynamic models.

In general, a probabilistic model of a set of discrete random variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ is the probability mass function

$$P(x_1, \ldots, x_n) \tag{8.6}$$

which explicitly states probabilities for all possible values of the random variables $X_1, \ldots, X_n$. The probability mass function for all variables is sometimes referred to as the joint probability mass function. The variables can be intermediate process variables, observation variables, alarm signals, fault variables, and so on.

Example 8.1 In the introductory example there were three variables, the boolean fault variables F_1, F_2, and the alarm variable A that was true when the residual r exceeded its corresponding threshold. Table 8.1 does not give the full probability mass function in the form (8.6), but using the chain rule (8.16) we obtain

$$P(a, f_1, f_2) = P(a|f_1, f_2)\, P(f_1|f_2)\, P(f_2) = \\ = P(a|f_1, f_2)\, P(f_1)\, P(f_2).$$

The last equality is due to the independence between F_1 and F_2. Direct calculations gives the full probabilistic model in Table 8.2.

An important aspect of modeling is how to determine the model parameters, perhaps from measured data. When developing control

Table 8.2: Full probability mass function $P(a, f_1, f_2)$ for the initial example.

a	f_1	f_2	$P(a, f_1, f_2)$
False	False	False	0.9508
False	False	True	0.0137
False	True	False	0.0002
False	True	True	$4 \cdot 10^{-6}$
True	False	False	0.0096
True	False	True	0.0059
True	True	False	0.0194
True	True	True	0.0004

oriented physical models, techniques from system identification (Ljung, 1999) can be used to determine the model parameters. For probabilistic models this corresponds to determining all parameters in the probability mass function in (8.6). This important topic is outside the scope of this text and the interested reader is referred to existing literature, e.g., (Jensen and Nielsen, 2007).

8.2.1 Inference

Probabilistic inference is, in this setting, nothing else than the computation of posterior probabilities of certain events, given observations. Here, observations are specified values of variables in the probabilistic model. In a diagnosis application, this can typically be measurement variables and outcomes of diagnostic tests. For example, computing the probability of a fault given the observations from the process could correspond to

$$P(F_1|r_3 > J_3)$$

In general, computing the probability for $X = x$ given evidence/observations e, i.e. assignment of values to random variables, is given by direct use of formulas for conditional probabilities as

$$P(x|e) = \frac{P(x, e)}{P(e)} = \alpha\, P(x, e) \tag{8.7}$$

Here, again note that the denominator $P(e)$ does *not* depend on the value of x. This means that the numerator can be computed for *all*

values of x, and then the normalization factor α can be determined from

$$1 = \sum_x P(x|e) = \alpha \sum_x P(x, e)$$

Often, observations for all variables are not available since we do not have sensors at all positions in the process. Then $P(x, e)$ in (8.7) is not the full probability mass function (8.6) since there are additional variables than x and e in the model. Denote the additional variables with z, then marginalization gives that

$$P(x, e) = \sum_z P(x, e, z)$$

The inference expression (8.7) can then be written as

$$P(x|e) = \alpha \sum_z P(x, e, z) \tag{8.8}$$

where $P(x, e, z)$ is the full probabilistic model.

Example 8.2 Return to the introductory example. To compute the probability of a fault f_1 given an alarm, we get

$$\begin{aligned} P(f_1|a) = \alpha P(f_1, a) = \alpha \sum_{f_2} P(f_1, f_2, a) = \\ = \alpha(P(f_1, \neg f_2, a) + P(f_1, f_2, a)) = \\ = \alpha(0.0194 + 0.0004) = \alpha \cdot 0.0198 \end{aligned} \tag{8.9}$$

A corresponding calculation for $P(\neg f_1|a)$

$$P(\neg f_1|a) = \alpha \cdot 0.0155 \tag{8.10}$$

This gives that, where F_1 is the binary random variable for fault mode 1,

$$P(F_1|a) = \alpha\langle 0.0198, 0.0155\rangle = \langle 0.439, 0.561\rangle$$

where the normalization constant α is determined such that the probabilities (8.9) and (8.10) add up to 1. Note that the probability for fault f_1 is 0.561 which is not equal to the probability of fault mode $FM = f_1$ which was 0.55. This is natural since f_1 appears in both the single fault mode f_1 and the multiple fault mode $f_1\&f_2$.

8.2.2 Model and inference complexity

From the description above it is clear that, in principle, it is straightforward to perform any probabilistic inference if the full probability mass function is available by a direct utilization of (8.8). Unfortunately, such a naive approach is seldom applicable due to the inherent combinatorial explosion of the size of the probabilistic model. Consider the case with n binary variables in the model, then the $P(x_1, \ldots, x_n)$ has 2^n parameters. This exponential growth in parameters makes (8.8) unfit for inference for anything but very small models. Put in other words, a model with 40 variables is not a big model, but 2^{40} is a big number.

The introductory example was originally fully specified in Table 8.1 that consisted of 5 parameters and the full probabilistic model in Table 8.2 consisted of 8 parameters. This difference, although very small for this tiny example, indicates that it is possible to specify the probabilistic model with less than an exponential number of parameters. The key to this lies in utilization of independence among the model variables. In the introductory example the fault variables F_1 and F_2 were stated to be independent. Consider a model with n independent boolean variables, then the full probability mass function can be expressed using the chain rule as

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i)$$

This means that the 2^n number of parameters have been transformed into $2n$ parameters, i.e., exponential growth in size has been transformed into linear growth in size when utilizing the independence assumption. Full utilization of variable independence is a key to not only efficient model representation but also efficient inference. Introduction of observations may result in conditional independence, i.e., two variables that are not independent in the original model may become independent when observations are introduced. A small example below will illustrate the basic principle that can be used to make inference significantly more efficient.

Example 8.3 Consider a probabilistic model in three variables x_1, x_2, x_3. Assume that the probability mass function of x_1 is only dependent on x_1 and x_3, not x_2. Direct application of the chain-rule

then gives that

$$P(x_1, x_2, x_3) = P(x_1|x_2, x_3)P(x_2|x_3)P(x_3) = \\ = P(x_1|x_3)P(x_2|x_3)P(x_3)$$

where the first equality is a direct application of (8.16) and the second equality due to the independence assumption. The variables x_1 and x_2 are not independent since their corresponding probability mass functions are both dependent on the common variable x_3. However, assume that the value of x_3 is measured, then

$$P(x_1, x_2|x_3) = \frac{P(x_1, x_2, x_3)}{P(x_3)} = \frac{P(x_1|x_3)P(x_2|x_3)P(x_3)}{P(x_3)} = \\ = P(x_1|x_3)P(x_2|x_3).$$

In the above inference statement the probability mass function $P(x_1, x_2|x_3)$ could be factored into $P(x_1|x_3)$ and $P(x_2|x_3)$, i.e., the variables x_1 and x_2 were conditionally independent when observing x_3.

In a worst-case scenario, where there is no independence structure to be utilized, exact inference in probabilistic models is very expensive but often there exists structure that can be utilized. Next section will briefly introduce Bayesian networks, which makes it possible to efficiently represent probabilistic models and make inference.

8.3 Bayesian Networks

The concept of Bayesian networks was first introduced in (Pearl, 1985) and has received considerable attention since then. Bayesian networks are also part of the broader field of probabilistic graphical models (Edwards, 2000). One way to view a Bayesian network is as a representation of a probabilistic model (8.6) that utilizes the independence structure of the model. The description in this text will be rather brief, readers who are interested in more details are referred to any of the numerous books that are written on this topic. Examples of introductory texts are the review paper (Darwiche, 2010) or any of the books (Russell and Norvig, 2003) and (Jensen and Nielsen, 2007).

Before a formal definition is given, consider again the introductory example where the fault variables F_1 and F_2 were independent random

variables and the alarm variable A were dependent on both faults. This can be illustrated using a directed graph as in Figure 8.1. The nodes in the graph are variables and an arc from node X_i to node X_j represents that X_j is dependent on node X_i. To each variable in the graph,

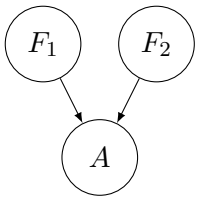


Figure 8.1: A graph representing the dependencies in the introductory example.

associate a probability table for the variable, given values of the parent variables. In this case, we have the probability tables for $P(f_1)$, $P(f_2)$, and $P(a|f_1, f_2)$. Now, utilizing the independence relations in Figure 8.1 and applying the chain-rule (8.16), with $(x_1, x_2, x_3) = (a, f_1, f_2)$, we get

$$P(a, f_1, f_2) = P(a|f_1, f_2)P(f_1|f_2)P(f_2) = P(a|f_1, f_2)P(f_1)P(f_2)$$

Here it is clear that the conditional probability tables associated with each node completely characterize the full probabilistic model. One can also say that the graph defines a factorization of the joint probability mass function. The above observation can be written in general as

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|x_1, \ldots, x_{i-1}) = \prod_{i=1}^{n} P(x_i|\text{parents}(x_i)) \quad (8.11)$$

where the function parents(X_i) returns all variables that are parents to variable X_i in the graph. For example, parents$(A) = \{F_1, F_2\}$ and parents$(F_1) = \emptyset$. To be able to make efficient inference of the models, it will be required that the graph does not contain any cycles. With this motivating discussion, a Bayesian network is defined as below.

Definition 8.1 (Bayesian network)**.** Let $\mathcal{X} = \{X_1, \ldots, X_n\}$ be a set of random variables with a finite set of values for each variable. A Bayesian network is then a pair $\mathcal{B} = \langle \mathcal{G}, \mathcal{P} \rangle$ where $\mathcal{G}$ is an acyclic

directed graph, defined on the nodes $\mathcal{X}$, and $\mathcal{P}$ a set of conditional probability tables, one for each node in the graph, defined as

$$P(x_i|\text{parents}(x_i)).$$

A Bayesian network $\mathcal{B}$ represents its corresponding joint probability mass function by expression (8.11).

Example 8.4 In the introductory example we had three variables, the fault variables F_1 and F_2, and the alarm variable A. The graph representing the dependencies is given in Figure 8.1. The independence between F_1 and F_2 is represented by the fact that there is no arc between the fault nodes and the alarm node is dependent on both variables nodes, which is represented by the two arcs.

The graph is clearly acyclic, and the conditional probability tables are given in Table 8.3. All variables are boolean, i.e., they can only have values *True* or *False*. Note that the values in the tables are taken directly from Table 8.1.

Table 8.3: Conditional probability tables for variables in the Bayesian Network in Figure 8.1.

(a)

F_i	$P(F_i)$
True	0.02
False	0.98

(b)

F_1	F_2	$P(A\|F_1, F_2)$	$P(\neg A\|F_1, F_2)$
False	False	0.01	0.99
False	True	0.30	0.70
True	False	0.99	0.01
True	True	0.99	0.01

One should note that there is not a unique Bayesian network for a given probabilistic model. Looking at the chain-rule (8.16), different variable orderings give different factorizations of the joint probability mass function. For example, the introductory example could be represented by the dependency graph in Figure 8.2 which corresponds to the factorization

$$P(a, f_1, f_2) = P(f_1)P(a|f_1)P(f_2|a, f_1)$$

This is possible, but the individual factors $P(f_1)$, $P(a|f_1)$, and $P(f_2|a, f_1)$ are perhaps less intuitive than those for the network in Figure 8.1.

Example 8.5 All previously mentioned Bayesian networks have been

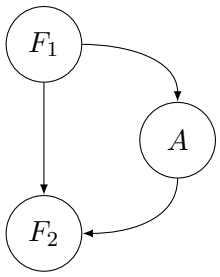


Figure 8.2: Alternative Bayesian network graph for the introductory example.

small enough to do the computations using the full joint probability mass function. To illustrate what industrial sized models can look like, consider the Bayesian model of the XPI fuel injection system in Scania trucks. The model is developed at Scania in (Cyon, 2012).

The dependency graph shown in Figure 8.3 consists of 141 nodes, where about 40 corresponds to components that should be monitored and the rest of the nodes correspond to observation nodes and internal variables. Many of the variables has 2 possible values, for example the residuals has *No Alarm* and *Alarm* as values, but some components has as many as 8 possible values and as an example, an inlet metering valve has the possible values *Fuel leak*, *Electrical fault*, *Stuck or clogged*, *Wrong pressure*, *Emission fault*, *Corrosion or cavitation*, *Air leak*, and *No Fault*. The complete joint probability mass function has in the order of 10^{50} values, which makes it clear that inference or representation of the full probability mass function is intractable.

However, using available techniques for inference it can still be possible to do the computations. For example, assume 4 random residuals, $r_1, \ldots, r_4$ in the system indicates an alarm. Computing the full conditional probability, using exact techniques,

$$P(x_1, \ldots, x_{141} | r_1, r_2, r_3, r_4)$$

took less than a second on a standard laptop (model year 2012) using the freely available software (GeNIe, 2012).

One should note that worst-case complexity of inference is high and in a model of this size, there is a good chance that there are cases where inference computations are not feasible with respect to memory and/or time. Then, instead of exact solutions, approximative approaches can

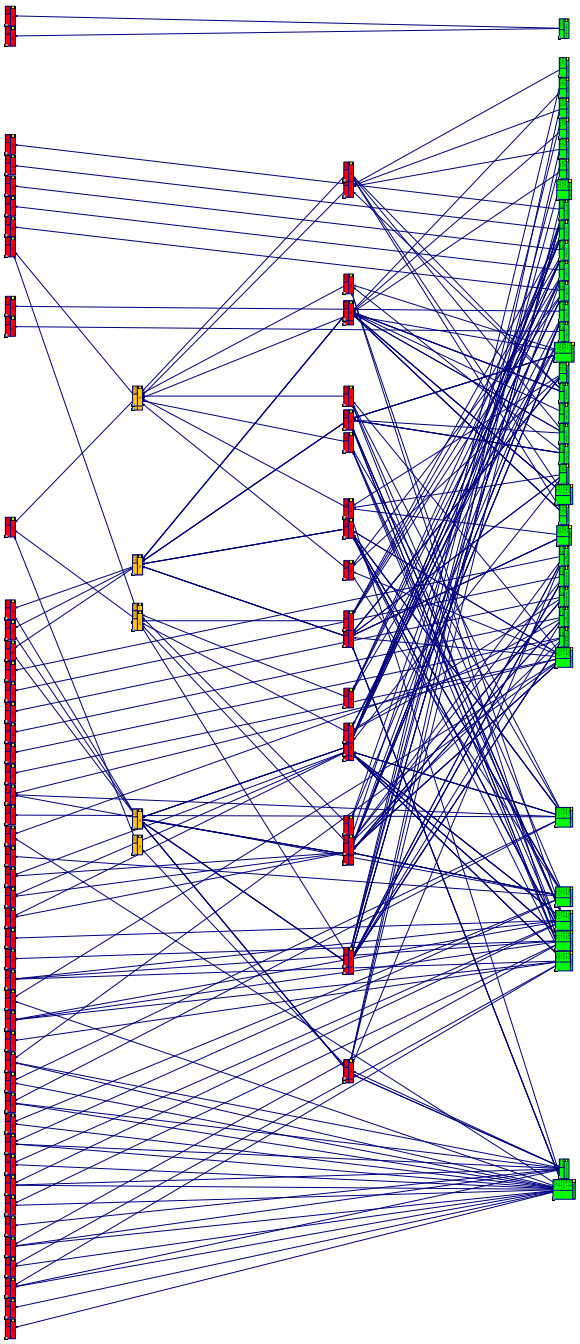


Figure 8.3: A screenshot of a model of a Scania XPI, fuel injection system, implemented in (GeNIe, 2012) by (Cyon, 2012) (used with permission from Scania).

be utilized as is briefly discussed in Section 8.3.2.

8.3.1 Canonical Models

A key problem when specifying the probability tables in a Bayesian network is that the size of the table is exponential in the number of parents. This means that even for moderate sized models, a great number of parameters need to be determined. Roughly, the probability values are either determined using expert knowledge or using experimental data. In either case, the problem starts being difficult for nodes with many parents and either the expert is unable to specify all the required values or there will be a need for huge amount of representative data to reliably infer the probabilities from data.

One way to handle the problem is to utilize what is referred to as canonical models, where the number of parameters needed to describe a high dimensional probability mass function is dramatically reduced. The model in Figure 8.3 makes extensive use of one particularly common canonical model, the leaky-or node.

Canonical models will be introduced, influenced by the presentation in (Dìez and Druzdzel, 2006), by first considering the case where a variable y is a deterministic function of the nodes in x, i.e., $y = f(x)$. The probability table representing a deterministic function has 0 parameters, regardless of the size of x and y, and is given by

$$P(y|x) = \begin{cases} 1 & y = f(x) \\ 0 & \text{otherwise} \end{cases} \tag{8.12}$$

For example, a probability table representing the or-function

$$y = x_1 \vee x_2$$

is

x_1	x_2	$P(y)$	$P(\neg y)$
false	false	0	1
false	true	1	0
true	false	1	0
true	true	1	0

Although deterministic functions can be very useful in a probabilistic model, the main motivation for introducing probabilities was to take

uncertainty into consideration and then deterministic functions only is not enough. For example, consider residual r_3 in (8.3) where the residual generates an alarm in case of fault F_1 or F_2. However, perfect performance is not generally realistic and a certain amount of false alarms and missed detections can not be avoided, and has to be included in the model.

To still utilize the idea from the deterministic model above, noisy models are introduced where intermediate variables Z_i are included according to Figure 8.4. The noisy model thus consists of two parts, one deterministic part, that has 0 parameters, and one noisy part that connects the variables X_i with the intermediate variables Z_i. Since the

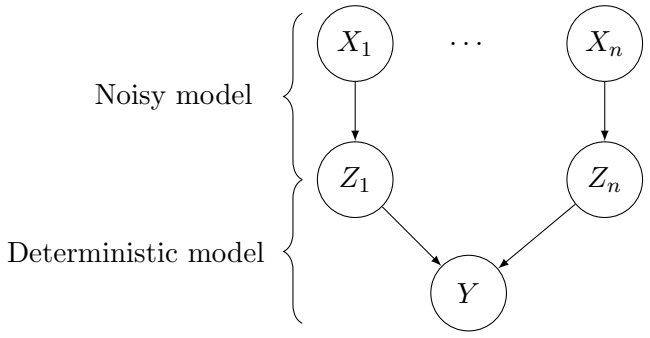


Figure 8.4: Structure of a noisy model.

X_i variables are assumed independent in Figure 8.4, the number of parameters in the Z_i-nodes are relatively small. For example, assume all variables are binary, then the probability mass function $P(y|x_1, \ldots, x_n)$ has 2^n parameters. With the noisy model in Figure 8.4, each Z_i node has 2 parameters, and the Y node 0 parameters. This means that the noisy model makes it possible to obtain linear, instead of exponential, growth in the number of parameters.

A common noisy model is the noisy-or, where Y is a deterministic or-function of the Z_i variables and the probability table for Z_i is given by

x_i	$P(z_i\|x_i)$	$P(\neg z_i\|x_i)$
False	0	1
True	c_1	$1 - c_1$

This table means that if X_i is false, so will Z_i but if X_i is true there is a probability $1 - c_1$ that Z_i will be false. With $c_i = 1$ we have the

deterministic or-function and with decreasing c_i there is an increasing uncertainty that Z_i equals X_i.

Example 8.6 Consider residual r_3 in the decision structure (8.3). If we assume that the faults F_1 and F_2 influence the residual independently, and that the probability for detection of faults F_1 and F_2 are 0.9 and 0.6 respectively. Then the probability table for the alarm variable, i.e., $P(\text{alarm}|f_1, f_2)$ can be modeled with a noisy-or with $c_1 = 0.9$ and $c_2 = 0.8$.

The noisy-or introduced above allowed us to introduce probability of detection and probability of missed detection in the model, but there is no possibility to model false alarms. This leads to so called leaky models shown in Figure 8.5, which is similar to noisy models but with the addition of a leak-node Z_l. Since the leaky-node appears in the deterministic part of the model, only the a-priori probabilities for the Z_l variable are added to the parameters to be determined.

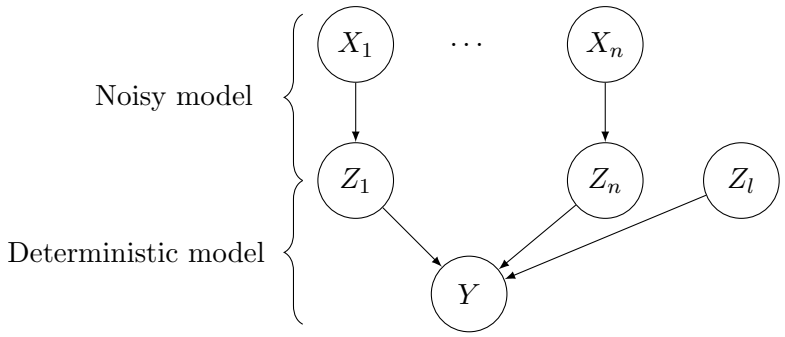


Figure 8.5: Structure of a leaky model.

Example 8.7 Return to Example 8.6 but with the addition of a false alarm probability of 0.01. This situation can be modeled using a leaky-or canonical model where the leak-node represents causes of false alarms. Thus, a leaky-or with the same parameters as in Example 8.6 and the probability table

$$P(Z_l) = \langle 0.99, 0.01 \rangle$$

for the leak node introduces also probability for false alarm in the model. Thus, this canonical model is well suited to model detection

probabilities and false alarms in a residual with only a few and easily interpreted parameters. The assumption that makes this possible is that faults and false alarm events are independent events, which may or may not be a reasonable assumption. This assumption is something that needs to be considered when using noisy or leaky nodes.

The canonical models in Figures 8.4 and 8.5 show general ways to compose noisy- and leaky-models that make reduction of the number of parameters needed to describe the probability mass function. Above, perhaps the most common canonical models, the noisy and leaky or-function was shown for binary variables. There is a straightforward generalization of the or-function for non-binary variables to the max-function and this canonical model is typically available in Bayesian modeling tools, for example in (GeNIe, 2012). Note also that noisy and leaky models trivially extend to any other deterministic function like and/or/not or even any general function $f(x)$ as in (8.12). One should note that it is common, for example in (GeNIe, 2012), to use the terminology noisy model when referring to leaky-models.

8.3.2 Inference in Bayesian networks

It is outside the scope of this text to give detailed descriptions of inference methods for Bayesian networks. The objective here is only to indicate ways to make inference more efficient and give references to works that describe algorithms in more detail.

To illustrate, consider the Bayesian network graph in Figure 8.6 and assume we want to compute

$$P(x_1|x_4, x_5)$$

Direct use of (8.8), using the full joint probability mass function, we obtain

$$P(x_1|x_4, x_5) = \alpha \sum_{x_2} \sum_{x_3} P(x_1, x_2, x_3, x_4, x_5) \tag{8.13}$$

As indicated by Example 8.5, this approach is not always feasible due to the size of the probability mass function. However, the factorization obtained from the Bayesian network dependency graph can be utilized. The joint probability mass function can be factored and the factors

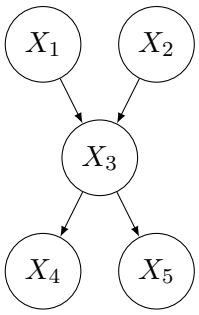


Figure 8.6: Example Bayesian network.

$P(x_1)$ and $P(x_2)$ can be factored out of summations according to

$$\begin{aligned} P(x_1|x_4,x_5) &= \alpha \sum_{x_2} \sum_{x_3} P(x_1,x_2,x_3,x_4,x_5) = \\ &\alpha \sum_{x_2} \sum_{x_3} P(x_1)P(x_2)P(x_3|x_1,x_2)P(x_4|x_3)P(x_5|x_3) = \\ &\alpha\, P(x_1) \sum_{x_2} P(x_2) \sum_{x_3} P(x_3|x_1,x_2)\underline{P(x_4|x_3)P(x_5|x_3)} \end{aligned} \quad (8.14)$$

Evaluating (8.14) instead of (8.13) simplifies the computations, but note that the underlined factors are independent of variable x_2. This means that this factor will be computed twice, to the same value, for each value of x_2 in the outer summation. This indicates further possibilities and avoiding such unnecessary computations makes little difference in this small example, but can make a significant difference for larger models.

Algorithms for inference in Bayesian networks can be categorized into either exact algorithms or approximate algorithms (Darwiche, 2010). In the exact algorithms, possibilities like the one illustrated above are utilized in techniques based on variable elimination (Dechter, 1996; Lauritzen and Spiegelhalter, 1998; Zhang and Poole, 1994). However, no matter what kind of technique that is employed, exact inference is NP-hard (Russell and Norvig, 2003) which means that in a worst case situation no efficient algorithm exists. But there are classes of networks which are easier than others. For example, networks where there is only one directed path between any two nodes, like the one in Figure 8.6, is called singly connected trees or polytrees. It can be

shown that if the network graph is a polytree, inference complexity is linear in the size of the tree.

The variable elimination method can be utilized to compute probabilities for a single variable in a Bayesian network. Often it is of interest to compute the probabilities for a set of variables, perhaps all variables, and then so called join-tree algorithms (Russell and Norvig, 2003) can be utilized. The basic idea of join-tree algorithms is to join nodes in the network graph to transform the Bayesian network into an equivalent Bayesian network that has a polytree structure. Then, size of the model representation is traded for inference efficiency. However, there are cases where the model is too complex for exact inference, and then one possibility is to resort to approximative inference methods, for example (Darwiche, 2010; Koller and Friedman, 2009; Pearl, 1988; Yedidia et al., 2005).

Appendix

8.A Notation

This section will briefly review the notation that will be used together with some fundamental results from basic probability theory that will be used. Note that only discrete random variables are considered. First, the probability that a random variable X, generally written in upper case, has the value x_i, written in lower case, is written as

$$P(X = x_i) \text{ or shorter } P(x_i).$$

For most of this presentation, when it is clear from context which random variable that is considered, the shorter form will be used. If the random variable X is boolean, i.e., can only have the value True or False, the short forms

$$P(x) \text{ and } P(\neg x)$$

will be used to denote

$$P(X = \text{True}) \text{ and } P(X = \text{False}).$$

Sometimes we want to talk about the probabilities for all the possible values of a random variable. Then $P(X)$ denotes a *vector* of values for all the probabilities. So, instead of summarizing the probabilities as in (8.5), we can write

$$P(FM|r > J) = \langle 0.27, 0.55, 0.17, 0.01 \rangle$$

Here we introduce the $\langle\rangle$-notation, where each element correspond to the probability of a certain value of the random variable. A fundamental object is the probability mass function (pmf) which is a function that gives the probability that a discrete random variable is exactly equal to some value. A probability mass function differs from a probability density function in that the latter is associated with continuous rather than discrete random variables.

A fundamental operation when computing probabilities is marginalization. For example, if we have two random variables X and Y and the corresponding probability mass function $P(x, y)$ and want to compute

the probability that Y has the value y, and then marginalization gives that

$$P(y) = \sum_x P(x, y) \tag{8.15}$$

where the summation is over all possible values of X. Another useful property is that a probability mass function can be factored using conditional probabilities using the chain rule as

$$P(x_1, \ldots, x_n) = \prod_i P(x_i | x_1, \ldots, x_{i-1}) \tag{8.16}$$

and this is particularly useful when finding efficient representations of the probability mass function. If two random variables X and Y are independent, then

$$P(x|y) = P(x).$$

A fundamental operation is to to update the belief, i.e., probability mass function of a random variable when new information arrives. To incorporate new information, sometimes referred to as evidence or simply observations, one can utilize expressions for conditional probabilities, or Bayes' rule, as

$$P(x|y) = \frac{P(x, y)}{P(y)} = \frac{P(y|x)P(x)}{P(y)}.$$

In this formula, the probability $P(x)$ is referred to as the prior and $P(x|y)$ as the posterior. The interpretation of the expression is how the prior knowledge $P(x)$ of the random variable X changes when we obtain the evidence that variable Y equals value y.

Bibliography

M. Basseville and I.V. Nikiforov. *Detection of Abrupt Changes*. PTR Prentice-Hall, Inc, 1993. ISBN 0-13-126780-9. URL: http://www.irisa.fr/sisthem/kniga/.

R.V. Beard. *Failure Accommodation in Linear System Through Self Reorganization*. PhD thesis, MIT, 1971.

James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer, 1985.

B. Bergman and B. Klefsjö. *Tillförlitlighet*. 1996.

G. Casella and R.L. Berger. *Statistical Inference*. Duxbury Press, 1990.

J. Chen and R. J Patton. *Robust Model-Based Fault Diagnosis for Dynamic Systems*. Kluwer Academic Publishers, 1999. ISBN 0-7923-8411-3.

E.Y. Chow and A.S. Willsky. Analytical redundancy and the design of robust failure detection systems. *IEEE Trans. on Automatic Control*, 29(7):603–614, 1984.

R.N. Clark. The dedicated observer approach to instrument fault detection. In *Proc. of the 15th CDC*, pages 237–241, 1979.

D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms - An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer Verlag, second edition, 1996.

A. Cyon. Modeling of fuel injection system for troubleshooting. Master's thesis, Royal Institute of Technology, KTH, Sweden, 2012.

A. Darwiche. Bayesian networks. *Communications of the ACM Magazine*, 53(12):80–90, 2010.

J. de Kleer, A.K. Mackworth, and R. Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56(2-3):197–222, 1992.

R. Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pages 211–219, 1996.

S.X. Ding, E.L. Ding, and T. Jeinsch. An approach to analysis and design of observer and parity relation based fdi systems. In *Proc. IFAC World Congress*, volume P, Beijing, P.R China, 1999.

X. Ding and P.M. Frank. Frequency domain approach and threshold selector for robust model-based fault detection and isolation. In *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pages 271–276, Baden-Baden, Germany, 1991.

F. J. Dìez and M. J. Druzdzel. Canonical probabilistic models for knowledge engineering. Technical Report CISIAD-06-01, UNED, Madrid, Spain, 2006.

D. Edwards. *Introduction to Graphical Modelling*. Springer, 2nd edition, 2000.

E. Frisk. *Residual Generation for Fault Diagnosis*. PhD thesis, Linköping University, November 2001.

E. Frisk and L. Nielsen. Robust residual generation for diagnosis including a reference model for residual behavior. IFAC, 1999.

GeNIe, 2012. GeNIe and SMILE v2.0, 2012. http://genie.sis.pitt.edu/.

J. Gertler. Analytical redundancy methods in fault detection and isolation; survey and synthesis. In *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pages 9–21, Baden-Baden, Germany, 1991.

J. Gertler. *Fault Detection and Diagnosis in Engineering Systems*. Marcel Dekker, 1998.

J. Gertler and D. Singer. A new structural framework for parity equation-based failure detecation and isolation. *Automatica*, 26(2): 381–388, 1990.

T. Glad and L. Ljung. *Reglerteori - Flervariabla och olinjära metoder*. Studentlitteratur, 1997.

N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *Proc. IEE-F*, 140 (2):107–113, 1993.

F. Gustafsson. *Adaptive filtering and change detection*. John Wiley & Sons, 2000. ISBN 0471 49287 6.

W. Hamscher, L. Console, and J. de Kleer. *Readings in MODEL-BASED DIAGNOSIS*. Morgan Kaufmann Publishers, 1992. ISBN 1-55860-249-6.

J. B. Heywood. *Internal Combustion Engine Fundamentals*. McGraw-Hill series in mechanical engineering. McGraw-Hill, 1992. ISBN 0-07-100499-8.

E. Hubert. Differential algebra for derivations with nontrivial commutation rules. *Journal of Pure and Applied Algebra*, 200(1-2):163–190, 2005.

E. Hubert. The diffalg package. This is an electronic document, http://www-sop.inria.fr/cafe/Evelyne.Hubert/diffalg/, 2008. Date retrieved: March 13, 2008.

T. Höfling and R. Isermann. Fault detection based on adaptive parity equations and single-parameter tracking. *Control Eng. Practice*, 4 (10):1361–1369, 1996.

R. Isermann. Process fault detection on modeling and estimation methods - a survey. *Automatica*, 20(4):387–404, 1984.

R. Isermann. Fault diagnosis of machines via parameter estimation and knowledge processing - tutorial paper. *Automatica*, 29(4):815–835, 1993.

F. V. Jensen and T. D. Nielsen. *Bayesian Networks and Decision Graphs*. Springer Verlag, second edition, 2007.

T. Kailath. *Linear Systems*. Prentice-Hall, 1980. ISBN 0-13-536961-4.

Thomas Kailath, Ali H Sayed, and Babak Hassibi. Linear estimation. 2000.

S.M. Kay. *Fundamentals of Statistical Signal Processing: Detection Theory*. Prentice Hall, 1998.

M. Kinnaert. Robust fault detection based on observers for bilinear systems. *Automatica*, 35:1829–1842, 1999.

J. De Kleer and B.C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.

J. De Kleer and B.C. Williams. Diagnosis with behavioral modes. In *Proceedings IJCAI-89*, pages 104–109, Detroit, MI, 1989.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

S.L. Lauritzen and D.J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of Royal Statistics Society*, pages 157–224, 1998.

E. L Lehmann. *Testing Statistical Hypotheses*. Springer Verlag, second edition, 1986.

P. Li and V. Kadirkamanathan. Particle filtering based likelihood ratio approach to fault diagnosis in nonlinear stochastic systems. *IEEE Trans. on Systems, Man, and Cybernetics*, 31(3):337–343, 2001.

L. Ljung. *System Identification: Theory for the User*. Prentice Hall, 2nd edition, 1999.

D. Luenberger. *Linear and Nonlinear Programming.* Addison Wesley, 1989. ISBN 0-201-15794-2.

J.M. Maciejowski. *Multivariable Feedback Design.* Addison Wesley, 1989.

R.S. Mangoubi. *Robust Estimation and Failure Detection, A Concise Treatment.* Springer Verlag, 1998.

M.A. Massoumnia, G.C. Verghese, and A.S. Willsky. Failure detection and identification. *IEEE Trans. on Automatic Control*, AC-34(3): 316–321, 1989.

S.E. Mattson, H. Elmqvist, and M. Otter. Physical system modeling with modelica. *Control Engineering Practice*, 6(4):501–510, 1998.

R.E. McDermott, M.R. Beauregard, and R.J. Mikulak. *The Basics of FMEA.* 1996.

E.A. Misawa and J.K. Hedrick. Nonlinear observers – a state-of-the-art survey. *IEEE Trans. of ASME*, 111:344–352, September 1989.

P.E. Moral and J.W. Grizzle. Observer design for nonlinear systems with discrete-time measurement. *IEEE Trans. on Automatic Control*, 40(3):395–404, 1995.

H.G. Natke and C. Cempel. *Model-Aided Diagnosis of Mechanical Systems - Fundamentals, Detection, Localization, Assessment.* Springer Verlag, 1997.

H. Nijmeijer and T.I. Fossen. *New Directions in Non-linear Observer Design.* Springer Verlag, 1999.

R. Nikoukhah. A new methodology for observer design and implementation. Technical Report 2677, INRIA, 1995.

R. Nikoukhah. A new methodology for observer design and implementation. *IEEE Trans. on Automatic Control*, 43(2):229–234, 1998.

M. Nyberg. Automatic design of diagnosis systems with application to an automotive engine. *Control Engineering Practice*, 7(8):993–1005, 1999.

M. Nyberg and E. Frisk. Residual generation for fault diagnosis of systems described by linear differential-algebraic equations. *IEEE Transactions on Automatic Control*, 51(12):1995–2000, 2006.

E.S. Page. Continuous inspection schemes. *Biometrika*, 41:100–115, 1954.

P. Palady. *Failure Modes & Effects Analysis*. 1995.

R. Patton, P. Frank, and R. Clark, editors. *Fault diagnosis in Dynamic systems*. Systems and Control Engineering. Prentice Hall, 1989.

R.J. Patton. Robust model-based fault diagnosis:the state of the art. In *IFAC Fault Detection, Supervision and Safety for Technical Processes*, pages 1–24, Espoo, Finland, 1994.

R.J. Patton, P.M. Frank, and R.N. Clark. *Issues of Fault Diagnosis for Dynamic Systems*. Springer Verlag, 2000.

J. Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proceedings of the Cognitive Science Society*, pages 329–334, 1985.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

C. De Persis and A. Isidori. On the observability codistributions of a nonlinear system. *System and Control Letters*, 40:875–880, 2000.

C. De Persis and A. Isidori. A geometric approach to nonlinear fault detection and isolation. *IEEE Trans. on Automatic Control*, 46(6): 853–865, 2001.

B. Jones P.H.Garthwaite, I.T. Jolliffe. *Statistical Interference*. Prentice Hall, 1995. ISBN 0-13-847260-2.

J.W. Polderman and J.C. Willems. *Introduction to Mathematical Systems Theory: A Behavioral Approach*. New York: Springer Verlag, 1997.

R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, April 1987.

J. Ritt. *Differential Algebra*. Dover Publications, 1950.

N.H. Roberts. *Fault Tree Handbook*. 1992.

E.L. Russell and L.H. Chiang R.D. Braatz. *Data-driven Methods for Fault Detection and Diagnosis in Chemical Processes*. Springer Verlag, 2000.

S. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach*. Prentice Hall, second edition, 2003.

M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, September 1995.

J.J. Slotine, J.K. Hedrick, and E.A. Misawa. On sliding observers for nonlinear systems. *Journal of Dynamic Systems, Measurement, and Control*, 109:245–252, September 1987.

B. Sohlberg. *Supervision and control for industrial processes : using grey box models, predictive control and fault detection methods*. Springer Verlag, 1998.

B.L. Walcott, M.J. Corless, and S.H. Żak. Comparative study of non-linear state-observation techniques. *Int. Journal of Control*, 45(6): 2109–2132, 1987.

J. Yedidia, W. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized beleif propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

N.L. Zhang and D. Poole. A simple approach to bayesian network computations. In *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, pages 171–178, 1994.

Index

– **W** –