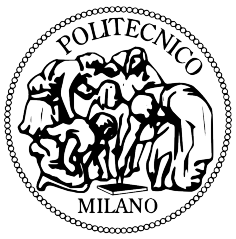


Politecnico di Milano

DIPARTIMENTO DI ELETTRONICA E INFORMAZIONE
Corso di Laurea Magistrale in Ingegneria dell'Automazione

Tesi di Laurea Magistrale

Modeling and Simulation for Predictive Maintenance: a survey

Candidato:
Davide Negretti
Matricola 824262

Relatore:
Ch.mo Prof. Gianni Ferretti

Correlatore:
Prof. Alberto Leva

Anno Accademico 2018-2019

Contents

Abstract

The aim of Predictive Maintenance is to detect a fault before it leads to a system failure. While some kinds of fault are sudden, some others (such as the wear of a mechanical component) are incremental, and they may affect its nominal behavior before the failure occurs.

If the nominal behavior of the system were known, it would be possible to detect the changes caused by faults, however disturbances and other factors also affect its behavior. A mathematical model of the system can be useful to analyze its behavior in both healthy and faulty conditions, and to understand whether a change in its behavior has been caused by a fault or by other reasons.

This research shows many possible uses for the models, analyzing their advantages and disadvantages.

Notation

Notation	Meaning (unless differently specified)
$\boldsymbol{v}$, $\boldsymbol{v}^\mathsf{T}$	column or row vector
$\boldsymbol{v}_j$	sub-vector of $\boldsymbol{v}$
v_i	i^{th} element of a (column or row) vector
$\boldsymbol{v_j}$, $\boldsymbol{v_j}^\mathsf{T}$	numbered column or row vector
$(\boldsymbol{v_j})_i$ or $\boldsymbol{v}_{\boldsymbol{j}i}$	i^{th} element of a numbered (column or row) vector $\boldsymbol{v_j}$
M or (m_{ij})	matrix
$M_{r\times c}$	matrix with r rows and c columns
M^T	transposed matrix
$M_{i,*}$ or M_i	i^{th} row of a matrix M
$M_{*,j}$ or $\boldsymbol{m_j}$	j^{th} column of a matrix M (column vector)
m_{ij} , $(\boldsymbol{m_j})_i$, $\boldsymbol{m}_{\boldsymbol{j}i}$	element at the i^{th} row and j^{th} column of a matrix M
I_n , I	identity matrix of size n, or appropriate size
$[0]_n$, $[0]_{n\times m}$, $[0]$	null matrix of size $n \times n$, $n \times m$, or appropriate size
$\mathbf{0}_n$, $\mathbf{0}$	null (column) vector of size n or appropriate size
$F(s)$	continuous-time transfer function (or transfer matrix)
$F(z)$	discrete-time transfer function (or transfer matrix)
$F_{ij}(s)$, $F_{ij}(z)$	continuous- or discrete-time transfer function from j^{th} input to i^{th} output, i.e. $y_i(t) = F_{ij}(s)\, u_j(t)$
$x(t)$, $x(k)$	continuous- and discrete-time signal (mono- or multi-dimensional)
$\boldsymbol{v}(t)$, $\boldsymbol{v}(k)$	time-dependent column vector
$\mathcal{A} = \{a_1, a_2, ...\}$	set (collection of elements)
$x \triangleq expr.$	definition (mathematical relation)
$x \coloneqq expr.$	assignment (numerical value)

Part I

Introduction

Chapter 1

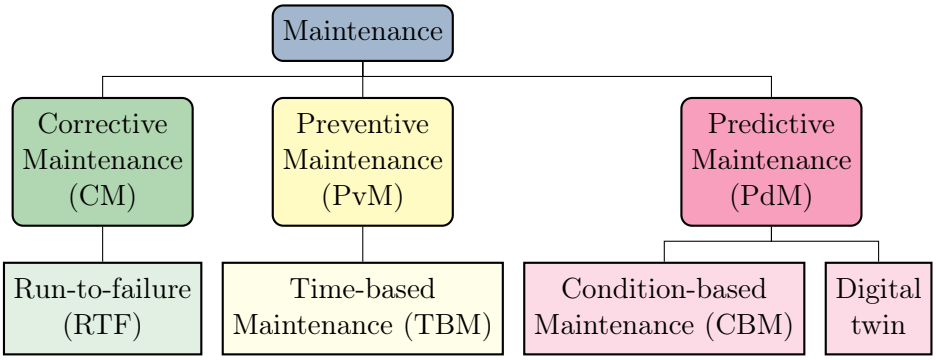
Introduction to Maintenance

According to the Oxford Dictionary, maintenance is *the action of keeping something in working order*. Maintenance of a technical system (mechanical, electrical, ...) aims to keep the system in an operational condition, or restore the operational condition after a failure, and can be carried out according to different strategies:

i **Corrective Maintenance (CM)** consists in identifying a fault after it has occurred (*"Run-to-failure"*), and restoring the system to a working operational condition.

ii **Preventive Maintenance (PvM)** consists in *scheduling* maintenance operations before the failure is expected to occur, according to theoretical, statistical or empirical considerations, such as the expected lifetime of a component (*"Time-based Maintenance"*).

iii **Predictive Maintenance (PdM)** consists in a real-time monitoring of the system aimed to estimate the health status of a component (*"Condition-based Maintenance"*) and detect an imminent failure of the system, in order to perform maintenance operations before the failure occurs.



1.1 Maintenance strategies

1.1.1 Corrective Maintenance

Corrective maintenance is the earliest and, apparently, the simplest maintenance strategy: equipment is allowed to run until a functional failure occurs (*"Run-to-failure"*, or RTF), then the damaged component has to be identified and replaced.

RTF approach has the advantage that it does not require maintenance planning. On the other hand, equipment becomes unavailable until the fault has been identified and repaired, and costs due to unplanned downtime may be higher than the ones associated to maintenance planning. Moreover, secondary damages may be observed in the system as consequence of the primary failure.

1.1.2 Preventive Maintenance

Unlike corrective maintenance, *preventive maintenance* is a strategy whose intent is avoiding failures: equipment undergoes periodical maintenance actions, according to a planned schedule (either with a *"time-based"* or *"experience-based"* approach), before it is expected to break down.

With this strategy, equipment failures are usually avoided, and planned maintenance actions can often be carried out while the equipment is still working (otherwise, a planned downtime still has a minor impact than an unplanned one), but unnecessary corrective actions are often performed.

1.1.3 Predictive Maintenance

Predictive maintenance can be considered as the evolution of preventive maintenance: the health status of the equipment is continuously monitored, and maintenance actions are planned only when failure is believed to be imminent (*"condition-based"* maintenance). This approach requires continual measurements on the plant, together with a deeper understanding of how the system works, but it allows to reduce unnecessary corrective actions.

Degradation models Nowadays CBM techniques make use of *degradation-based* measures, which are sensory informations that can be used to estimate the health of a piece of equipment, i.e. its *Remaining Useful Life* (RUL). Such measures, which usually concern vibrations, temperatures or sounds) often exhibit characteristic patterns, known as *degradation signals*, which can be used to estimate the residual life of those components which are subject to wear.

The Digital Twin A new approach in PdM consists in running a simulation of the system in parallel with the system itself. If the simulation results start to diverge from the measurements on the actual system, a failure may be imminent and the cause of the divergence should be investigated.

1.2 Main aspects of predictive maintenance

Faults and failures A failure is an event that ceases the ability of an entity (a piece of equipment) to perform specific functions. A fault is the inability of an entity to perform a specific function. Failure can be associated to the concept of event, and fault can be associated to the concept of state in which the entity can be found due to a failure. Corrective Maintenance deals with *fault detection (identification)* and *fault diagnosis (isolation)*, and Predictive Maintenance deals with *fault prediction*.

Process monitoring *Process monitoring* (PM) is the act of monitoring a system in order to identify significant changes which may be indicative of a fault or an imminent fault. PM requires to distinguish between *common-cause* variations, i.e. the natural variability of the process, and *special-cause* variations, i.e. malfunctions of the system. A process with special-cause variations is said to be *out of control*.

Condition monitoring In CBM, *Condition monitoring* (CM) is the act of monitoring a *parameter of condition* in the equipment (*degradation-based measure*), in order to identify significant changes which may be indicative of a developing fault. Condition monitoring techniques include:

- vibration monitoring
- acoustic emissions monitoring
- infrared thermography inspection
- lubricant analysis
- electrical insulation analysis

Redundancy Redundancy allows to improve the reliability of a system.

Physical redundancy consists of equipping the system with redundant physical devices, like sensors and actuators, so that if a fault occurs, the redundant device replaces the functionality of the faulty one. Additionally, redundant sensors allow to detect faults of the sensors themselves.

Analytical redundancy is a completely different approach, which consists of using an accurate model of the system to simulate the real process behavior. If a fault occurs, the difference between a measurement on the real system and the corresponding output of the model, which is called the *residual signal*, allows to detect a malfunction. The *digital twin* is the evolution of this approach.

1.3 Diagnostics and prognostics

Diagnostics deals with fault detection, isolation and identification of faults when a failure occurs. **Prognostics** deals with fault prediction before failure occurs and attempts to estimate the *Remaining Useful Life (RUL)* of a piece of equipment.

Both diagnostics and prognostics make use of automated methods to detect, diagnose and analyze the degradation of the performance of the equipment.

Some methods for diagnostics will be shown in chapters 4 and 5.

Data-driven methods Data-driven models are based upon statistical and learning techniques, such as *statistical hypothesis testing*, *Statistical Process Control* (SPC) for *univariate analysis* and *Principal Component Analysis* (PCA) for *multivariate analysis*. The main disadvantage of these methods is that the behavior of the system may depend on many internal and external factors, thus making necessary to collect and elaborate a huge amount of training data.

Analytical methods (mathematical models) Analytical methods make use of mathematical models which are directly tied to the physical processes that affect the health of related components in the equipment. These methods include the *analytical redundancy* approaches, like the *digital twin*, which make use of *residuals*, combined with statistical techniques to define thresholds for detecting the presence of faults.

Mathematical models work well under any load profile and in different operating conditions, and they can also be used to simulate component failures.

(peng) functional mapping between drifting parameters and selected prognosis features can be established...

1.4 Failure analysis

Maintenance can be very complicated in large systems, because every component may be associated with many failure modes, and it is difficult to understand how the failure of a single component affects the whole system. Independently from the maintenance strategy, it may be useful to preform a Failure Mode and Effect Analysis (FMEA) and to build a fault tree in order to have a better understanding of the system from a maintenance point of view.

Chapter 2

The role of the models in Predictive Maintenance

Mathematical models are widely used in engineering application for both design and control of dynamical systems. They can be very useful during the project phase because they allow to simulate the behavior of the system, they are used on the real plant to tune control parameters, and they are required by advanced control techniques (such as Kalman filtering and MPC). With the introduction of predictive maintenance techniques, models have been assuming an increasingly important role in maintenance applications: a good model allows to study the behavior of a system in both healthy and faulty conditions, and this leads to many possible uses.

2.1 Classification of mathematical models

Physical systems can be modeled in many ways. In order to simulate the dynamic behavior of a system, the differential algebraic equations (DAEs) that describe it are commonly used. On the other hand, transfer functions are widely used in control engineering applications to describe the dynamic behavior of a system with the purpose of tuning a control system.

Causal vs. acausal models Depending on how the equations are written and solved, it is possible to identify two types of models. **Acausal models** are those in which equations are written as implicit relations between some variables, that is, implicit functions in the form $f(v_1(t), v_2(t), \dots) = 0$, and they are not intended as *"right-to-left"* assignments; equations can be written in any order, and the way of solving them is left to the simulation algorithm. **Causal models** are those in which equations are written as explicit assignments and solved in a specific order; in other words, equations can be seen as functions which take some *known* arguments as input in order to calculate the value of an output variable, that is, $o(t) = f(i_1(t), i_2(t), \dots)$.

While causal models are written having in mind the resolving algorithm, acausal models focus on the physical reality they want to describe, and it is simpler to write and solve them as they don't require to manipulate mathematical expressions and to solve differential equations. Traditionally, simulation environments make use of causal models, but nowadays modern simulation environments allow to use acausal models.

First-principle models A first-principles model (FPM) is a mathematical model that is made up of the fundamental equations describing the underlying physical laws of a system, that is, the behavior of its component and the connections among them.

For example, the behavior of a simple RC circuit powered by a constant voltage source is described by the following equations:

$$\begin{array}{ll} v_R(t) = R\, i_R(t) & \textit{constitutive equation of the resistor} \\ i_C(t) = C\, dv_C(t)/dt & \textit{constitutive equation of the capacitor} \\ v_R(t) + v_C(t) = v(t) & \textit{Kirchhoff's voltage law} \\ i_R(t) - i_C(t) = 0 & \textit{Kirchhoff's current law} \\ v(t) = V_{DC} & \textit{voltage source} \end{array} \tag{2.1}$$

The set of equations (2.1) describes an acausal model. Depending on what we want to know, some of the equations of a FPM may become unnecessary. For example, if we want to know $v_C(t)$ and $i_C(t)$ we can write:

$$i_C(t) = C\, dv_C(t)/dt\ , \quad R\, i_C(t) + v_C(t) = V_{DC} \tag{2.2}$$

If we are only interested in knowing $v_C(t)$ given $v(t)$ we can write:

$$RC\, \frac{dv_C(t)}{dt} + v_C(t) = V_{DC} \tag{2.3}$$

The reduced set of equations (2.2) and equation (2.3) alone are no longer FPMs, but they still represent acausal models, and they require further elaboration in order to be solved. Equation (2.3) is a simple DAE whose solution is:

$$v_C(t) = [v_C(0) - V_{DC}] \cdot e^{-t/RC} + V_{DC} \tag{2.4}$$

Equation (2.4) is a causal model, as it is possible to identify one input $v(t) = v_{DC}$ and one output v_C and it can be written as $v_C(t) = f(v(t))$.

Block diagrams Block diagrams are based on *oriented* blocks and connections. A block is a graphical element representing a mathematical function, and it has input and output ports. A connection between two (or more) blocks represent the oriented signal flow from an output of the first block to an input of the second block (an output can be connected to more inputs).

The value of the output of the first block must be computed before being given as input to the second block. Block diagrams are causal models.

Although blocks can contain any kind of mathematical functions, they are often used to represent transfer functions, which allow to model dynamic systems in an easier way.

Simulink is a commercial modelization and simulation environment focused on casual models. The modelization environment allows to build block diagrams in both continuous and discrete time domains. A block can have both input and output ports, and an input port can be connected to one or more output ports. A connection represent an oriented signal flow. Commonly used blocks represent:

- transfer functions, dynamical systems in state-space form, non-linearities and discontinuities (blocks with one input and one output)
- mathematical operations (blocks with two inputs and one output)
- signal and data sources (blocks with one output) and sinks (blocks with one input)
- subsystems (blocks with many inputs and/or outputs)

Block diagrams for control design applications are described in chapter 7.

Object diagrams Other kinds of diagrams, such as electrical schematics and Piping and Instrumentation Diagrams (P&IDs), represent physical objects and the connections between them. These diagrams are acausal models.

Modelica is an object oriented modelization language focused on acausal models. The language itself is open source, and many implementations exist, both free (like *OpenModelica* and *JModelica*) and commercial (like *Dymola* and *SimulationX*). Some implementations include a graphical modelization environment which allow to build an object diagram that is automatically translated into Modelica code. Unlike Simulink, Modelica objects mostly represent acausal block, whose ports, or *interfaces*, represent physical connections and are not oriented. Casual blocks and oriented signals are also allowed.

2.2 Purpose and complexity of the models

Mathematical model can be used for many purposes. These include:

- **System design.** Models are often used in project phase to check if the system behaves as expected, and if it meets the requirements.
- **Control design.** Many control techniques require to know the behavior of the system. Regulator tuning often relies on the transfer function of the system, and advanced techniques require the state-space representation of the system.

- **Hazard analysis.** A model allow to simulate the behavior of the system in a worst-case scenario.
- **State observation.** In some applications the value of non-observable state variables can be estimated from a measurement of input and output variables.
- **Parameter identification.** Given a model of the system, it is possible to estimate unknown parameters given some measurements on the system.
- **Hazard and degradation model.** The failure probability, the degradation level and the RUL of a component can be estimated from its age and from external operating condition.
- **Digital twin.** A model of the healthy system is continuously compared to the real system in order to detect abnormal behaviors.

The complexity of a model depend on its application: some require the model to be as precise as possible, some allow to do huge approximations.

Maintenance applications With the introduction of predictive maintenance techniques, models have been assuming an increasingly important role in fault detection, especially with the introduction of the digital twin. The main issue with model-based techniques is that it's almost impossible, or extremely difficult and time-consuming, to create a good model, especially for larger systems. Furthermore, the effect of boundary conditions such as the external temperature is difficult to be considered in the model, as this would requires additional measures on the actual system which would have to be given as input to the model simulation.

If the model isn't good enough the simulation diverges from the actual system independently from the health status of the equipment. Therefore, it's important to understand whether the divergence is caused by modeling errors or equipment failures. This can be done using a machine learning system, which have to be trained using data collected from the plant.

Control design applications On the contrary, models used in control applications are never complete, and don't need to be, as a good feedback controller should be robust enough with respect to modeling approximations and parametric uncertainties, as well as measurement errors and exogenous variables. Similarly, models used in project phase take into account only the behaviors of the system we are interested in. However, it could be possible to adapt these simple models for maintenance applications.

Approximations Depending on the application, the following characteristics of a system may be left out of the model:

- boundary conditions (e.g. the external environment)
- noises an disturbances
- other exogenous variables
- parameter shifts

From the modelization point of view, the following approximations may be introduced:

- linearization of non-linear behaviors (before writing the equations)
- linearization of non-linear equations around the operating point

If the system is described through its transfer functions, the following approximations are common:

- approximation of a transfer functions to its dominant poles
- approximation of a closed-loop transfer function to a low-pass filter

Online and offline uses in maintenance applications Online use consists in running a real-time simulation of a model in order to check if the real system is behaving as expected. In this case the non-negligible exogenous variables acting on the real system should be measured and given as input to the model. The typical example of online use is the digital twin.

Offline use consist in running many simulations in order to study the behavior of the system in different operating conditions. Collected data may be used to compare the behavior of the real system with the different behaviors of the model in order to identify the operating condition of the system that most likely led to the observed behavior.

2.3 Possible uses for the models

The main purpose of this research is to understand whether and how simple models such as the ones built for system and control design could be also employed for Predictive Maintenance (PdM) purposes. These models are usually too simple to be used as digital twins, but other possible uses can be found.

The main idea is to modify these models in order to simulate faults and disturbances, then compare the behavior of the model in different faulty conditions with its nominal behavior, and see if results of the comparison can be used on the real plant in order to detect and identify faults, as shown in fig. 2.1. Many ideas arose, for both online and offline uses:

- **Offline use**
    - Data-driven methods for diagnostics, which will be described in chapter 4, require to know the behavior of the system in healthy condition. If it is not possible to collect data from the system,

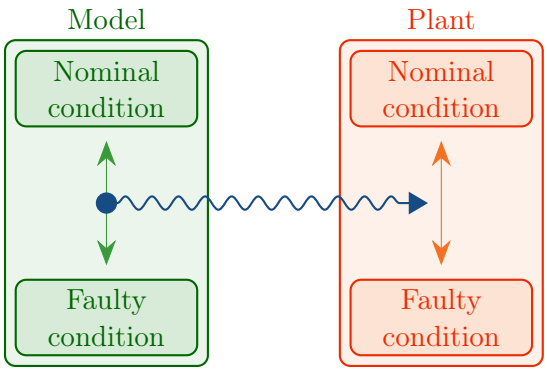


FIGURE 2.1: Model vs. Plant

they can be generated by the model, although the variance of the generated data may be smaller.

- The model allow to simulate a large number of different operating condition, and to collect and classify data for each of these. A measurement on the real plant can be compared to the simulation data in order to identify the operating condition that most likely led to the observed behavior.
- The model allow to see how any variation (of inputs, parameters, ...) propagates in the model, that is, which variables affects. This may allow to study how the effect of faults and degradation propagates, thus making possible to build fault trees and to understand how to structure a monitoring system.
- A model that include fault probability for each component allows to determine which faults are most likely to happen, and most important to happen simultaneously, so that the fault analysis can focus on these (combinations of) faults.
- The useful life of a component often depend on the environmental conditions and on its "mechanical" age (e.g. the number of revolutions of a rotating component), which can both be determined by using the model.

- **Online use**
  - Although the creation of the digital twin of the whole system may be extremely difficult, many small digital twins could be created for critical components only.
  - In many cases the effect of a failure in a system with a feedback control system may be compensated by the action of the controller. For example, the wear of a mechanical transmission may cause an increase of consumption of the actuator upstream, if the speed

of the component downstream is controlled. The value of control variables on the plant can be compared with the corresponding value on the model in order to detect an abnormal behavior.

- **Other uses**
  - The violation of a balance equation is a common symptom of a fault. The model can be useful to identify balance equations whose variables are measurable on the real system, in order to use these measures to detect faults.

Chapter 3

Critical analysis

3.1 Faults vs. disturbances

A **fault** can be seen as an event that leads to the failure of the system, that is, the inability to perform its function. A **disturbance** can be seen as a phenomenon that may affect the system performances but is often unavoidable; to a certain extent, its effects can be compensated or tolerated. Disturbances include signal noises, exogenous variables, unwanted harmonics (e.g. mechanical vibrations), and uncertainties of the model.

Sometimes the same phenomenon can be seen as both a fault or a disturbance, depending on its entity. For example, the increasing wear of a rotational mechanical component may introduce vibrations that are considered as a disturbance until they start to compromise the correct behavior of the system. In other cases, a fault and a disturbance may have the same effect on the system. The main issue with fault identification is to understand whether an unexpected behavior of the system is caused by a fault or by a disturbance.

In predictive maintenance applications, as both faults and disturbances may affect monitored variables of the real system it is necessary to understand what causes the deviation from the nominal behavior obtained from simulations. The following criteria may be used:

- **Threshold.** When the same phenomenon can be considered as both a fault or a disturbance, a threshold can be set. The entity of the phenomenon determines the entity of its effect on a monitored variable: this allows to set the maximum acceptable entity of the effect and determine the corresponding entity of the phenomenon, or vice versa.
- **Triggering limit.** In many cases it is possible to assume that the effect of disturbances on the monitored variables is negligible with respect to the effect of faults, therefore a triggering limit could be set. If the deviation of the monitored variable with respect to the nominal

behavior

- **Trends.** Wear is usually increasing in time, therefore the entity of its effect on the system is also increasing. The entity of the effect of noises is usually constant or depends on external factors such as environmental causes or electromagnetic fields. Exogenous variables may affect the system in many ways, and their effect may have the same entity as the effect of a fault, but in this case the entity may both increase and decrease in time. Keeping track of the past values of monitored variables can be useful to understand the cause of a phenomenon.

- **Harmonic content.** If the bandwidth of a noise is known, it is possible to know its effect on monitored variables.

- **Speed of change.** The speed of change of some disturbances and the incremental wear rate may be knewn a priori. In case of sudden faults, the effect on monitored variable may be gradual as most physical systems act as a low-pass filter, but the speed of change can be computed or simulated.

3.1.1 Modelization of disturbances

Disturbances are usually not included in the model, unless it is important to know their effect. Even if they were included, they are unknown by nature, although some characteristics such as their harmonic content may be known *a priori*. However, most of the aforementioned criteria require to know the effect of disturbances.

Transfer function block diagrams Casual models used for control design (block diagrams) usually model additive disturbances as generic inputs whose bandwidth is known (see sections 5.1 and 7.1). These models allow to find the transfer functions from any disturbance to any other variable of the system. This allow to design the controller so that the effect of the disturbances is mitigated, that is, the magnitude of these transfer functions is smaller than 1 dB in the disturbance bandwidth.

In predictive maintenance applications, transfer functions allow to know the effect of a disturbance on the monitored variables. In case of high-frequency noises, their bandwidth is often known and determining the bandwidth of their contribution on the monitored variables is trivial. In case of exogenous variables, the analysis could be restricted to their step and frequency response in the following cases: average value, maximum and minimum value, maximum admissible step variation. These analysis can be performed using Simulink or any simulation environment for acausal models. It must be remembered that transfer functions often approximate the behavior of the

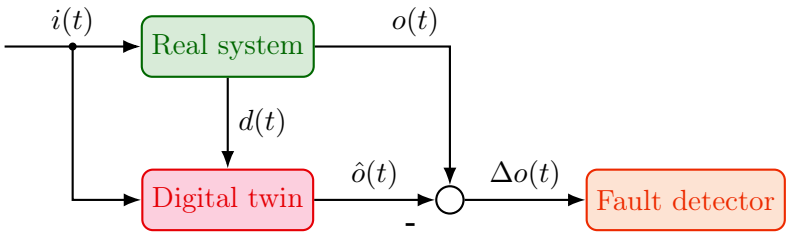


FIGURE 3.1: Digital twin

system in a specific operating point, and the results of these analysis may be not reliable.

Object diagrams and other acausal models Blocks in object diagrams usually describe the behavior of the component in the time domain. In this case it is not possible to obtain the step and frequency response of disturbances in analytical way, but any kind of disturbance can be simulated. As for the block diagrams, the analysis could be restricted to some cases.

3.2 Digital twin

The *digital twin* shown in fig. 3.3 is a model of the system which is run in parallel with the system itself. In order to perfectly replicate the behavior of the system, the digital twin must know all variables acting on it, including command inputs, exogenous variables, and other disturbances. While command inputs are known, disturbances must be measured on the real system. In predictive maintenance applications, the digital twin can be used to replicate the nominal behavior of the system. In order to detect faults, some measurements on the system (output variables $o(t)$) are compared with the corresponding output $\hat{o}(t)$ of the simulation. If the model were a perfect replica of the system, including disturbances, in absence of faults it must be $\Delta o(t) = o(t) - \hat{o}(t) = 0$. Unfortunately, as previously discussed in chapter 2, the model is never a perfect replica, and unmodeled disturbances often affect monitored variables, therefore $\Delta o(t) \neq 0$.

However, if the system is simple enough, a precise model can be built, so that the result of the simulation is very close to the behavior of the real system, that is, $\Delta o(t) \approx 0$. If the effect of unmodeled disturbances is negligible with respect to the effect of faults, it is possible to set a tolerance threshold on $\Delta o(t)$ below which the difference between $\hat{o}(t)$ and $o(t)$ is most likely caused by disturbances and modeling errors.

In the case of complex systems, it is still possible to build a digital twin for

some components only. This also allows to avoid the amplification of errors, as the digital twin of a subsystem will take as input the measurement of the output of the upstream component of the real system and not the result of the simulation of its model.

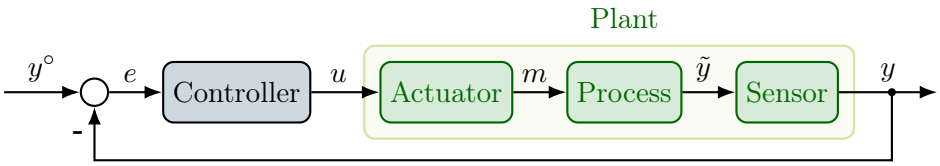


FIGURE 3.2: Typical feedback control scheme

Digital twin and feedback control systems Figure 3.2 shows the typical feedback control scheme. A first example of digital twin for a feedback control system consists in a model of the plant that is fed with the output of the controller, as shown in fig. 3.3. Another possible solution is to model the process and the sensors only, and feed the model with a measurement of the manipulated variable $m(t)$, as shown in fig. 3.4.

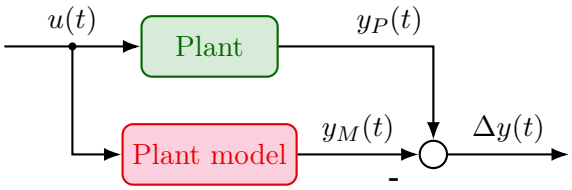


FIGURE 3.3: Digital twin of the whole plant

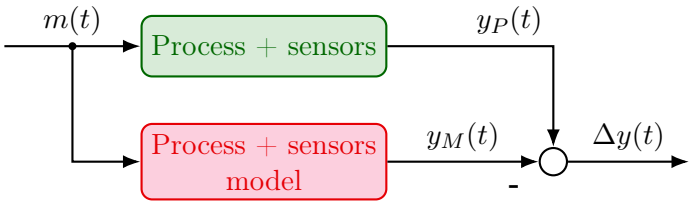


FIGURE 3.4: Digital twin of the process only

In some situations, the presence of a fault may not affect the controlled variable $y(t)$, as its effect could be balanced by the controller. In other words, the fault of a component may not lead to a system failure, as the system may be still able to perform its main function, and it would be $\Delta y(t) \approx 0$. We consider for example a water level control system consisting of a tank and a controlled valve that regulates the inlet flow. In presence of a small leakage, the inlet flow may be sufficient to compensate it, however the valve aperture would be bigger than in absence of leakages, that is, the value of the control variable $u(t)$ would be bigger. In this case, the configuration for the digital twin shown in fig. 3.5 could allow to detect the fault, as it would be $\Delta u(t) \neq 0$.

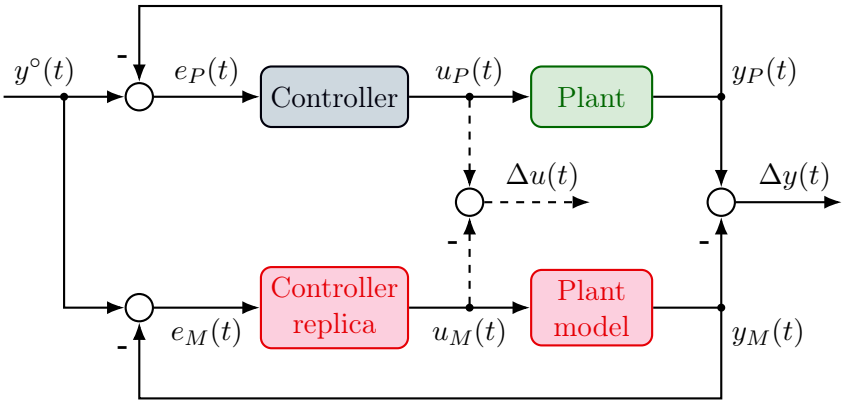


FIGURE 3.5: Digital twin (plant and controller)

In some situations the digital twin could be used in presence of unmodeled exogenous variables even if their effect is not negligible, such as highly variable external temperatures in a chemical plant. If exogenous variables can be measured, and an approximate transfer function from these variables to the monitored variables can be obtained, then it is possible to estimate the duration of their effect on the system, and increase the tolerance on $\Delta y(t)$ for the duration of the transient.

3.3 Residual generator (EDIT)

Residual generators are similar to the digital twin, and are based on the discrete-time transfer function of the model. The residual generator takes as input both the input and the output of the system, and gives as output a vector of *residuals* which can be used to detect and isolate one or more faults. This technique will be described in chapter 5.

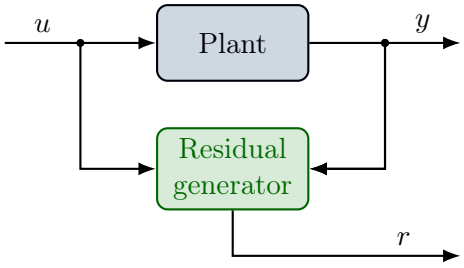
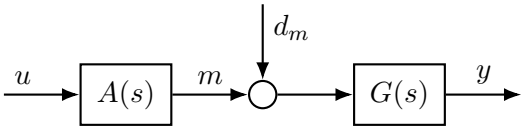


FIGURE 3.6: Residual generator

3.4 Fault incidence matrix

Fault detection techniques require to monitor some variables. Different faults may affect different variables to different degrees. In order to detect and, more importantly, to isolate a fault, it is important to know a priori which variables should be monitored and how each fault affects each of these variables. This analysis should also be extended to disturbances, so that more information is available to distinguish between faults and disturbances.

An incidence matrix is a matrix that shows the relationship between the elements corresponding to rows and the elements corresponding to columns. In predictive maintenance applications, incident matrices can be used to show how input variables of parameter changes affect output variables. An input variable can be a command, a disturbance, or the output of a component upstream.



The incidence matrix of this simple block diagram is:

	m	y
u	✓	✓
d_m		✓
m		✓

The elements of a fault incidence matrix M may be boolean or real. In the first case, m_{ij} is true if a change of the variable corresponding to the i^{th} row of the matrix causes a change of the one corresponding to its j^{th} column (a threshold may be set in order to exclude variations caused by

other reasons). In the second case, m_{ij} may be an indicator of how much the variable changes.

The incidence matrix can be generated by analyzing the model or by simulating it many times for different values of input variables. Observations on the real plant may also allow to build an incidence matrix for a reduced set of variables.

Part II

Diagnostics

Chapter 4

Data-driven methods

Data-driven methods for diagnostics based on statistical analysis can be divided in two classes: univariate and multivariate analysis. The former involves the analysis of a single variable, the latter considers two or more variables.

4.1 Statistical Process Control

Statistical Process Control (SPC) is a quality control technique which employs statistical methods in order to understand if a significant change in a monitored variable of a process is due to the natural variability of the process (*common-cause* variations, process *in control*), or may be caused by malfunctions of the system (*special-cause* variations, process *out of control*).

Statistical distribution of a variable Assuming that the process is in control, if the monitored variable had a normal distribution $X \sim \mathcal{N}\left(\mu, \sigma^2\right)$, where μ and σ are the mean and the standard deviation of the variable (see definition A.2.1), then 99.7% of the observations would fall in the range $\mu \pm 3\sigma$. If the observations of the variable start to appear outside these limits, the process is likely to be out of control with respect to that variable. Narrower control limits, e.g. $\pm 2\sigma$, could be used, allowing to detect more special-cause variations but increasing the risk of false positives.

If the distribution of X is not normal, according to *Chebyshev's inequality*[1]the number of observations that fall in the range $\mu \pm 3\sigma$ is lower (down to 89%). However, many physical measurements are approximately normally distributed. The idea behind SPC is that the approximation can be improved by using means of *groups* of observations.

[1] *Chebyshev's inequality* states that no more than $1/k^2$ values of X can be more than $k\sigma_X$ away from μ_X, i.e. at least $100 \cdot (1 - 1/k^2)\%$ of the values of X are within $\mu_X \pm k\sigma_X$

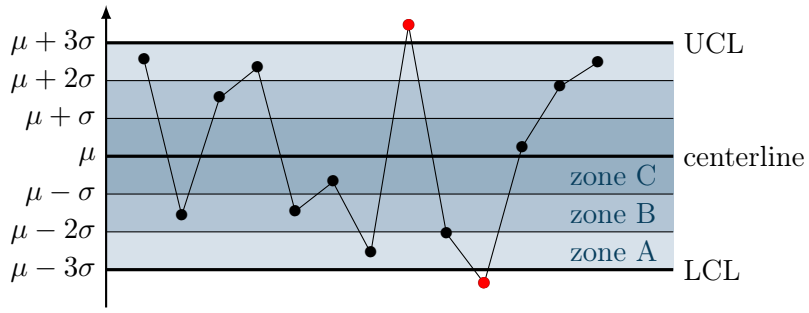


FIGURE 4.1: Centerline and control limits in control charts

Control charts SPC makes use of *control charts*, each one representing a *statistic*, and decision rules to discriminate between common- and special-cause variations.

A typical control chart consist of:

- points representing a *statistic*, such as the sampled mean and variance of groups of data (see appendix A.4)
- a *centerline*, representing the expected value of the statistic when the process is in control
- upper and lower *control limits* (UCL and LCL), representing an acceptable range for the statistic (tipically $\pm 3\sigma$)

4.1.1 $\bar{\text{X}}$-R and $\bar{\text{X}}$-s charts

$\bar{\text{X}}$-R and $\bar{\text{X}}$-s charts are used to monitor the variations of the measurements of a process variable X. Given k groups of data of n samples each (tipically, $k \geq 20$ and $n = 3 \div 6$) we compute for each group i the sample mean $\bar{X}_i$ and the dispersion (range R or sample standard deviation s).

The centerline of the $\bar{\text{X}}$ chart is the average $\bar{\bar{X}}$ of $\bar{X}_i$, and the control limits are $\bar{\bar{X}} \pm 3\hat{\sigma}_{\bar{X}}$, where $\hat{\sigma}_{\bar{X}}$ is an estimate of the standard deviation $\sigma^2_{\bar{X}} = \sigma^2/\sqrt{n}$ of $\bar{\bar{X}}$. These values are computed as follows, using the parameters in table C.1. Finally, empirical rules are used to identify special-cause variations.

$\bar{\text{X}}$-R charts

In $\bar{\text{X}}$-R charts $\bar{\bar{X}} \pm \hat{\sigma}_{\bar{X}}$ is estimated from the range R as $\bar{\bar{X}} \pm A_2\bar{R}$, and the standard deviation of X is estimated as $\sigma^2_X = \bar{R}/d_2$.

	$\bar{\text{X}}$ charts	R charts
Centerline	$\bar{\bar{X}} = \frac{1}{k}\sum_{i=1}^{k} \bar{X}_i$	$\bar{R} = \frac{1}{k}\sum_{i=1}^{k} R_i$
Upper Control Limit	$UCL_{\bar{X}} = \bar{\bar{X}} + A_2\bar{R}$	$UCL_R = D_4\bar{R}$
Lower Control Limit	$LCL_{\bar{X}} = \bar{\bar{X}} - A_2\bar{R}$	$LCL_R = D_3\bar{R}$

$\bar{\text{X}}$-s charts

In $\bar{\text{X}}$-s charts $\bar{\bar{X}} \pm \hat{\sigma}_{\bar{X}}$ is estimated from the sample standard deviation s as $\bar{\bar{X}} \pm A_3\bar{s}$, and the standard deviation of X is estimated as $\sigma_X^2 = s/c_4$.

	$\bar{\text{X}}$ charts	s charts
Centerline	$\bar{\bar{X}} = \frac{1}{k}\sum_{i=1}^{k} \bar{X}_i$	$\bar{s} = \frac{1}{k}\sum_{i=1}^{k} s_i$
Upper Control Limit	$UCL_{\bar{X}} = \bar{\bar{X}} + A_3\bar{s}$	$UCL_s = B_4\bar{s}$
Lower Control Limit	$LCL_{\bar{X}} = \bar{\bar{X}} - A_3\bar{s}$	$LCL_s = B_4\bar{s}$

Western Electric Rules (WER)

Western Electric Rules are a set of empirical decision rules for detecting non-random patterns in control charts [Com58].

A process is out of control when one of the following conditions hold:

- one or more points exceeding the control limits
- 2 out of 3 consecutive points exceeding the warning limits $\pm 2\sigma_X^2$
- 4 out of 5 consecutive points exceeding the band $\pm\sigma_X^2$
- 8 consecutive points on the same size of the centerline

If a non-random pattern has been detected, the procedure in fig. 4.2 is followed.

Control limits must be recomputed when:

- the process is changed
- the cause of the change is known
- the cause is expected to be persistent
- enough data are available

4.2 Multivariate analysis

Univariate analysis produces many indicators that cannot be easily correlated to the presence of faults, and some faults do not result in threshold violations. In these cases a multivariate analysis is required in order to properly consider the correlation among the variables.

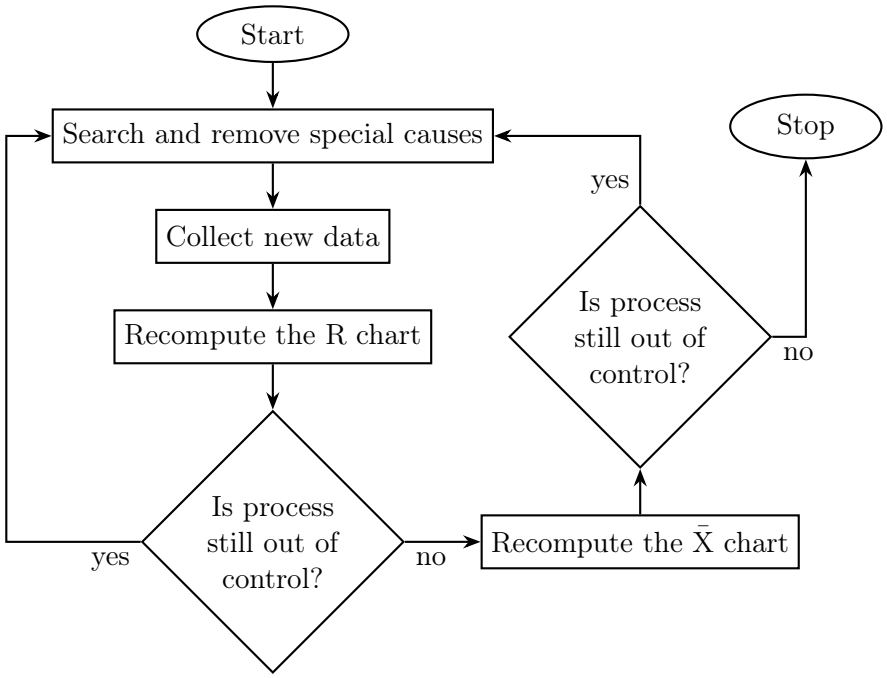


FIGURE 4.2: Identification of non-random patterns

The statistical distance T^2 allows to define a multivariate confidence region which can be used with hypothesis testing (see appendix A). Principal Component Analysis (PCA) is a technique that can be used when there are many variables and few measurements, as it allows to reduce the dimensionality of the training set.

Chapter 5

Analytical methods

5.1 Classification of faults and disturbances

The behavior of a system may differ from the behavior of its model due to the presence of:

- additive faults on the sensors and the actuators
- noises affecting the sensors, the actuators and the plant signals
- additive disturbances (exogenous variables) acting on the plant
- additive faults on the plant (leakages, ...)
- multiplicative faults on the plant (variations of the model parameters)
- modeling errors on the plant (uncertainty of the model transfer function or of the *underlying parameters*)

	Additive	**Multiplicative**
Faults	Sensor faults f_y Actuator faults f_u Additive plant faults f_M	Parametric plant faults $\delta\vartheta_F$
Disturbances	Plant disturbances d_M	Modeling errors ΔM, $\delta\vartheta_E$
Noises	Sensor noises ξ_y Actuator noises ξ_u Plant noises ξ_M	–

TABLE 5.1: Classification of faults, noises and disturbances

Additive fault, noises and disturbances Some kinds of fault, like sensor and actuator faults and leakages in the plant, can be represented as unknown extra signals that sum up to the input, output and internal signals of the plant. These signals may exhibit different behaviors (drifts, steps, intermittent, sinusoidal). Signals of the plant are also affected by noises, which usually are

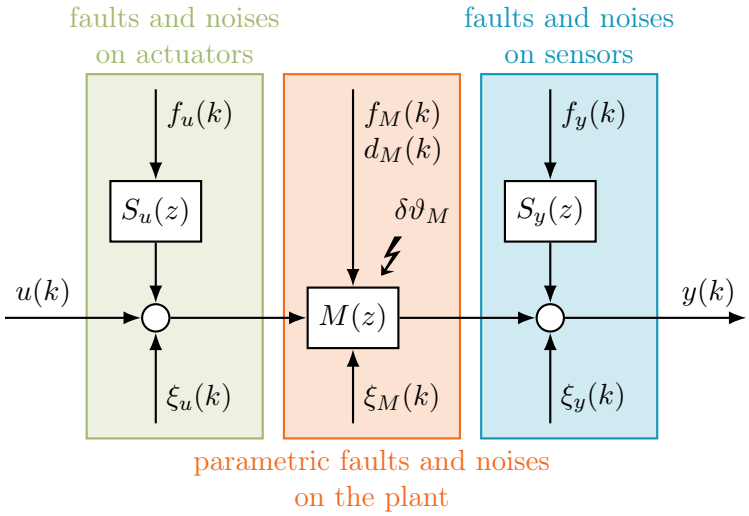


FIGURE 5.1: Faults and disturbances

persistent signals with zero mean and high frequencies and can be treated differently from the additive faults. Finally, the system may be affected by disturbances, that are exogenous inputs (like the external temperature) that don't represent a fault.

Multiplicative faults and disturbances Other kinds of fault can be represented as a change in the behavior of the plant, that is, the transfer function of the faulty plant becomes different from the model one. In a similar way, the latter may be already different from the (healthy) plant one, because of the unavoidable modeling errors.

Multiplicative faults and disturbances can be described in two ways:

- We denote with $\tilde{M}(z)$ the transfer function of the system, and with $M(z)$ the transfer function of its model. We define:
  - the discrepancy $\Delta\tilde{M}_F(z)$ between the system $\tilde{M}(z)$ in its current (healthy or faulty) state and the healthy system $\tilde{M}^\circ(z)$;
  - the discrepancy $\Delta M_E^\circ(z)$ between the healthy system $\tilde{M}^\circ(z)$ and its model $M^\circ(z)$.

  Hence we write:

$$\tilde{M}^\circ(z) = M^\circ(z) + \Delta M_E^\circ(z)\,, \quad \tilde{M}^\circ = \tilde{M}(z) - \Delta\tilde{M}_F(z) \quad \Longrightarrow$$
$$\tilde{M}(z) = M^\circ(z) + \Delta M(z)\,, \quad \Delta M(z) \triangleq \Delta\tilde{M}_F(z) + \Delta M_E^\circ(z)$$

- If all the parameters of the model that are subject to faults and uncertainty can be defined with respect to a small set $\boldsymbol{\vartheta} = (\vartheta_i)$ of *underlying parameters*, we can define $M(z)$ as a function of $\boldsymbol{\vartheta} = \boldsymbol{\vartheta}^\circ + \delta\boldsymbol{\vartheta}$,

where $\boldsymbol{\vartheta}^\circ$ represents the real value of the underlying parameters and $\delta\boldsymbol{\vartheta} = \delta\boldsymbol{\vartheta}_F + \delta\boldsymbol{\vartheta}_E$ the discrepancy due to faults and modeling errors. The transfer function of the model of the healthy system is $M(z;\boldsymbol{\vartheta}^\circ)$; note that, in general, $M(z;\boldsymbol{\vartheta}^\circ) \neq \tilde{M}^\circ(z)$, because the model is still an approximation of the reality, independently from the accuracy of its parameters.

5.2 Parity space methods

The purpose of parity space methods is to check the *parity* (consistency) of the measurements acquired from the monitored system with their expected value (in absence of faults). These methods make use of *parity equations*, which are relations between the observable signals of a system that always hold when no unknown inputs (faults and disturbances) are present, e.g. $f(y(t)) = g(u(t))$ or $p(y(t),\, u(t)) = 0$.

Parity equations can be used to generate *residuals*, which are quantities that have to be zero in absence of faults.

5.2.1 Generic residual generator

Consider a discrete-time MIMO system described with the generic I/O relationship:

$$y(k) = M(z)\, u(k) + S(z)\, f(k) \tag{5.1}$$

where $u(k) \in \mathbb{R}^m$ is the vector of inputs, $y(k) \in \mathbb{R}^p$ is the vector of outputs, $f(k) \in \mathbb{R}^\nu$ is the vector of additive faults acting on the system, and $M(z)$ and $S(z)$ are transfer functions of the appropriate size.

The *generic residual generator* is a linear discrete-time system:

$$\boldsymbol{r}(k) = V(z)\, u(k) + W(z)\, y(k) \tag{5.2}$$

where $\boldsymbol{r}(k) \in \mathbb{R}^q$ is the vector of residuals (or *residual set*), and $V(z)$ and $W(z)$ are transfer functions of the appropriate size such that the parity equation $\boldsymbol{r}(k) = \mathbf{0}$ is verified when $f(k) = 0 \;\Leftrightarrow\; y(k) = M(z)\, u(k)$.

Substituting eq. (5.1) into eq. (5.2) gives:

$$\begin{aligned} \boldsymbol{r}(k) &= V(z)\, u(k) + W(z) \cdot [M(z)\, u(k) + S(z)\, f(k)] = \\ &= [V(z) + W(z)\, M(z)]\, u(k) + W(z)\, S(z)\, f(k) \end{aligned} \tag{5.3}$$

Hence, in order to have the residuals equal to zero when no faults occur, the transfer functions $V(z)$ and $W(z)$ must satisfy the *parity equation*:

$$[V(z) + W(z)\, M(z)]\, u(k) = 0$$

from which we obtain:

$$V(z) = -W(z)\, M(z) \tag{5.4}$$

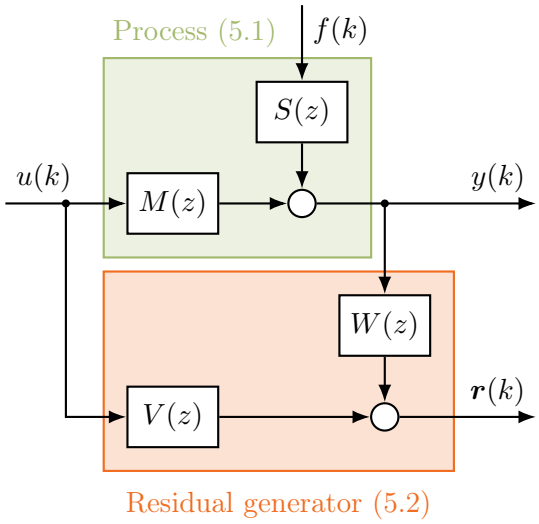


FIGURE 5.2: Generic residual generator from eqs. (5.1) and (5.2)

Substituting eq. (5.4) into eq. (5.2) gives the following equation, called *computational form* of the residual generator:

$$\boldsymbol{r}(k) = W(z) \cdot [y(k) - M(z)\, u(k)] \tag{5.5}$$

Finally we substitute eq. (5.4) into eq. (5.3) (or equivalently we combine eqs. (5.1) and (5.5)), and we obtain:

$$\boldsymbol{r}(k) = -W(z)\, M(z)\, u(k) + W(z) \cdot [M(z)\, u(k) + S(z)\, f(k)]$$

which leads to the following equation[1], called *internal form* of the residual generator, where $Z(z) = W(z)\, S(z)$ and $\tilde{f}(k) = S(z)\, f(k)$:

$$\boldsymbol{r}(k) = W(z)\, S(z)\, f(k) = W(z)\, \tilde{f}(k) = Z(z)\, f(k) \tag{5.6}$$

While we use eq. (5.5) to actually generate the residuals, eq. (5.6) shows how they depend on the faults. By properly choosing the transfer function Z(z) we can give them the desired form, that is, the one which better allows to detect failures.

The *response set* to a fault f_i is defined as:

$$\boldsymbol{r}(k|f_i) = W(z)\, S_{*,i}(z)\, f_i(k) = Z_{*,i}(z)\, f_i(k) \tag{5.7}$$

and the response of a single residual $r_h(k)$ is:

$$r_h(k|f_i) = W_{h,*}(z)\, S_{*,i}(z)\, f_i(k) = z_{h,i}(z)\, f_i(k) \tag{5.8}$$

[1]Since we only know an approximation M of the real transfer function $\tilde{M}$, we should write eq. (5.1) as $\boldsymbol{r} = (V + W\tilde{M})u + WSf$, and hence eq. (5.6) as $\boldsymbol{r} = W(\tilde{M} - M)u + WS\hat{f}$, so that $r = W(\tilde{M} - M)u \approx 0$ for $f = 0$ and $M \approx \tilde{M}$

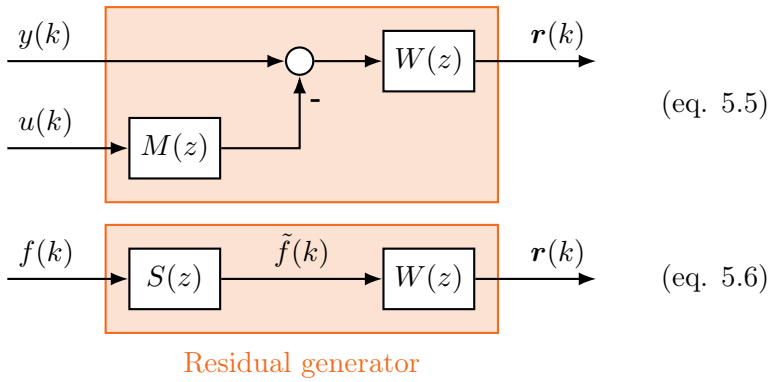


FIGURE 5.3: Generic residual generators (parity space equations)

Specification and implementation Detection and isolation properties of a residual r_h depend on the given *specification* $Z_h(z)$, which is realized through a suitable *implementation* $W_h(z)$.

A specification $Z_h(z) = \begin{bmatrix} 0 & \dots & 0 \end{bmatrix}$, which is made by ν rational functions for $f \in \mathbb{R}^\nu$, is said *homogeneous*, otherwise it is said *non-homogeneous*.

An implementation $W_h(z)$ is made by p rational functions, therefore it can't satisfy more than p conditions. A specification with $\nu = p$ is said *full*, and a specification with $\nu = p - 1$ is said *almost-full*. There is a special case of specification which contains $p - 1$ zero elements and one non-zero element, and this is called *full almost-Heneous specification*.

5.2.2 Isolation properties of the residual generator

A single residual allows to detect the presence of a fault, but its isolation among ν independent faults $\mathcal{F} = \{f_1, \dots, f_\nu\}$ requires a set of at least $q \geq \nu$ residuals $\boldsymbol{r} = \begin{bmatrix} r_1 & \dots & r_q \end{bmatrix}^\mathsf{T}$.

Structured residuals

When a fault f_i occurs, some residuals respond, and the others do not. Structured residuals are designed so that each residual r_h is sensitive to a different subset $\mathcal{F}_h$ of faults.

Each fault f_i is characterized by the pattern of the response set $\boldsymbol{r}(k|f_i)$. We define the *fault signature*, or *fault code*, as the binary column vector $\boldsymbol{\varphi}_i \in \mathbb{R}^q$ whose h^{th} element is 1 if r_h is sensitive to f_i, and 0 otherwise:

$$\boldsymbol{\varphi}_i = \begin{bmatrix} \varphi_{1,i} & \dots & \varphi_{q,i} \end{bmatrix}^\mathsf{T} \qquad (\boldsymbol{\varphi}_i)_h = \begin{cases} 0, & f_i \in \mathcal{F}_h \\ 1, & f_i \notin \mathcal{F}_h \end{cases}$$

The *matrix of the signatures* $\Phi = (\varphi_{h,i}) = \begin{bmatrix}\boldsymbol{\varphi_1} & \dots & \boldsymbol{\varphi_\nu}\end{bmatrix}$ is the $q \times \nu$ matrix whose columns $\boldsymbol{\varphi_i}$ are the fault signatures. It can be defined as a matrix whose elements are 1 if the fault $f_i(k)$ affects the residual $r_h(k)$, and 0 otherwise, and it represents the pattern of the specification $Z(z)$.

$$\Phi = \begin{array}{c|cccc} & f_1 & f_2 & \cdots & f_\nu \\ \hline r_1 & 1 & 1 & \cdots & 0 \\ r_2 & 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_q & 1 & 0 & \cdots & 0 \end{array} \qquad \Phi_{*,i} = \boldsymbol{\varphi}_i = \begin{bmatrix}\varphi_{1,i} \\ \vdots \\ \varphi_{q,i}\end{bmatrix} \qquad \varphi_{h,i} = \begin{cases}0, & z_{h,i}(z) \approx 0 \\ 1, & z_{h,i}(z) \neq 0\end{cases}$$

Observed fault code and fault isolation When a non-null response set is observed, we apply a threshold test:

$$\varepsilon_h = \begin{cases}0\,, & |r_h(k)| < \lambda_h \\ 1\,, & |r_h(k)| \geq \lambda_h\end{cases} \quad , \qquad h = 1 \dots q$$

where $\lambda_i \geq 0$ is the threshold beyond which the residual is considered to be responding. The vector $\boldsymbol{\varepsilon} = \begin{bmatrix}\varepsilon_1 & \dots & \varepsilon_q\end{bmatrix}^\mathsf{T}$ is the observed fault signature, and we can isolate the fault by comparing $\boldsymbol{\varepsilon}$ with the known fault signatures: if $\boldsymbol{\varepsilon} = \boldsymbol{\varphi}_i$ then the fault f_i has occurred.

Triggering limit Assuming that a fault f_i has a (known) nominal value $\bar{f}_i$, the steady-state step response of r_h to f_i is:

$$\lim_{k\to\infty} \left| r_h\Big(k \;\Big|\; \bar{f}_i^{step}\Big)\right| = |W_{h,*}(z)\, f_i(z)|_{z=1} \tag{5.9}$$

We define the (normalized) *triggering limit* as:

$$\eta_{h,i} = \frac{\lambda_h}{\bar{f}_i\, |W_{h,*}(z)\, f_i(z)|_{z=1}} \tag{5.10}$$

A small triggering limit $\eta_{h,i}$ indicates a high sensitivity of r_h to the fault f_i. If $\eta_{h,i} > 1$ the fault f_i does not bring the residual r_h to its threshold λ_i at steady-state, so it should be $\eta_{h,i} < 1\,,\ \forall\,(h,i)$.

Multiple fault isolation with structured residuals When more than one fault is present simultaneously, the response set to these faults is equal to the sum of the individual response sets to each fault[2], that is:

$$\boldsymbol{r}(\,\cdot \mid f_i \in \mathcal{F}_l) = \sum_{f_i \in \mathcal{F}_l} \boldsymbol{r}(\,\cdot \mid f_i)\,, \quad \mathcal{F}_l \subseteq \mathcal{F}$$

[2]in some very rare cases, the effects of the faults may cancel each others

The signature $\varphi_{\mathcal{F}_l}$ of a combination of faults $\mathcal{F}_l$ is generally equal to the boolean sum of the signatures φ_i of the faults $f_i \in \mathcal{F}_l$, and we can define an augmented matrix of the signatures that includes all the fault combinations $\mathcal{F}_l$ or the most probable ones (e.g. all the combinations of 2 or 3 faults):

$$\Phi_{\text{aug}} = \begin{array}{c|cccc|ccc} & f_1 & f_2 & \cdots & f_\nu & \mathcal{F}_1 & \mathcal{F}_2 & \cdots \\ \hline r_1 & 1 & 1 & \cdots & 0 & 1 & 1 & \cdots \\ r_2 & 0 & 1 & \cdots & 1 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \\ r_q & 1 & 0 & \cdots & 0 & 1 & 1 & \cdots \end{array}$$

5.2.3 Linear systems in state-space representation

We consider the following linear state-space representation for a discrete-time dynamical system of the n^{th} order, describing a process subject to additive faults:

$$\begin{cases} x(k+1) = A\,x(k) + B_u\,u(k) + B_f\,f(k) & \text{(5.11a)} \\ y(k) = C\,x(k) + D_u\,u(k) + D_f\,f(k) & \text{(5.11b)} \end{cases}$$

where f is the vector of faults, and:

$$\begin{array}{lllll} x \in \mathbb{R}^n & u \in \mathbb{R}^m & A \in \mathbb{R}^{n\times n} & B_u \in \mathbb{R}^{n\times m} & B_f \in \mathbb{R}^{n\times \nu} \\ y \in \mathbb{R}^p & f \in \mathbb{R}^\nu & C \in \mathbb{R}^{p\times n} & D_u \in \mathbb{R}^{p\times m} & D_f \in \mathbb{R}^{p\times \nu} \quad p \le n \end{array}$$

Assuming that the pair (C, A) is observable, the behavior of the system in the sliding time window $[k-N, k]$ is described by:

$$\begin{aligned} \underbrace{\begin{Bmatrix} y(k-N) \\ y(k-N+1) \\ \vdots \\ y(k) \end{Bmatrix}}_{Y(k)} &= \underbrace{\begin{bmatrix} C \\ CA \\ \vdots \\ CA^N \end{bmatrix}}_{O} x(k-N) + \\ &+ \underbrace{\begin{bmatrix} D_u & 0 & \dots & 0 \\ CB_u & D_u & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N-1}B_u & CA^{N-2}B_u & \dots & D_u \end{bmatrix}}_{K} \underbrace{\begin{Bmatrix} u(k-N) \\ u(k-N+1) \\ \vdots \\ u(k) \end{Bmatrix}}_{U(k)} + \\ &+ \underbrace{\begin{bmatrix} D_f & 0 & \dots & 0 \\ CB_f & D_f & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N-1}B_f & CA^{N-2}B_f & \dots & D_f \end{bmatrix}}_{H} \underbrace{\begin{Bmatrix} f(k-N) \\ f(k-N+1) \\ \vdots \\ f(k) \end{Bmatrix}}_{F(k)} \end{aligned} \qquad (5.12)$$

$$
\begin{aligned}
&O\text{: matrix } (N+1)\,p \times n && Y \in \mathbb{R}^{(N+1)p} \\
&K\text{: matrix } (N+1)\,p \times (N+1)\,m && U \in \mathbb{R}^{(N+1)m} \\
&H\text{: matrix } (N+1)\,p \times (N+1)\,\nu && F \in \mathbb{R}^{(N+1)\nu}
\end{aligned}
$$

$$
Y(k) = O\,x(k-N) + K\,U(k) + H\,F(k) \tag{5.13}
$$

The observability matrix O, the Hankel matrices K and H, the inputs $U(k)$ and the outputs $Y(k)$ in the sliding window are known, while the faults $F(k)$ in the sliding window and the state $x(k-N)$ are not known.

We assume that $p \leq n$ and $\operatorname{Rank} C = p$. The pair (A, C) is observable if and only if $\operatorname{Rank} O = n$.

Primary and secondary residuals

Starting from eq. (5.13) we define the **primary residuals** as:

$$
\begin{aligned}
\boldsymbol{o}(k) &= Y(k) - KU(k) = && \text{(5.14a)} \\
&= Ox(k-N) + HF(k) \;\in\; \mathbb{R}^{(N+1)p} && \text{(5.14b)}
\end{aligned}
$$

and the **secondary residuals**, or simply *residuals*, as:

$$
\begin{aligned}
r(k) &= \boldsymbol{w}^{\mathsf{T}} \cdot [Y(k) - KU(k)] = && \text{(5.15a)} \\
&= \boldsymbol{w}^{\mathsf{T}} \cdot [Ox(k-N) + HF(k)] \;\in\; \mathbb{R} && \text{(5.15b)}
\end{aligned}
$$

where $\boldsymbol{w}^{\mathsf{T}} = \begin{bmatrix} \boldsymbol{w}_N^{\mathsf{T}} & \boldsymbol{w}_{N-1}^{\mathsf{T}} & \dots & \boldsymbol{w}_0^{\mathsf{T}} \end{bmatrix}$ is a row vector of $(N+1)\,p$ free parameters, with $\boldsymbol{w}_\kappa \in \mathbb{R}^p$ for $\kappa = 0 \dots N$.

Both primary and secondary residuals depend on the state $x(k-N)$ and the vector of faults $F(k)$. The design of the secondary residuals consist in the choice of a vector $\boldsymbol{w}$ such that:

- $\boldsymbol{w}^{\mathsf{T}} O = \mathbf{0}_n^{\mathsf{T}}$, so that $r(k) = \boldsymbol{w}^{\mathsf{T}} H\,F(k)$ is insensitive to the state $x(k-N)$;
- $\boldsymbol{w}^{\mathsf{T}} H = \boldsymbol{z}^{\mathsf{T}}$, where $\boldsymbol{z}^{\mathsf{T}} = \begin{bmatrix} \boldsymbol{z}_N^{\mathsf{T}} & \boldsymbol{z}_{N-1}^{\mathsf{T}} & \dots & \boldsymbol{z}_0^{\mathsf{T}} \end{bmatrix}$ is a row vector of $(N+1)\,\nu$ free design parameters, with $\boldsymbol{z}_\kappa \in \mathbb{R}^\nu$ for $\kappa = 0 \dots N$, which has to be chosen in order to guarantee identification and isolation properties.

The space of all the vectors $\boldsymbol{w}$ that decouple the residuals r from the state x, defined as $\mathcal{P} = \{\boldsymbol{w} \,|\, \boldsymbol{w}^{\mathsf{T}} O = \mathbf{0}_n^{\mathsf{T}}\}$ is called *parity space*, and any vector $\boldsymbol{w} \in \mathcal{P}$ is called *parity vector*.

If the conditions above are satisfied, the expression of secondary residuals in internal form (eq. 5.15b) becomes:

$$
r(k) = \boldsymbol{w}^{\mathsf{T}} H\,F(k) = \boldsymbol{z}^{\mathsf{T}} F(k) \tag{5.16}
$$

which can be expanded as:

$$r(k) = \boldsymbol{z}^\mathsf{T} F(k) = \sum_{\kappa=0}^{N} \boldsymbol{z}_\kappa^\mathsf{T} f(k-\kappa) = \sum_{\kappa=0}^{N} \sum_{i=1}^{\nu} z_{\kappa,i}\, f_i(k-\kappa) \tag{5.17}$$

By properly choosing the vector $\boldsymbol{z}^\mathsf{T}$ we can decide which faults does r depend on, as shown in eq. (5.17). Then, the parameters $\boldsymbol{w}^\mathsf{T}$ can be obtained from $\boldsymbol{z}^\mathsf{T}$ by solving the following linear system of $n + (N+1)\,\nu$ equations in $(N+1)\,p$ unknowns:

$$\boldsymbol{w}^\mathsf{T} \begin{bmatrix} O & H \end{bmatrix} = \begin{bmatrix} \boldsymbol{0}_n^\mathsf{T} & \boldsymbol{z}^\mathsf{T} \end{bmatrix} \tag{5.18}$$

A vector $\boldsymbol{z_h}$ is a residual specification; the corresponding vector $\boldsymbol{w_h}$ that allows to compute $r_h(k)$ from input and output data is the residual implementation.[3]

Equivalence with the generic residual generator

Equations (5.15a) and (5.16) can be rewritten as:

$$r(k) = \boldsymbol{w}^\mathsf{T} Y(k) + \boldsymbol{v}^\mathsf{T} U(k) = \boldsymbol{w}^\mathsf{T} H\, F(k) = \boldsymbol{z}^\mathsf{T} F(k) \tag{5.19}$$

where $\boldsymbol{v}^\mathsf{T} = -\boldsymbol{w}^\mathsf{T} K = \begin{bmatrix} \boldsymbol{v}_N^\mathsf{T} & \boldsymbol{v}_{N-1}^\mathsf{T} & \dots & \boldsymbol{v}_0^\mathsf{T} \end{bmatrix}$, $\boldsymbol{v}_k \in \mathbb{R}^m$. If we define the transfer function matrices

$$W(z) = \sum_{\kappa=0}^{N} \boldsymbol{w}_\kappa^\mathsf{T} z^{-\kappa}\,, \quad V(z) = \sum_{\kappa=0}^{N} \boldsymbol{v}_\kappa^\mathsf{T} z^{-\kappa}\,, \quad Z(z) = \sum_{\kappa=0}^{N} \boldsymbol{z}_\kappa^\mathsf{T} z^{-\kappa}$$

of size $(1 \times p)$, $(1 \times m)$ and $(1 \times \nu)$, respectively, we can write:

$$\boldsymbol{w}^\mathsf{T} Y(k) = \sum_{\kappa=0}^{N} \boldsymbol{w}_\kappa^\mathsf{T}\, y(k-\kappa) = \sum_{\kappa=0}^{N} \boldsymbol{w}_\kappa^\mathsf{T} z^{-\kappa}\, y(k) = W(z)\, y(k)$$
$$\boldsymbol{v}^\mathsf{T} U(k) = \dots = V(z)\, u(k)\,, \quad \boldsymbol{z}^\mathsf{T} F(k) = \dots = Z(z)\, f(k)$$

and eq. (5.19) can be rewritten as:

$$r(k) = W(z)\, y(k) + V(z)\, u(k) = Z(z)\, f(k) \tag{5.20}$$

which is identical to eqs. (5.2) and (5.6).

If we pre-multiply each member of eq. (5.13) by $\boldsymbol{w}^\mathsf{T}$, recalling that $\boldsymbol{w}^\mathsf{T} O = \boldsymbol{0}^\mathsf{T}$ we get:

$$\boldsymbol{w}^\mathsf{T} Y(k) = \boldsymbol{w}^\mathsf{T} K\, U(k) + \boldsymbol{w}^\mathsf{T} H\, F(k) = -\boldsymbol{v}^\mathsf{T} U(k) + \boldsymbol{s}^\mathsf{T} F(k)$$

[3] the (boldface) index $\boldsymbol{h}$ in $\boldsymbol{z_h}$ and $\boldsymbol{w_h}$ identifies a particular residual specification and the corresponding implementation, and it should not be confused with the index $\kappa = 1 \dots N$ denoting subvectors of $\boldsymbol{z}$ and $\boldsymbol{w}$ in the previous equations

where $\boldsymbol{z}^\mathsf{T} = \boldsymbol{w}^\mathsf{T} H = \begin{bmatrix}\boldsymbol{z}_N^\mathsf{T} & \boldsymbol{z}_{N-1}^\mathsf{T} & \cdots & \boldsymbol{z}_0^\mathsf{T}\end{bmatrix}$, $\boldsymbol{z}_i \in \mathbb{R}^\nu$.

Assuming that we can find two transfer function matrices $M(z)$ and $S(z)$ such that $W(z)\,M(z) = -V(z)$ and $W(z)\,S(z) = Z(z)$, we can write:

$$W(z)\,y(k) = -V(z)\,u(k) + Z(z)f(k) = W(z)\,M(z)\,u(k) + W(z)\,S(z)\,f(k)$$

from which we obtain

$$y(k) = M(z)\,u(k) + S(z)\,f(k)$$

which is identical to eq. (5.1).

Equation (5.19) can be expanded as:

$$r(k) = \frac{\boldsymbol{w}_N^\mathsf{T}\,z^N + \cdots + \boldsymbol{w}_1^\mathsf{T}\,z + \boldsymbol{w}_0^\mathsf{T}}{z^N}\,y(k) + \frac{\boldsymbol{v}_N^\mathsf{T}\,z^N + \cdots + \boldsymbol{v}_1^\mathsf{T}\,z + \boldsymbol{v}_0^\mathsf{T}}{z^N}\,u(k)$$

This shows that the residual has a *dead-beat* response of N^{th} order, which settles in at most N steps.

Residual specifications

In order to be able to freely choose the structure of the secondary residuals, assuming that $\boldsymbol{z} \neq \boldsymbol{0}$ (*non-homogeneous specification*) the number of the unknowns must be greater than or equal to the number of equations:

$$\begin{aligned}\underline{\boldsymbol{z} \neq \boldsymbol{0}} \qquad \#\mathsf{unknowns} \geq \#\mathsf{equations} \qquad &\Longrightarrow \\ (N+1)\,p \geq n + (N+1)\,\nu \qquad &\Longrightarrow \qquad (N+1)(p-\nu) \geq n\end{aligned}$$

In order to make a residual insensitive to faults, we need to set $\boldsymbol{z} = \boldsymbol{0}$ (*homogeneous specification*) and solve the homogeneous system $\boldsymbol{w}^\mathsf{T}\begin{bmatrix}O & H\end{bmatrix} = \boldsymbol{0}_{n+(N+1)\,\nu}^\mathsf{T}$. In this case the condition becomes:

$$\begin{aligned}\underline{\boldsymbol{z} = \boldsymbol{0}} \quad \#\mathsf{unknowns} \geq \#\mathsf{equations} + 1 \qquad &\Longrightarrow \\ (N+1)\,p \geq n + (N+1)\,\nu + 1 \quad &\Longrightarrow \quad (N+1)(p-\nu) \geq n+1\end{aligned}$$

In summary:

$$(N+1)\,p \geq \begin{cases} n + (N+1)\,\nu, & z \neq 0 \qquad (5.21a) \\ n + (N+1)\,\nu + 1, & z = 0 \qquad (5.21b)\end{cases}$$

These conditions can be fulfilled only if $\nu < p$.

Strictly input faults and reduced system

A fault f_i which doesn't directly affect the output y of the system, i.e. such that the i^{th} column of the matrix D_f is null, is called *strictly input fault*. If there are ν_I strictly input faults, we can define from D_f a new matrix D_f^*

without the ν_I null columns, obtaining from the matrix H a reduced matrix H^* of size $[(N+1)\,p] \times [(N+1)\,\nu - \nu_I]$, and from the system (5.18) a reduced system of $n + (N+1)\,\nu - \nu_I$ equations in $(N+1)\,p$ unknowns:

$$H^* = \begin{bmatrix} D_f & 0 & \dots & 0 \\ CB_f & D_f & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N-1}B_f & CA^{N-2}B_f & \dots & D_f^* \end{bmatrix}$$

$$\boldsymbol{w}^{\mathsf{T}} \begin{bmatrix} O & H^* \end{bmatrix} = \begin{bmatrix} \mathbf{0}_n^{\mathsf{T}} & \boldsymbol{z}^{*\mathsf{T}} \end{bmatrix} \qquad \text{(5.18 red.)}$$

where $\boldsymbol{z}^{*\mathsf{T}} = \begin{bmatrix} \boldsymbol{z}_N^{*\mathsf{T}} & \dots & \boldsymbol{z}_0^{*\mathsf{T}} \end{bmatrix}$ is a row vector of $(N+1)\,\nu - \nu_I$ parameters, with $\boldsymbol{z}_\kappa^* \in \mathbb{R}^\nu$ for $\kappa = 1 \dots N$, and $\boldsymbol{z}_0^* \in \mathbb{R}^{\nu - \nu_I}$. The complete specification $\boldsymbol{z}$ is obtained by adding a zero to $\boldsymbol{z}_0^*$ (in the correct position) for each strictly input fault.

In summary, conditions in section 5.2.3 become:

$$(N+1)\,p \geq \begin{cases} n + (N+1)\,\nu - \nu_I, & z \neq 0 \qquad \text{(5.22a)} \\ n + (N+1)\,\nu - \nu_I + 1, & z = 0 \qquad \text{(5.22b)} \end{cases}$$

These conditions can be fulfilled only if $\nu \leq p$.

Implementation of the minimum complexity solution

The *minimum complexity solution* is the one with the smallest horizon N that allows to solve the system (5.18) with respect to $\boldsymbol{w}$, and it is obtained when the number of equations is equal to the number of unknowns (+1):

$$(N+1)\,(p-\nu) = \begin{cases} n - \nu_I, & z \neq 0 \qquad \text{(5.23a)} \\ n - \nu_I + 1, & z = 0 \qquad \text{(5.23b)} \end{cases}$$

We consider the following cases:

- **non-homogeneous specification** ($\boldsymbol{z} \neq \mathbf{0}$)
  Minimum complexity solution: $(N+1)\,(p-\nu) = n - \nu_I$
  - $\nu = p$ (full specification): it requires $n = \nu_I$ and the condition (5.22) is satisfied for any $N \in \mathbb{N}$; this is a special case which implies $p = n$ and $\nu_I = \nu$, that is, $D_f = [0]$, therefore $\begin{bmatrix} O & H^* \end{bmatrix}$ would be singular for $N = 0$ and it must be $N \geq 1$.
  - $\nu = p-1$ (almost-full specification): the condition (5.22) is satisfied for $N \geq n - \nu_I - 1$; in the special case of $p = n$ and $\nu_I = \nu$, the condition is satisfied for any $N \in \mathbb{N}$, but $\begin{bmatrix} O & H^* \end{bmatrix}$ would be singular for $N = 0$ and it must be $N \geq 1$.

In summary, the minimum complexity solution is:

$$\boldsymbol{z} \neq \mathbf{0} \Rightarrow N = \begin{cases} 1 & p = n = \nu = \nu_I \\ 1 & p = n \text{ and } \nu = \nu_I = p - 1 \\ n - \nu_I - 1 & p < n \text{ and } \nu = p - 1 \end{cases} \tag{5.24}$$

In both cases, $\boldsymbol{w}^\mathsf{T} \coloneqq \begin{bmatrix} \mathbf{0}_n^\mathsf{T} & \boldsymbol{z}^{*\mathsf{T}} \end{bmatrix} \begin{bmatrix} O & H^* \end{bmatrix}^{-1}$

- **Homogeneous specification** ($\boldsymbol{z} = \mathbf{0}$)
  Minimum complexity solution: $(N+1)(p-\nu) = n - \nu_I + 1$
  - $\nu = p$ (full specification): this requires $\nu_I = n + 1$, which is not possible since $\nu_i \leq \nu = p \leq n$, therefore a full homogeneous specification cannot be given.
  - $\nu = p-1$ (almost-full specification): the condition (5.22) is satisfied for $N \geq n - \nu_I$.

  In summary, the minimum complexity solution is:

$$\boldsymbol{z} = \mathbf{0} \Rightarrow N = n - \nu_I \,, \quad \nu = p - 1 \tag{5.25}$$

  In the case of almost-full specification, in order to find an implementation $\boldsymbol{w}$ such that $\boldsymbol{w}^\mathsf{T} \begin{bmatrix} O & H^* \end{bmatrix} = \begin{bmatrix} \mathbf{0}_n^\mathsf{T} & \boldsymbol{z}^{*\mathsf{T}} \end{bmatrix}$ we have to assume the value of one of the elements of $\boldsymbol{w}$. If we set $(\boldsymbol{w}_{N-j'})_{k'} \coloneqq c$ we can rearrange eq. (5.18 red.) as:

$$\begin{bmatrix} \boldsymbol{w}_N^\mathsf{T} & \dots & (\boldsymbol{w}_{N-\kappa'}^\mathsf{T})_{\neg k'} & \dots & \boldsymbol{w}_N^\mathsf{T} \end{bmatrix} = -c \begin{bmatrix} O_{r'} & H^*_{r'} \end{bmatrix} \begin{bmatrix} O_{\neg r'} & H^*_{\neg r'} \end{bmatrix}^{-1}$$

  where $(\boldsymbol{w}_{N-\kappa'}^\mathsf{T})_{\neg k'}$ is $\boldsymbol{w}_{N-j'}^\mathsf{T}$ without its $(k')^{th}$ element, $O_{r'}$ and $H^*_{r'}$ are the $(r')^{th}$ rows[4] of O and H^*, and $\tilde{O}_{\neg r'}$ and $\tilde{H}^*_{\neg r'}$ are the matrices O and H^* without their $(r')^{th}$ row, and $r' = \kappa' N + k'$.

Insensitivity to faults

In order to make a residual insensitive to a specific subset $\tilde{\mathcal{F}}$ of $\tilde{\nu}$ faults, we can consider the subsystem $(A,\, B_u, \tilde{B}_f,\, C,\, D_u, \tilde{D}_f)$ subject to those faults only, and find an implementation $\boldsymbol{w}$ for the homogeneous specification $\tilde{\boldsymbol{z}} = 0$:

1. we define the sub-matrices $\tilde{B}_f$ and $\tilde{D}_f$ composed of the columns $B_{f_{*,i}}$ and $D_{f_{*,i}}$ such that $f_i \in \mathcal{F}$
2. we define the corresponding matrix $\tilde{H}$ (or $\tilde{H}^*$ if $\tilde{D}_f$ has $\tilde{\rho}$ null columns)
3. we set $\tilde{\boldsymbol{z}} \coloneqq \mathbf{0}^\mathsf{T}_{(N+1)\tilde{\nu}-\tilde{\rho}}$ (homogeneous specification) and we find the corresponding residual implementation $\boldsymbol{w}$ by solving the system $\boldsymbol{w}^\mathsf{T} \begin{bmatrix} O & \tilde{H}^* \end{bmatrix} = \begin{bmatrix} \mathbf{0}_n^\mathsf{T} & \mathbf{0}^\mathsf{T}_{(N+1)\tilde{\nu}-\tilde{\rho}} \end{bmatrix}$

[4] according to the row numbering of the full matrix H

4. we compute the (non-homogeneous) specification $\boldsymbol{z}^{\mathsf{T}} \coloneqq \boldsymbol{w}^{\mathsf{T}} H$ for the full set of faults $\mathcal{F}$ (where $z_{i+\nu j} = (\boldsymbol{z}_j)_i = 0\,,\ \forall\, i : f_i \in \tilde{\mathcal{F}}\,,\ j = 0 \dots N$)
5. we check *a posteriori* that the residual $r(k) = \boldsymbol{w}^{\mathsf{T}} H\, F(k) = \boldsymbol{z}^{\mathsf{T}} F(k)$ is sensitive to the other $\nu - \tilde{\nu}$ faults $f_i \notin \tilde{\mathcal{F}}$

Example (5.2.1)
Consider the following system, with $p = \nu = n = 2$:

$$A = \begin{bmatrix} 0.8 & 0 \\ 1 & 0.5 \end{bmatrix} \qquad B_u = \begin{bmatrix} 2 & 0 \\ 1 & 3 \end{bmatrix} \qquad B_f = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix}$$
$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad D_u = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad D_f = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$$

We want to find a residual which is sensitive to f_2 only (full non-homogeneous specification). The matrix D_f has one null column, therefore $\nu_I = 1$ and the condition (5.22a) *can't be fulfilled. In order to make the residuals insensitive to f_1, we consider the subsystem with this fault only, and we search an implementation for the homogeneous specification $\tilde{\boldsymbol{z}} = \boldsymbol{0}$:*

$$\tilde{B}_f = \begin{bmatrix} B_{f\,*,1} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \tilde{D}_f = \begin{bmatrix} D_{f\,*,1} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \tilde{\nu} = 1 \qquad \tilde{\nu}_I = 0$$

Since $p = \nu - 1$, the specification is almost-full. According to eq. (5.23)*, the minimum complexity solution is the one with $N = n - \tilde{\nu}_I = 2$. The matrices $\tilde{H}$ and O are:*

$$O = \begin{bmatrix} C \\ CA \\ CA^2 \end{bmatrix}\,, \quad \tilde{H} = \begin{bmatrix} \tilde{D}_f & 0 & 0 \\ C\tilde{B}_f & \tilde{D}_f & 0 \\ CA\tilde{B}_f & C\tilde{B}_f & \tilde{D}_f \end{bmatrix}$$

$$O = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0.8 & 0 \\ 1 & 0.5 \\ 0.64 & 0 \\ 1.3 & 0.25 \end{bmatrix}\,, \quad \tilde{H} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0.8 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

We set $(\boldsymbol{w}_2)_1 \coloneqq 1$ and we compute:

$$\begin{bmatrix} (\boldsymbol{w}_2)_2 & (\boldsymbol{w}_1)_1 & (\boldsymbol{w}_1)_2 & (\boldsymbol{w}_0)_1 & (\boldsymbol{w}_0)_2 \end{bmatrix} \coloneqq$$
$$= -1 \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0.8 & 0 & 1 & 1 & 0 \\ 1 & 0.5 & 0 & 1 & 0 \\ 0.64 & 0 & 0.8 & 1 & 1 \\ 1.3 & 0.25 & 1 & 0 & 1 \end{bmatrix}^{-1} =$$
$$\cong \begin{bmatrix} 0.071 & -0.929 & 0.214 & 0.714 & -0.714 \end{bmatrix}$$

The implementation of the homogeneous specification is:

$$\boldsymbol{w}^{\mathsf{T}} \cong \begin{bmatrix} 1 & 0.071 & -0.929 & 0.214 & 0.714 & -0.714 \end{bmatrix}$$

The (reduced) matrix H^* for the original system is:

$$H^* = \begin{bmatrix} D_f & 0 & 0 \\ CB_f & D_f & 0 \\ CAB_f & CB_f & D_f^* \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 0 & 3 & 1 & 0 & 0 \\ 0.8 & 1.6 & 1 & 2 & 1 \\ 1 & 3.5 & 0 & 3 & 1 \end{bmatrix}$$

The specification $\boldsymbol{z}$ for the full set of faults is:

$$\boldsymbol{z}^{*\mathsf{T}} = \begin{bmatrix} \boldsymbol{z}_2^{*\mathsf{T}} & \boldsymbol{z}_1^{*\mathsf{T}} & (\boldsymbol{z}_0^*)_1 \end{bmatrix}^{\mathsf{T}} \coloneqq \boldsymbol{w}^{\mathsf{T}} H^* = \begin{bmatrix} 0 & -2.57 & 0 & -0.71 & 0 \end{bmatrix}$$

with $(\boldsymbol{z}_0)_2 = 0$ as f_2 is a strictly-input fault and it does not affect the output at the current time instant. The same result can be obtained from the:

$$\boldsymbol{z}^{\mathsf{T}} \coloneqq \boldsymbol{w}^{\mathsf{T}} H = \begin{bmatrix} 0 & -2.57 & 0 & -0.71 & 0 & 0 \end{bmatrix}$$

Finally, we check if the residual is sensitive to f_2:

$$r(k) = \boldsymbol{z}^{\mathsf{T}} F(k) = \begin{bmatrix} 0 & -2.57 & 0 & -0.71 & 0 & 0 \end{bmatrix} \begin{bmatrix} f_1(k-2) \\ f_2(k-2) \\ f_1(k-1) \\ f_2(k-1) \\ f_1(k) \\ f_2(k) \end{bmatrix} =$$

$$= -2.57\, f_2(k-2) - 0.71\, f_2(k-1)$$

The equation of the residual can be rewritten as:

$$r(k) = Z(z)\, f(k) = Z_2\, f_2(k) = -(2.57\, z^{-2} + 0.71\, z^{-1})\, f_2(k)$$
$$Z(z) = \begin{bmatrix} Z_1(z) & Z_2(z) \end{bmatrix} = \begin{bmatrix} 0 & -(2.57\, z^{-2} + 0.71\, z^{-1}) \end{bmatrix}$$

5.2.4 Linear systems in transfer function form

We consider the transfer function representation for a discrete-time dynamical system:

$$y(k) = M(z)\, u(k) + S(z)\, f(k) \tag{5.26}$$

from which we obtain:

$$y(k) - M(z)\, u(k) = S(z)\, f(k) \tag{5.27}$$

Transfer functions can be obtained from sate-space representation as:

$$M(z) = C(zI - A)^{-1} B_u + D_u \ , \quad S(z) = C(zI - A)^{-1} B_f + D_f$$

Since $y(k) - M(z)\, u(k) = 0$ for $f(k) = 0$ we can define the primary and secondary residuals as:

$$\boldsymbol{o}(k) = y(k) - M(z)\, u(k) = S(z)\, f(k) \tag{5.28a, 5.28b}$$

$$r(k) = \boldsymbol{w}^{\mathsf{T}} \cdot [y(k) - M(z)\, u(k)] = \tag{5.29a}$$
$$= \boldsymbol{w}^{\mathsf{T}} \cdot [S(z)\, f(k)] \ \in \ \mathbb{R} \tag{5.29b}$$

where $\boldsymbol{w}^{\mathsf{T}} \in \mathbb{R}^p$. The definition of secondary residuals in eq. (5.29) is identical to those given in eqs. (5.5) and (5.6) for the computational and internal forms of the generic residual generator.

The naïve implementation $\boldsymbol{w}(z) = Z(z)\, S(z)^{-1}$, which requires $p = \nu$, is generally wrong, because:

- it may be unstable (the residual would diverge even with no faults);
- it may be non-casual (not realizable)

When $p = \nu$ we can set:

$$\boldsymbol{w}(z) = Z(z)\, T(z)\, S(z)^{-1}$$
$$T(z) \triangleq \frac{\pi(z)}{z^{\gamma}} \ \mathrm{diag}\left\{ \frac{1}{z^{\alpha_1}}, \ldots, \frac{1}{z^{\alpha_p}} \right\} , \quad \pi(z) = \prod_{i=1}^{\gamma} (z - a_i)$$

where $a_1, ..., a_\gamma$ are the unstable invariant zeros of $S(z)$, and α_i is the maximum of the relative degrees of the transfer functions in the i^{th} row of $S(z)^{-1}$.

Multiplicative faults

In order to consider parametric faults, we use the parametric form of the transfer function:

$$y(k) = M(z; \boldsymbol{\vartheta})\, u(k) \ , \quad \boldsymbol{\vartheta} = \boldsymbol{\vartheta}^{\circ} + \delta\boldsymbol{\vartheta} \tag{5.30}$$

This equation can be expanded as:

$$y(k) \cong M(z; \boldsymbol{\vartheta}^{\circ})\, u(k) + \sum_{j=1}^{\nu} \boldsymbol{\delta m_j}\, \delta\vartheta_j(k)\, u(k) \ , \quad \boldsymbol{\delta m_j} \triangleq \left. \frac{\partial M(z; \boldsymbol{\vartheta})}{\partial \vartheta_j} \right|_{\boldsymbol{\vartheta}^{\circ}}$$

which can be rewritten as:

$$\underbrace{y(k) \cong M^{\circ}(z)\, u(k)}_{\text{nominal model}} + N(z; k)\, \delta\boldsymbol{\vartheta}(k)$$
$$M^{\circ}(z) \triangleq M(z; \boldsymbol{\vartheta}^{\circ}) \ , \quad N(z; k) \triangleq \left[\boldsymbol{\delta m_1}(z)\, u(k) \quad \ldots \quad \boldsymbol{\delta m_\nu}(z)\, u(k) \right]$$

where $M^\circ(z)$ is the transfer function of the healthy system.

Primary and secondary residuals are defined as:

$$\begin{aligned} \boldsymbol{o}(k) &= y(k) - M^\circ(z)\, u(k) \cong N(z;k)\, \delta\boldsymbol{\vartheta}(k) \\ r(z) &= W(z;k) \cdot [y(k) - M^\circ(z)\, u(k)] \cong \\ &\cong W(z;k)\, N(z;k)\, \delta\boldsymbol{\vartheta}(k) = Z(z)\, \delta\boldsymbol{\vartheta}(k) \end{aligned}$$

Since the matrix $N(z)$ is time-varying, in order to have the same specification $Z(z)$ the implementation $W(z;k)$ should be computed at every time instant.

The use of parity equations with multiplicative faults is limited by the fact that different parametric faults δ_i affecting the same row of $M(z)$ can be detected but not isolated, as their effect can't be distinguished.

5.2.5 State observers

Consider again the state-space representation (5.11):

$$\begin{cases} x(k+1) = A\, x(k) + B_u\, u(k) + B_f\, f(k) & \text{(5.11a)} \\ y(k) = C\, x(k) + D_u\, u(k) + D_f\, f(k) & \text{(5.11b)} \end{cases}$$

The state observer $\hat{x}(\,\cdot\,)$ is defined by:

$$\begin{cases} \hat{x}(k+1) = A\, \hat{x}(k) + B_u\, u(k) + L \cdot [y(k) - \hat{y}(k)] & \text{(5.31a)} \\ \hat{y}(k) = C\, \hat{x}(k) + D_u\, u(k) & \text{(5.31b)} \end{cases}$$

The estimation errors are defined as:

$$\begin{cases} e_x(k) = x(k) - \hat{x}(k) & \text{(5.32a)} \\ e_y(k) = y(k) - \hat{y}(k) & \text{(5.32b)} \end{cases}$$

and they evolve according to the dynamical system:

$$\begin{cases} e_x(k+1) = (A - LC)\, e_x(k) + (B_f - LD_f)\, f(k) & \text{(5.33a)} \\ e_y(k) = C\, e_x(k) + D_f\, f(k) & \text{(5.33b)} \end{cases}$$

where eq. (5.33a) is obtained by subtracting eq. (5.31a) from eq. (5.11a), after replacing $y(k)$ and $\hat{y}(k)$ with eq. (5.11b) and eq. (5.31b), and eq. (5.33b) is obtained by subtracting eq. (5.31b) from eq. (5.11b).

Estimation error and primary residuals

If we choose L such that $(A - LC)$ is asymptotically stable, then:

$$f = 0 \quad \Longrightarrow \quad e_x(k)\,,\ e_y(k) \to 0\,, \quad \text{for } k \to \infty$$

therefore the estimation error $e_y(k)$ can be used as a residual for the identification of faults. A suitable choice of L can guarantee fault identification and isolation properties.

Computational form The estimation error $e_y(k)$ can be expressed as function of inputs $u(k)$ and outputs $y(k)$. Equation (5.31a) can be rewritten as:

$$\hat{x}(k) = (zI - A)^{-1} B_u \, u(k) + (zI - A)^{-1} L \, e_y(k) \tag{5.34}$$

Equation (5.32b), combined with eq. (5.31b) and eq. (5.34), gives:

$$\boldsymbol{o}(k) = [I - C \, (zI - A)^{-1} L] \, e_y(k) = y(k) - M(z) \, u(k) \tag{5.35}$$

where $M(z) \triangleq [C \, (zI - A)^{-1} B_u + D_u]$. This equation is identical to the primary residuals as defined in eq. (5.14a).

If we apply the matrix inversion lemma[5] to $[I - C \, (zI - A)^{-1} L]$ we can write:

$$e_y(k) = [I - C \, (zI - A + LC)^{-1} L] \cdot (y(k) - M(z) \, u(k)) \tag{5.36}$$

This is a residual in computational form, as it depends on inputs $u(k)$ and outputs $y(k)$ only.

Internal form Equation (5.33a) can be rewritten as:

$$e_x(k) = (zI - A + LC)^{-1} \, (B_f - LD_f) \, f(k)$$

By substituting this in eq. (5.33b) we obtain:

$$e_y(k) = [C \, (zI - A + LC)^{-1} \, (B_f - LD_f) + D_f] \cdot f(k) \tag{5.37}$$

This is a residual in internal form, as it depends on faults $f(k)$ only.

Secondary residuals

If we multiply the primary residuals (5.36) by a matrix $H_{q \times p}$ we get:

$$\boldsymbol{r}(k) = W(z) \cdot [y(k) - M(z) \, u(k)] \in \mathbb{R}^q \tag{5.38}$$

where $W(z) \triangleq H \cdot [I - C \, (zI - A + LC)^{-1} L]$.

Choosing of the transfer function matrix $W(z)$ is equivalent to finding a pair of matrices (L, H) which guarantee the asymptotic stability of the observer and the required isolation properties for the secondary residual in internal form:

$$\boldsymbol{r}(k) = H \cdot [C \, (zI - A + LC)^{-1} \, (B_f - LD_f) + D_f] \cdot f(k) \tag{5.39}$$

[5] Woodbury matrix identity: $(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$

Part III

Use of the Models in Predictive Maintenance

Chapter 6

Fault implementation with Modelica

6.1 Introduction to Modelica language

Modelica is an object-oriented programming language focused on first-principles models (FPMs).

A dynamic system is described using a *declarative (acausal)* notation, that is, through its physical equations, which can be written in any order. This makes Modelica different from procedural programming languages, which use an *imperative (casual)* notation, that is, a set of assignments which are executed in a specific order.

The main advantages of a declarative notation are that:

- there is no need to write equations in closed-form (as assignments) and in a specific order;
- there is no need to identify inputs and outputs among all variables;
- the model structure is easy to understand.

Modelica classes According to the principles of object-oriented programming (OOP), Modelica language is based on *classes* and *objects* (instances of a class). A Modelica class has the following structure:

```
class newClass
  // LIST OF DECLARATIONS
  // Variables, parameters, constants, and objects
equation
  // LIST OF EQUATIONS
initial equation
  // LIST OF INITIAL EQUATIONS (used to initialize the
      simulation)
end newClass;
```

Modelica also defines seven specialized classes: **model**, **connector**, **record**, **block**, **type**, **package**, and **function**. While **model** and

class keywords are interchangeable, classes defined with other keywords can have restrictions (e.g. a **connector** class cannot contain equations) or enhancements.

A class can represent a component or system model, and can be simulated. In order for there to be a unique solution, a model must be balanced, that is, the number of variables and equations of a model (including the inherited ones) must be the same. This applies to any non-specialized and non-partial[1] class.

The following class allows to simulate the differential algebraic equation $\dot{x}(t) = 1 - x$, with $x(0) = 0$:

```
class FirstOrderDAE
  Real x;
initial equation
  x = 0;
equation
  der(x) = 1-x;
end FirstOrderDAE;
```

Components and interfaces Modelica allows to create *components* that can be connected to each others through physical *interfaces*. Basic components (such as electrical *one-ports*) are defined by a **model** class, and physical interfaces (such as electrical pins) are defined by a **connector** class.

More components can be connected together in a new **model** class, containing an instance of each component and a **connect** statement for each connection between any two interfaces of the components[2] . This allows to create complex components, subsystems, and systems.

Modeling and simulation environments like *OpenModelica Connection Editor* and *Dymola* allow to create two kinds of graphical views for each class:

- an *icon*, which represents the class (in the library browser) and its instances (in the diagram of another class), including its interfaces;
- a *diagram*, which allows to build a model by placing components from the library and drawing connections between their interfaces.

Each time a component is placed on a model diagram a new object is declared, and each time two interfaces are connected a new **connect** statement is created.

Flow and effort variables A connector can contain two kinds of variables: *effort* variables (which are the default kind) and *flow* variables (defined through the **flow** keyword). The connection of two interfaces generates an *equality* equation for each pair of effort variables, and a *balance* equation for each pair of flow variables.

[1] see paragraph *"Partial classes"* on page 45

[2] see paragraph *"Connect equations"* on page 46

For example, an electrical pin interface is a **connector** class that contains two variables: the voltage (effort) and the current (flow):

```
connector Pin
  Real v      "voltage (effort variable)";
  flow Real i "current (flow variable)";
end Pin;
```

A connector can be instantiated as an object inside a component model, and its variables can be accessed using the dot notation:

```
model Resistor
  Pin p "positive pin";
  Pin n "negative pin";
  Real v "voltage";
  Real i "current";
  parameter Real R "Resistance";
equation
  v = p.v - n.v;
  0 = p.i + n.i;
  i = p.i;
  R*i = v;  // constitutive equation
end Resistor;
```

If an interface is not connected, when the model is instantiated all flow variables are set to zero by adding a new equation. In the previous example, the current through the pin interfaces must be zero, therefore the equations `p.i = 0.0` and `p.i = 0.0` are automatically created.

Partial classes When more components share some common features, it is possible to gather them together in a **partial** class. The **partial** keyword defines a class as incomplete, and therefore it cannot be instantiated nor simulated. A partial class can only be inherited by another class through the **extends** keyword.

For example, all electric one-port components share the following features, according to the passive sign convention:

- they have a positive and a negative electrical pin;
- there is a voltage drop from the negative to the positive pin;
- the current flowing into the the positive pin is equal to the current flowing out from the negative pin;

These features can be modeled as a partial model `OnePort`, then for each one-port component it is sufficient to create a class that extends `OnePort` and defines the missing equation, that is, the constitutive equations between voltage and current:

```
partial model OnePort
  Pin p, n;
  Real v, i;
equation
  v = p.v - n.v;
  0 = p.i + n.i;
  i = p.i;
```

```
end OnePort;

model Resistor
  extends OnePort;
  parameter Real R "Resistance";
equation
  R*i = v;
end Resistor;

model Capacitor
  extends OnePort;
  parameter Real C "Capacitance";
equation
  C*der(v) = i;
end Capacitor;
```

When the `Resistor` class is instantiated, all the inherited equations and declarations are included, thus making the model balanced (6 equations and 6 variables):

```
model Resistor
  Real p.v "voltage (effort variable)";
  Real p.i "current (flow variable)";
  Real n.v "voltage (effort variable)";
  Real n.i "current (flow variable)";
  Real v "voltage";
  Real i "current";
  parameter Real R "Resistance";
equation
  R * i = v;
  v = p.v - n.v;      // inherited from OnePort
  0.0 = p.i + n.i;    //
  i = p.i;            //
  p.i = 0.0;    // automatically created
  n.i = 0.0;    //
end Resistor;
```

Connect equations The **connect** statement takes two compatible connector instances as argument and creates equality and balance equations between any pair of effort and flow variables.

The following example shows how to connect a resistor and a capacitor:

```
model RC
  Pin p "positive pin";
  Pin n "negative pin";
  Resistor R1;
  Capacitor C1;
equation
  connect(R1.n, C1.p);
end RC;
```

The **connect** statement creates the following equations:

```
C1.p.v = R1.n.v;         // equality equation
R1.n.i + C1.p.i = 0.0;   // balance equation
```

In addition, the following balance equations are created for unconnected pins:

```
R1.p.i = 0.0;
C1.n.i = 0.0;
```

In order to use the previous model as a component, interfaces `RC.p` and `RC.n` should be added, as it would be unpractical to access to `RC.R1.p` and `RC.C1.n`. It is also possible to define the voltage drop across and the current through the new component:

```
model RC
  Pin n, p;
  Real v, i;
  Resistor R1;
  Capacitor C1;
equation
  v = p.v - n.v;
  i = p.i;
  connect(R1.p, p);
  connect(R1.n, C1.p);
  connect(C1.n, n);
end RC;
```

Conditional models Modelica allows to write models that behave differently depending on the value of a conditional equation. This can be done using an *if-then-else* statement.

Unlike other programming languages, the *if* statement is subject to some restrictions, in order to guarantee that the number of equations doesn't change: each branch of the statement must compute the same variables, and there must always be the **else** alternative.

```
model ConditionalModel
  // set of m+n variables
equation
  // set of m equations
  if [conditional expression] then
    // set of n equations
  elseif [conditional expression] then
    // set of n equations
  else
    // set of n equations
  end if;
end ConditionalModel;
```

Implicit connections Implicit connections allow to share the value of a variable in a class (the *superclass*) with other objects (instances of *subclasses*) that are declared in the same class. The variable must appear in the superclass with the **inner** prefix, and in the subclasses with the **outer** prefix. The value of the variable is determined by the superclass, either as a declaration or as the result of the equations.

```
model Pipe;
  outer Temperature EnvironmentTemp "reference value";
  // ...
equation
  // ...
end Pipe;

model Tank;
  outer Temperature EnvironmentTemp "reference value";
  // ...
equation
  // ...
end Tank;

model Plant
  inner Temperature EnvironmentTemp "shared values";
  Tank T1, T2, T3;
  Pipe P1, P2;
equation
  // ...
end Plant;
```

I/O variables and casual blocks Modelica allows to define *input/output* variables according to the casual "block diagram" paradigm. These variables are defined through the **input** and **output** prefixes. Input variables can be only connected to output variables, and vice versa.

A **connector** can contain both input and output variables, together with flow and effort variables. A **block** class has the restriction that a **connect** statement can only contain one input and one output variable.

6.2 Fault implementation in Modelica

The Modelica Association develops the free Modelica Standard Library (MSL), which contains hundreds of component models, none of which considers the faulty behavior. The problem of fault implementation in Modelica has been widely studied in the last few years, but a standardized implementation doesn't exists yet.

Fault characteristics A fault can be either in active or inactive *state*, and some kinds of fault may have different levels of *severity*. The activation of a fault may require a `Boolean` variable, or an `Integer` variable to choose among different fault behaviors; the specification of the level of severity require a `Real` variable. It is possible to use **parameter** variables to set fault state and severity before the simulation, and **input** connectors or **outer** variables to set them during the simulation.

In view of this, existing approaches to fault implementation focuses on three aspects:

- **fault classification**
- **fault architecture** (concerning the way fault are modeled)
- **fault activation** (concerning the way the system choose which faults are active during the simulation)

6.2.1 State of the art

Existing approaches aim to simulate one or more faulty behaviors in addition to the nominal behavior, but they use different architectures for the fault modelization and different methods to activate fault.

The FAME framework De Kleer et al. [dKle+13] developed the *FAME framework*, a semiautomatic way to extend the Modelica Standard Library (MSL) models in order to allow the simulation of faulty behaviors. For each basic component of the library which is susceptible to faults they created a new *fault-augmented* model that describes the behavior of each fault mode, as well as the nominal behavior from the original model. Basic components are those without internal `connect` statement; fault augmentation for non-basic components is made by inserting faults in the basic components they include.

When a component is susceptible to faults, a new set of equations is written for each fault mode, and the new model of the component allows to select the operating mode – among the nominal one and the fault ones – through a conditional statement. An `Integer` parameter allows to choose the *operating mode* (which cannot change during the simulation), and a `Real` variable interface allows to select the *fault amount*, where 0 is the absence of the fault and 1 is its maximum severity.

The partial automation of fault implementation relies on the identification of standard power interfaces, which contain an effort and a flow variable. In the case of faults that affect the power flow between two or more components, the fault augmentation process adds a power dissipation component.

The Fault Triggering library Van der Linden [vdLin14] proposed a new standard for the implementation of faults in Modelica. The *Fault Triggering* library introduces a set of standardized *"fault-output"* causal blocks[3] which allow to create component models that include optional faults. These blocks indicates if the fault is active and, depending on the type of fault, its intensity. Care has been taken to make sure that all the types of faults can be modeled through these blocks, and that the nominal system can be simulated as well.

The library contains function `createFaultPackage`, which detects all the fault blocks in a model and creates a "model wrapper" package that contains the model of the system and a *fault trigger block*, which is used to activate faults.

[3] see paragraph *"I/O variables and casual blocks"* on page 48

The Fault library Gundermann et al. [Gun+19] developed the *Fault Library*, which is is based on both the fault augmented models of the *FAME library* [dKle+13] and the standardized blockset of the *Fault Triggering* library [vdLin14]. This library relies on both fault augmented models, like the *FAME framework*, and additional blocks for connector and bridge faults (see paragraph *"Within the fault library"*, page 51) .

Comparison of the different approaches While the *FAME Framework* models all kinds of faults through the fault augmentation of the components, the *Fault Library* uses a mixed approach, in which the fault augmentation is used only when the fault affects a component internally, and additional components are used when the fault affects the connection between two components.

The *Fault Triggering* library introduces blocks whose output specify the fault state and intensity, but de Kleer et al. don't define a standard for fault implementation, as the purpose of the library is centralizing the fault activation rather than simplifying the fault modelization.

Drawbacks The aforementioned approaches present many drawbacks. First of all, none of them works with *OpenModelica*. The *Fault Triggering* library [vdLin14] requires *Dymola*, and the *Fault Library* [Gun+19] requires *SimulationX*, which are both commercial environments. Additionally, the *Fault Library* is no longer mantained and the last version (0.6.6) is not compatible with the current version of *Dymola*. The *FAME Framework* [dKle+13] works with *JModelica* and relies on other external tools.

Another drawback is that the *FAME framework* and the *Fault Library* approaches required to create from scratch the fault-augmented components, although the process

6.2.2 Fault classification and architecture

A proper classification of faults is a critical aspect in view of their implementation in a Modelica model. A first classification of faults and disturbances has been given in section 5.1, but it is functional to block schemes, which represent casual models.

A fault may affect an internal parameter of a component, a constitutive relation (that is, a relation between two physical quantities), the whole behavior of the component, or the connection between more components. Different kinds of faults may require to be implemented in completely different ways, and it is fundamental to classify them in view of this.

Within the FAME framework De Kleer et al. [dKle+13] divided the faults in three classes:

- **catastrophic faults**, which change the behavior of a component into something different (e.g. ruptures);
- **power flow faults**, which affect the power flow to or from a component (e.g. short circuits and leakages);
- **parametric faults**, which reflect the shifting of a supposedly fixed physical quantity that describes the component (e.g. wear and fatigue).

This classification is functional to the automatic augmentation of the MSL components. While the modelization of *catastrophic faults* require a physical understanding of the system, the modelization of *parametric faults* is trivial. Regarding *power flow faults*, the way components are implemented in Modelica language makes easy to detect power flow interfaces (which usually are connectors containing both effort and flow variables), and this allows to model *power dissipation* (e.g. fluid leakages, loose mechanical joints or bad electrical contact) or unwanted *power bridges* (e.g. parasitic capacitance between two electrical connectors) in a semiautomatic way. A power dissipation is modeled through the insertion of a damper element, which is attached to each interested external interface (connector) of the component, and a power bridge is modeled through the insertion of an element which connects each interested pair of connectors of the component.

Within the FaultTriggering library Van der Linden [vdLin14] defines three types of faults:

- **"on-off" faults**, which have only one discrete faulty state (in addition to the nominal state) and can be modeled through a `Boolean` variable;
- **case faults**, which have more than one discrete faulty state (in addition to the nominal state) and can be modeled through an `Integer` variable;
- **continuous faults**, whose entity is modeled through a `Real` variable.

From the *simulation* point of view, the faults are classified as:

- **parameter faults**, which have a low time constant with respect to the simulation time, so that it's not necessary to model the fault transients;
- **variable faults**, which have a high time constant with respect to the simulation time, so that it is necessary to model the fault transients (this is often the main purpose of the simulation).

The combination of these two classifications leads to six different types of faults.

Within the fault library Gundermann et al. [Gun+19] defines two types of faults:

- **continuous faults**, whose *intensity* is modeled through a `Real` variable, which is `0` when the fault is not present (nominal behavior) and `1` when the fault is maximum.
- **discrete faults**, whose *state* is modeled through a `Boolean` variable, which is true if the fault is active.

From the *modelization* point of view, the faults are classified as:

- **connector faults**, which can be:
  - components with two interfaces that can cut an existing connection (e.g. open circuits, pipe obstructions, ...);
  - components with one interface that can be added to an existing connection (e.g. additional frictions, fluid leakages, ...);
- **bridge faults**, which can be inserted between existing connections (e.g. short circuits);
- **fault augmented models** with parametric faults

6.2.3 A new classification for a hybrid fault architecture

The classification introduced in the *Fault library* reflects the mixed approach to fault implementation, but it is incomplete as it does not consider the case in which the dynamic equations of the system change. The classification introduced in the *FAME framework* also includes that missing case and, even though it is functional to the fault augmentation approach, it can be easily adapted to the mixed approach.

FAME	Fault Library	FaultTriggering library
Power flow faults	Connector/bridge faults	Continuous, on-off
Catastrophic faults	–	Case, on-off
Parametric faults	Fault augmented models	Continuous

The following classification is an attempt to combine the aforementioned ones in view of a hybrid fault architecture that makes use of both additional connectors and fault augmented components:

- **Power flow faults.** These faults can be modeled through additional connectors which depend on a `Real` variable (the fault is maximum if the variable is 1) or a `Boolean` variable (the fault is active if the variable is true).
  - **Interposed connectors (two interfaces).** They cut an existing connection.
  - **Juxtaposed connectors (one interface).** They can be added to an existing connection.
  - **Bridge connectors (two interfaces).** They can be inserted between existing connections.
- **Internal faults.** These faults can be modeled through fault augmented components.
  - **Parametric faults.** A fixed parameter becomes a `Real` variable
  - **Behavioral faults.** The system can have one or more faulty behaviors, in addition to the nominal behavior, corresponding to different sets of dynamic equations; the choice of the proper

behavior may depend on a `Boolean` variable if there is only one faulty behavior (which is `false` in the nominal case) or an `Integer` variable (which is `0` in the nominal case),

Implementation of connector faults Depending on the application domain (electric, hydraulic, ...) a connector fault may be continuous (`0` if inactive, `1` if maximum) or discrete (`false` if inactive, `true` if active). An example of discrete connector fault is a welding on an electric board, which is a short circuit that becomes an open circuit when broken. An example of continuous connector fault is a mechanical transmission belt that may work at maximum efficiency (the fault intensity is `0`), get too loose (the fault intensity is between `0` and `1`) or break (the fault intensity is `1`).

Implementation of parametric faults The main issue with parametric faults is the impossibility to express their intensity as a percent scale, as the parameter may vary in a non-finite range. A solution could be to express the difference δp between a parameter p and its nominal value p_n (which could be assumed to be its expected value μ_p), with respect to the parameter variance σ_p^2, that is, as $\varepsilon_p \triangleq \delta p/\sigma_p^2$. If $|\varepsilon_p| \leq 2 \div 3$ the parameter can be assumed to be at its nominal value, otherwise a fault may have occurred. The acceptable range of ε_p may vary according to the parameter and the specific application.

6.3 The role of probability

6.3.1 Failure mechanisms in the FAME framework

De Kleer et al. also modeled some parametric failure mechanisms, including wear, fatigue, corrosion, and stress rupture. In the case of progressive damage mechanisms, the *FAME framework* allows to compute the PDF of the amount of damage for a determined set of parameters, such as the component age and geometry; given the PDFs for a limited set of parameter values it is possible to use interpolation in order to get the PDFs for other values. For a sudden rupture mechanism it is possible to compute the PDF of the age at which the rupture happens.

Possible uses for Predictive Maintenance The paper suggests some possible uses of these failure mechanisms for both design and diagnostic purposes:

- determining whether a system requirement is met, given the components age or a set of faults.
- determining the RUL and the MTTF of a compoment;
- performing FMEA and FMECA.

In view of predictive maintenance, the following possibilities should be investigated:

- Given the age of each component, determine which components are most likely to fail next, then study the behavior of the system when each of those components fail.
- Given the PDFs, determine which components are most likely to fail simultaneously, then study the behavior of the system in the case of simultaneous failure.
- Given some performance requirements on the system, determine the maximum acceptable wear of each component.
- ...

6.3.2 Using Bayesian inference for diagnostic purposes

Minhas et al. [Min+14] propose a diagnosis framework based on the *Bayesian inference* and the augmented models generated by the *FAME framework* [dKle+13]. The main idea behind the diagnosis framework is to assess through Bayesian inference the likelihood of a (nominal or faulty) model given the observed data. Numerical simulations of fault augmented Modelica models are used to obtain a statistical model of the system for all of its operating modes through an Approximate Bayesian Computation (ABC) approach.

Prior and posterior probability Let $M(\vartheta)$ be a model characterized by a parameter ϑ. This can be considered as the realization of a random variable[4] Θ, whose probability density distribution[5] $f(\vartheta)$, called *prior distribution*, is known.

Any datum x_i generated from the model (such as the value of a physical variable in a specified time instant, its settling time or its average value) can be considered as the outcome of an experiment (the simulation), that is, the realization of a random variable X_i. Assuming that the value of the parameter is ϑ, we define the joint probability density function $f(x_1, \ldots, x_n|\vartheta)$ as the *likelihood* of $X_1, \ldots, X_n$. Conversely, if the observed data are $X_1 = x_1, \ldots, X_n = x_n$ then the probability density function of ϑ becomes [Ros04]:

$$f(\vartheta|x_1, \ldots, x_n) = \frac{f(\vartheta, x_1, \ldots, x_n)}{f(x_1, \ldots, x_n)} = \frac{f(\vartheta)\, f(x_1, \ldots, x_n|\vartheta)}{\int f(x_1, \ldots, x_n|\vartheta)\, f(\vartheta)\, d\vartheta}$$

The conditional density function $f(\vartheta|x_1, \ldots, x_n)$ is called *posterior density function*, and can be considered as the prior density function which has been updated with a new evidence, that is, the observed data through a process that is called *Bayesian updating*.

[4] see definition A.1.2
[5] see definition A.1.4

Bayes estimator Since the best estimate of a random variable is its mean (see definition A.1.6), the best estimate of ϑ given the data $X_1 = x_1, \ldots, X_n = x_n$, denoted with $E[\vartheta|X_1, \ldots, X_n]$, is the mean of its posterior density function:

$$E[\vartheta|X_1, \ldots, X_n] = \int \vartheta f(\vartheta|x_1, \ldots, x_n)\, d\vartheta$$

Bayes estimator can be useful in order to estimate the value of those parameters which are related to faults.

Bayes factor Bayes' theorem (A.3.1) allows to get the posterior probability of a specific parameter value ϑ given an observation $x_1, \ldots, x_n$:

$$P(\vartheta|X_1, \ldots, X_n) = \frac{P(X_1, \ldots, X_n|\vartheta)\, P(\vartheta)}{P(X_1, \ldots, X_n)}$$

Bayes factor B_ij is defined as the ratio between the posterior probabilities of two different parameter values ϑ_i and ϑ_j:

$$B_{i,j} \triangleq \frac{P(\vartheta_i|X_1, \ldots, X_n)}{P(\vartheta_j|X_1, \ldots, X_n)} = \frac{P(X_1, \ldots, X_n|\vartheta_i)\, P(\vartheta_i)}{P(X_1, \ldots, X_n|\vartheta_j)\, P(\vartheta_j)}$$

This value describes the likelihood of ϑ_i over ϑ_j. Different scales exist in order to determine the likelihood: generally speaking, if $B_{i,j} > 3$ the likelihood can be considered substantial.

This approach can be also used to compare the likelihood of different models or different operating modes of the same model.

Approximate Bayesian Computation ABC consists in a class of computational methods that can be used to approximate the likelihood function through a large number of simulations and estimate the posterior distributions of model parameters. The most basic ABC method is the *rejection algorithm*, which consists in the following steps:

1. sample a set of parameter values $\vartheta_1, \ldots, \vartheta_N$ from the prior distribution;
2. use the model to generate a set of data $\hat{d}_j$ for each sampled value ϑ_j;
3. compare each generated set of data $\hat{d}_j$ – or an appropriate statistical descriptor $\Phi[\hat{d}_j]$ – with the observed set of data d – or $\Phi[d_j]$) – using an appropriate distance metric $\rho\Big(\Phi[\hat{d}_j], \Phi[d_j]\Big)$;
4. define for the distance metric a threshold ε above which the generated set of data, and the corresponding parameter value, is rejected for the observed set of data.

Chapter 7

Feedback control systems

Feedback control systems are commonly modeled through transfer functions. This chapter shows the effects of linearization, parametric errors, parametric faults, and disturbances.

7.1 Feedback control scheme

7.1.1 Block diagrams

Block diagram of the physical system The block diagram shown in fig. 7.1 describes a system with feedback control, where:

- $P(s)$ is the transfer function of the process;
- $y(t)$ is the physical variable of the process which has to be controlled, and the signal $y_s(t)$ is its measure;
- $w(t)$ is the desired value for $y(t)$, and $w_s(t)$ is the desired value (reference signal) for its measure $y_s(t)$;
- the error $e_s(t)$ is the difference between the reference signal $w_s(t)$ and the measurement signal $y_s(t)$ of the output;
- $T(s)$ is the stable transfer function of the transducer (sensor) and $RSG(s)$ is the reference signal generator (ideally, $RSG(s) = T(s)$);
- $A(s)$ is the transfer function of the actuator, and $m(t)$ is the physical variable which the actuator is acting on;
- $R(s)$ is the transfer function of the controller, and its output $\tilde{u}(t)$ is the control variable;
- $d_A(s)$, $d_P(s)$ and $d_T(s)$ are exogenous variables, which represent noises and disturbances acting on the actuator, the process and the transducer, respectively, through the corresponding transfer function $H_*(s)$.

Simplified block diagram for control design purposes The block diagram in fig. 7.1 can be elaborated as shown in fig. 7.2, where the block

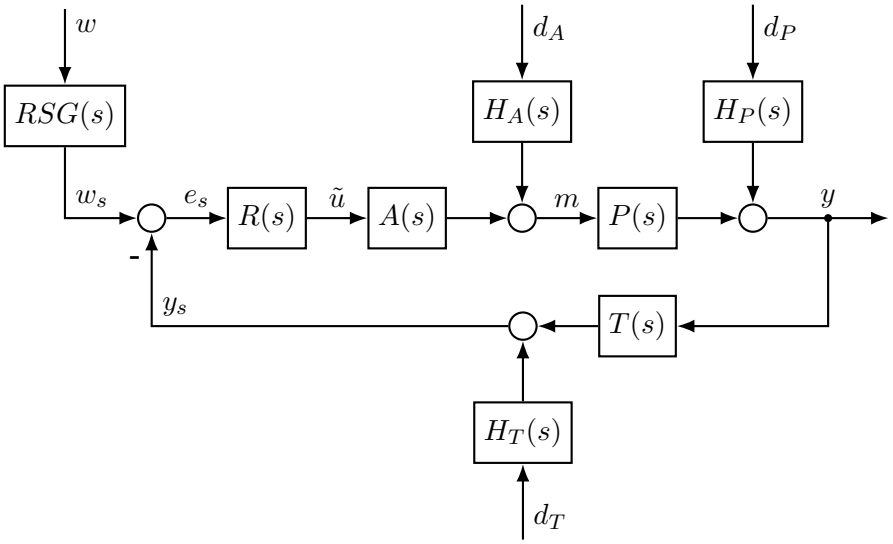


FIGURE 7.1: Block diagram of a feedback control system

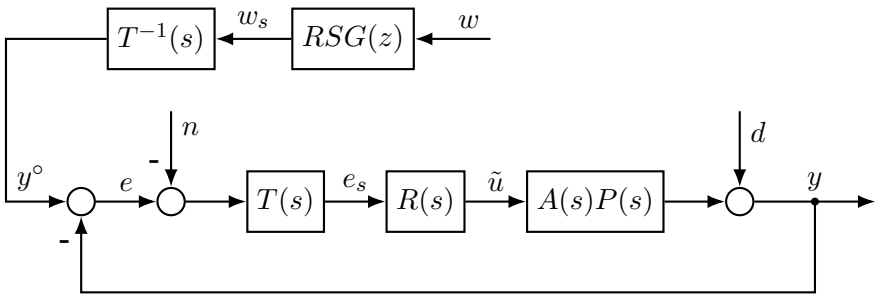


FIGURE 7.2: Re-elaborated block diagram of a feedback control system

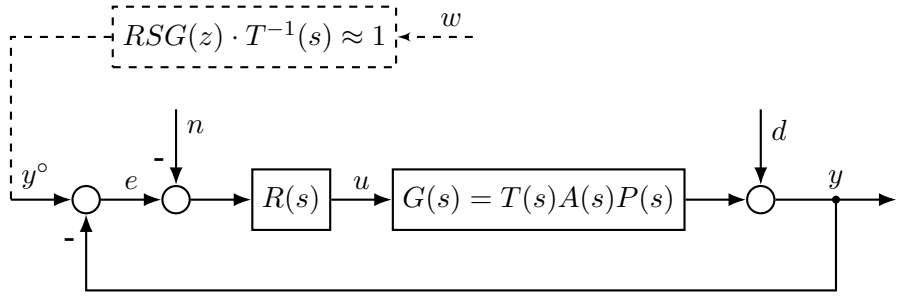


FIGURE 7.3: Feedback control scheme (full)

$T(s)$ has been moved at the beginning of the forward line, according with block diagram elaboration rules. In this diagram:

- the control system error $e(t)$ is the difference between the actual output variable $y(t)$ and its filtered desired value $y^\circ(t)$, where $y^\circ(t) \approx w(t)$ if $RSG(s) \approx T(s)$;
- $n(t) = T^{-1}(s)H_T(s)d_T(t)$ represents the noises and disturbances on the transducer;
- $d(t) = H_P(s)d_P(t) + P^{-1}(s)H_a(s)d_A(t)$ represents the noises and disturbances on the plant and the actuator.

Finally, by replacing the block $R(s)$ with $T^{-1}(s)R(s)T(s)$ we obtain the simplified feedback control scheme shown in fig. 7.3, where:

- the block $T(s)$ has been moved at the beginning of the forward line, according to block diagram elaboration rules;
- the control system error $e(t)$ is the difference between the actual output variable $y(t)$ and its filtered desired value $y^\circ(t)$, where $y^\circ(t) \approx w(t)$ if $RSG(s) \approx T(s)$;
- the (strictly proper) transfer function $G(s)$ represents the whole process, which is made of the actuator, the transducers and the actual plant.

It should be noted that the block $R(s)$ shown in fig. 7.4 apparently has a different input and output with respect to the block shown in fig. 7.1. However, recalling that the blocks $R(s)$ and $G(s)$ are actually a simplification of $T(T^{-1}RT)AP$, this control scheme is equivalent to the physical control system, although the variables $e_s(t)$ and $\tilde{u}(t)$ have been hidden by the simplifications. The simplified control scheme allows to design the controller considering the actual physical variable instead of its measurement, independently of the dynamics of the transducer (which are part of the process' transfer function $G(S)$).

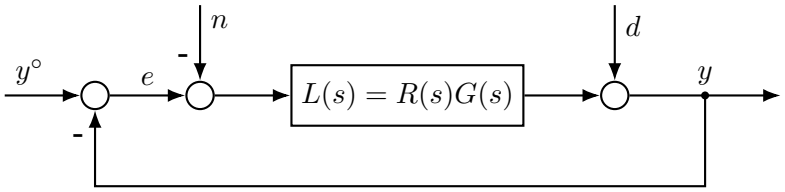


FIGURE 7.4: Feedback control scheme (simplified)

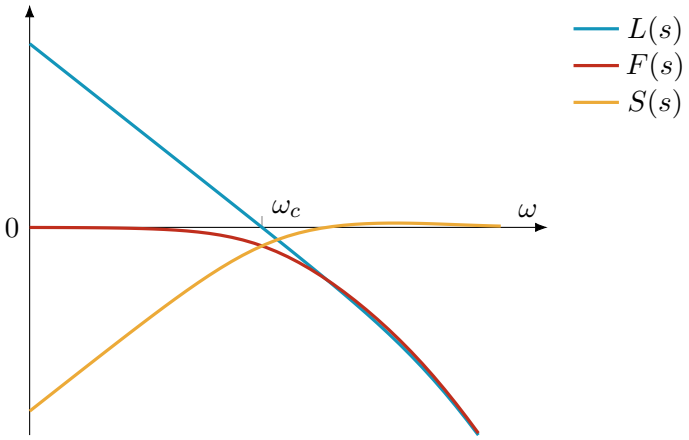


FIGURE 7.5: Sensitivity Functions

7.1.2 Open-loop transfer function and sensitivity functions

We define the *open-loop transfer function* $L(s) \triangleq R(s)G(s)$ as the forward path of the loop. Starting from $L(s)$, we define the *sensitivity* $S(s)$, the *complementary sensitivity* $F(s)$ and the control sensitivity $Q(s)$ as:

$$S(s) \triangleq \frac{1}{1+L(s)}\,, \quad F(s) \triangleq \frac{L(S)}{1+L(s)}\,, \quad Q(s) \triangleq \frac{R(S)}{1+L(s)}$$

Sensitivity functions are the transfer functions between the input and output signals of the feedback control scheme in fig. 7.3:

out \ in	$W(s)$	$D(s)$	$N(s)$
$Y(s)$	$F(s)$	$S(s)$	$-F(s)$
$U(s)$	$Q(s)$	$-Q(s)$	$-Q(s)$
$E(s)$	$S(s)$	$-S(s)$	$F(s)$

If Bode hypotheses hold (see appendix B.2), it can be shown that:

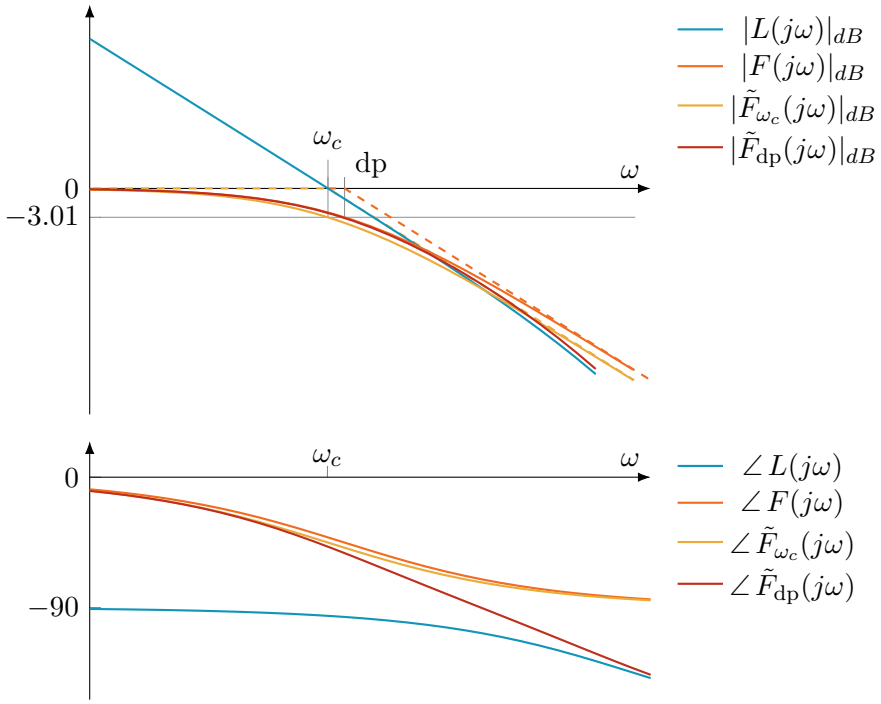


FIGURE 7.6: Approximation of the complementary sensitivity functions

$$|F(j\omega)| \approx \begin{cases} \mu_F\,, & ,\ \omega \leq \omega_c \\ |L(j\omega)| & ,\ \omega > \omega_c \end{cases} \quad , \quad \mu_F = \begin{cases} 1 & ,\ g_L > 0 \\ \mu_L/(1+\mu_L) & ,\ g_L = 0 \end{cases}$$

$$|S(j\omega)| \approx \begin{cases} \dfrac{1}{|L(j\omega)|} & ,\ \omega \leq \omega_c \\ \mu_S & ,\ \omega > \omega_c \end{cases} \quad , \quad \mu_S = \begin{cases} 1/(1+\mu_L) & ,\ g_L > 0 \\ 1 & ,\ g_L = 0 \end{cases}$$

$$|Q(j\omega)| \approx \begin{cases} \dfrac{1}{|G(j\omega)|} & ,\ \omega \leq \omega_c \\ |R(j\omega)| & ,\ \omega > \omega_c \end{cases}$$

Approximation of the complementary sensitivity function

Dominant poles of $F(s)$ are close to the gain crossover frequency ω_c (see definition B.2.1), therefore $F(s)$ can be approximated with a low-pass filter with gain μ_F and cut-off frequency ω_c. Figure 7.6 shows the Bode plots of $F(s)$ together with its dominant pole approximation $\tilde{F}_{\mathrm{dp}}(s)$ and its approximation $\tilde{F}_{\omega_c}(s)$ to a low-pass filter with cut-off frequency ω_c

Given the phase margin φ_m of $L(s)$ (see definition B.2.1), the following empirical rule holds:

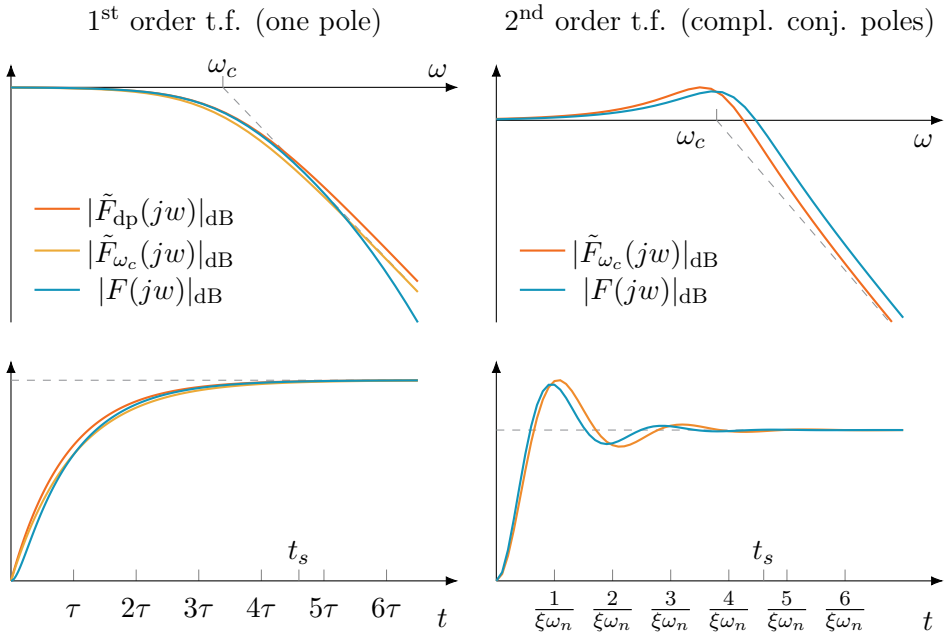


FIGURE 7.7: Magnitude plot and step response $F(s)$

- if $\varphi_m \geq 75°$ then $F(s)$ is likely to have a real dominant pole, whose time constant is $\tau \simeq 1/\omega_c$, and the settling time of the step response is $t_s \simeq 5\tau$;
- if $\varphi_m \leq 75°$ then $F(s)$ is likely to have two complex conjugated dominant poles, with $\omega_n \simeq \omega_c$ and $\xi \simeq \sin(\varphi_m/2) \simeq \varphi_m/100°$, and the settling time is $t_s \simeq 5/\xi\omega_c$.

The approximated transfer function is:

$$F(s) \approx \begin{cases} F_{\text{real}}(s) = \dfrac{\mu_F}{1+s\tau} & , \quad \varphi_m \geq 75° \\ F_{\text{c.c.}}(s) = \dfrac{\mu_F}{1+2\,\xi\, s/\omega_n + s^2/\omega_n^2} & , \quad \varphi_m < 75° \end{cases}$$

The approximation is usually good for $\omega \leq \omega_c$; since high frequencies poles are neglected, the real transfer function usually have a higher slope and a different phase for $\omega \gg \omega_c$.

Example (7.1.1)
We consider a process described by the transfer function $G(s)$, and two different closed-loop regulators $R_I(s)$ and $R_{II}(s)$:

$$G(s) = \frac{0.05}{(1+2s)(1+100s)}\,, \quad R_I(s) = \frac{1+100s}{s}\,, \quad R_{II}(s) = \frac{20\,(1+100s)}{s}$$

The open-loop transfer functions are:

$$L_I(s) = \frac{0.05}{s\,(1+2s)}\,, \quad L_{II}(s) = \frac{1}{s\,(1+2s)}$$

In the first case $\varphi_m = 84.3°$ *and* $\omega_c = 0.0497\,\mathrm{rad/sec}$*; the dominant pole of* $F(s)$ *is real, as expected, at* $-0.0564\,\mathrm{rad/sec}$*. In the second case* $\varphi_m = 38.7°$ *and* $\omega_c = 0.6248\,\mathrm{rad/sec}$*; the dominant poles of* $F(s)$ *are complex conjugate, as expected, with* $\omega_n = 0.7071$ *and* $\xi = 0.3536$*; an approximation of these values can be obtained as* $\omega_c \simeq \omega_n$ *and* $\xi \simeq \varphi_m/100° = 0.387\,\mathrm{rad}$*.*

Figure 7.7 shows the approximations of $F(s)$ *and their step response.*

7.2 Transfer functions and parametric faults

The parameters of a transfer function, such as the gain μ and the time constants τ_i, depend on the physical quantities characterizing the system. In case of parametric faults, that is, undesired variations of the physical quantities of the system, the transfer function of the faulty system will have different parameters, but its expression (i.e. the type g and the number of singularities) won't change.

For example, the transfer function of a low-pass RC filter is $T(s;\tau) = 1/(1+s\tau)$, where the time constant is $\tau = R \cdot C$. If the resistor breaks down, its resistance R becomes very small, and the time constant of the filter changes accordingly: the new transfer function will be $T(s;\tilde{\tau}) = 1/(1+s\tilde{\tau})$, with $\tilde{\tau} \ll \tau$.

Any variation of the parameters causes a variation of the (step or frequency) response of the system. A parametric fault causes the system to behave differently from its (healthy) model, and unavoidable uncertainties and approximations of the parameters *always* cause the model to behave differently from the system. With the only knowledge of the responses of the system and the transfer function there is no way to attribute the cause of their difference to a fault rather than to an approximation. However, the variation of a parameter due to a fault usually has a higher magnitude than any uncertainty or approximation. Besides that, the transfer function is often a further simplification of the model, as it could be a dominant pole approximation or a linearization, and this also causes its response to be different.

In summary, the differences between the transfer function model and the real system can be attributed to the following reasons:

- model parameters are an approximation of the real parameters
- the transfer function is a dominant pole approximation
- the real system is not LTI (parameters depend on time and linearization point)
- the transfer function doesn't include delays

7.2.1 Linearization effects

In order to describe a non-linear system through a transfer function it is necessary to linearize the system around a given operating or equilibrium point. If the dynamic equations of the system are regular enough, the transfer function has the operating point as parameter. If we change the operating point without updating the transfer function we are introducing a parametric error, which may become bigger (diventa più grande più ci si allontana dal punto di lavoro). Intuitively, trespassing a singular point may introduce unpredictable effects on the step response, which may be difficult to quantify and may be confused with a fault.

Example: Pendulum position control (7.2.1)
The following equation describes the dynamic of a pendulum consisting of a mass M which is linked to a rigid weightless bar of length l, to which are applied a torque $u(t)$ and a friction torque which is proportional to the angular speed (with a coefficient $k > 0$):

$$\begin{aligned} Ml\,\ddot{\vartheta}(t) &= -Mg\,\sin(\vartheta(t)) - \frac{k\,\vartheta(t)}{l} + \frac{u(t)}{l} \quad \Longrightarrow \\ \ddot{\vartheta}(t) &= -\frac{g}{l}\,\sin(\vartheta(t)) - \frac{k}{Ml^2}\,\dot{\vartheta}(t) + \frac{1}{Ml^2}\,u(t) = f(\vartheta, u) \end{aligned} \tag{7.1}$$

The state-space representation of the system is:

$$\begin{cases} \dot{x}_1(t) = x_2(t) & = f_1(x_1, x_2, u) \\ \dot{x}_2(t) = -\dfrac{g}{l}\,\sin(x_1(t)) - \dfrac{k}{Ml^2}\,x_2(t) + \dfrac{1}{Ml^2}\,u(t) & = f_2(x_1, x_2, u) \\ y(t) = x_1(t) & = g(x_1, x_2, u) \end{cases}$$

$$x(t) = \begin{bmatrix} x_1(t) & x_2(t) \end{bmatrix}^{\mathsf{T}} = \begin{bmatrix} \vartheta(t) & \dot{\vartheta}(t) \end{bmatrix}^{\mathsf{T}}, \quad f(x,u) = \begin{bmatrix} f_1(x,u) & f_2(x,u) \end{bmatrix}^{\mathsf{T}}$$

Given a constant input $u(t) = \bar{u}$, an equilibrium state for the pendulum is a constant solution $\bar{x} = \begin{bmatrix} \bar{x}_1 & \bar{x}_2 \end{bmatrix}^{\mathsf{T}} = \begin{bmatrix} \bar{\vartheta} & 0 \end{bmatrix}^{\mathsf{T}}$ such that $f_2(\bar{\vartheta}, 0, \bar{u}) = 0$, and the variables can be written as:

$$x_1(t) = \bar{x}_1 + \delta x_1(t)\,, \quad x_2(t) = \delta x_2(t)\,, \quad u(t) = \bar{u} + u\vartheta(t)$$

The linearized equations are:

$$\begin{cases} \delta\dot{x}_1(t) = \delta x_2(t) \\ \delta\dot{x}_2(t) = -\dfrac{g}{l}\,\cos(\bar{\vartheta}) \cdot \delta x_1(t) - \dfrac{k}{Ml^2}\,\delta x_2(t) + \dfrac{1}{Ml^2}\,\delta u(t) \\ \bar{y} + \delta y(t) = \bar{x}_1(t) + \delta x_1(t) \end{cases}$$

In matrix form:

$$\begin{cases} \delta\dot{x}(t) = A_P\,\delta x(t) + B_P\,u(t) \\ \delta y(t) = C_P\,\delta x(t) \end{cases}$$

$$A_P(\bar{\vartheta}) = \begin{bmatrix} 1 & 0 \\ -\dfrac{g}{l}\cos(\bar{\vartheta}) & -\dfrac{k}{Ml^2} \end{bmatrix}, \quad B_P = \begin{bmatrix} 0 \\ \dfrac{1}{Ml^2} \end{bmatrix}, \quad C_P = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

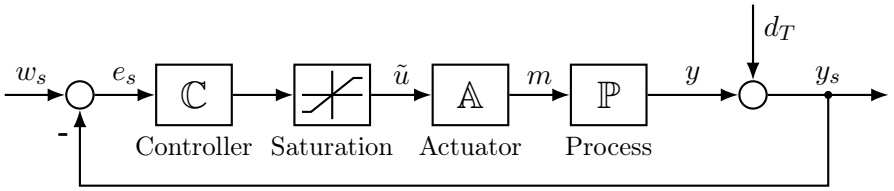
Assuming $l = 1$, $M = 1$, $k = 1$, the transfer function of the linearized system is:

$$P(s;\bar{\vartheta}) = C_P \left[sI - A_P(\bar{\vartheta})\right]^{-1} B_P = \frac{1}{s^2 - s + g\,\cos\bar{\vartheta}}$$

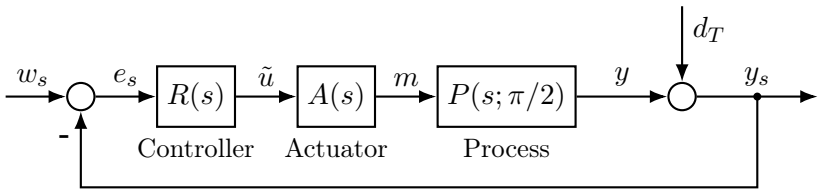
The actuator's dynamic is described by the transfer function:

$$A(s) = \frac{1}{1 + 0.5\,s}$$

We assume that the transfer function of the transducer is $T(s) = 1$, so that $G(s) = A(s)\,P(s)$. Figure 7.8 shows the block diagram of the non-linear (a) and of the linearized (b) system.



(a) Non-linear system



(b) linearized system $(\bar{\vartheta} = \pi/2)$

FIGURE 7.8: Bock diagram of the pendulum position controller

We consider two regulators, with (R_1) and without (R_2) integral action:

$$R_1(s) = \frac{20\,(1 + 0.3\,s)\,(1 + 3\,s)}{s\,(1 + 0.003\,s)}, \quad R_2(s) = \frac{100\,(1 + 0.1\,s)}{1 + 0.01\,s}$$

The output of the regulators has been limited to $[-50, +50]$ in the non-linear model.

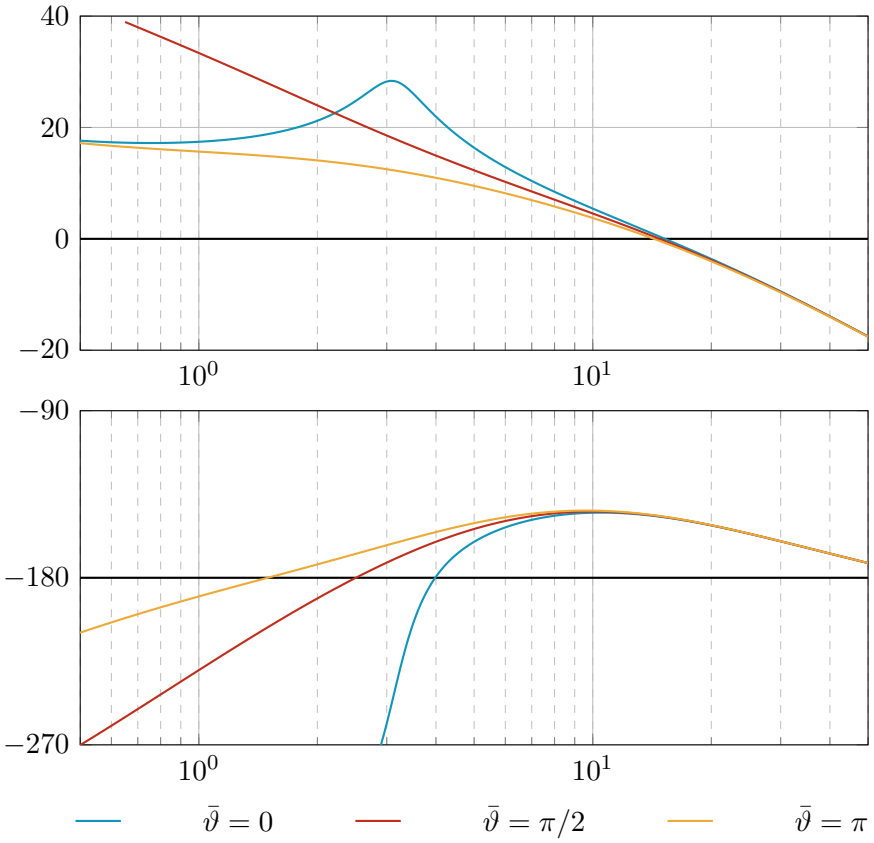


FIGURE 7.9: Bode Diagram of $L(s)$ with $R_1(s)$

The loop transfer functions are given by $L_i(s;\bar{\vartheta}) = R_i(s)\,A(s)\,P(s;\bar{\vartheta})$*; the Bode plots are shown in figs. 7.9 and 7.10, and gain cross-over frequency and margin are:*

	$L_1(s)$			$L_2(s)$		
$\bar{\vartheta}$	0	$\pi/2$	π	0	$\pi/2$	π
ω_c	15.25	14.79	14.30	11.91	11.42	10.90
φ_m	32.56	33.12	33.70	7.26	7.55	7.81

The response $\bar{y} + \delta y(t)$ *of the linearized system introduces an error* $\varepsilon_y(t)$ *with respect to the response* $y(t)$ *of the non-linear system. Figures 7.11 to 7.14 show the response of the non-linear and the linearized system to a step signal and a sine (on the left side), and the error introduced by the linearization* $\varepsilon_y(t) = ((\bar{y} + \delta y(t)) - y(t))$ *(on the right side).*

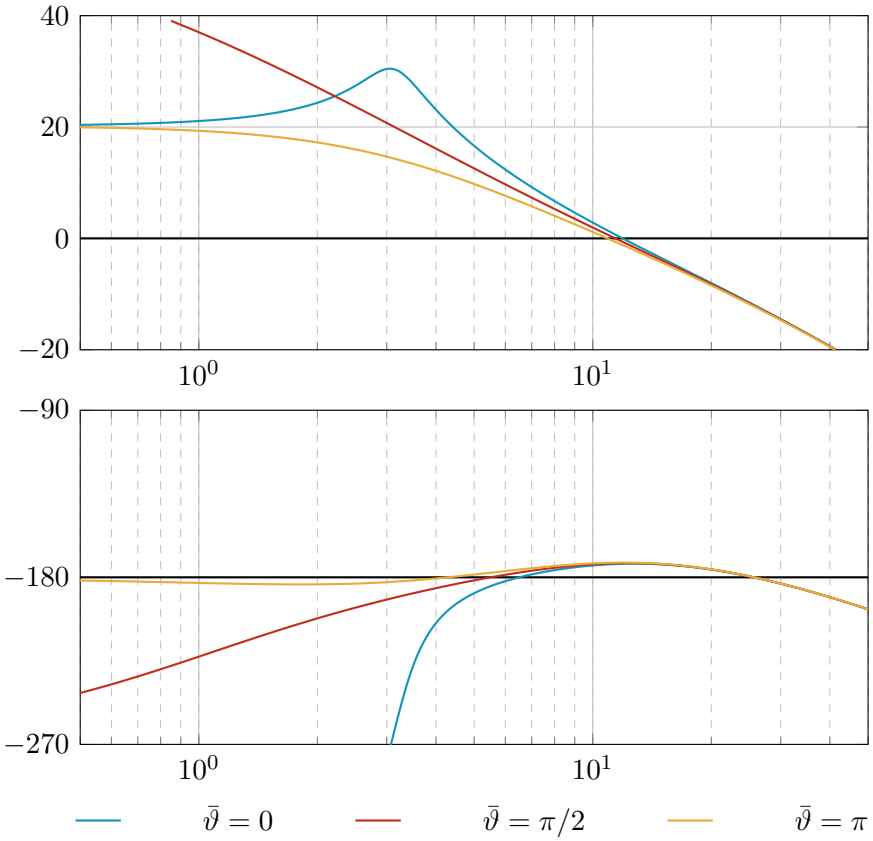


FIGURE 7.10: Bode Diagram of $L(s)$ with $R_2(s)$

Setpoint $\vartheta^\circ(t)$	ϑ_0	Lin. points $\bar{\vartheta}$	Figure
$\pi/2 \cdot \text{step}(t)$	0	$0, \pm\pi/2, \pi/4, \pi$	7.11 (R_1 and R_2)
$\pi/2 + \pi/4 \cdot \sin(\pi\, t)$	0	$0, \pi/2$	7.12 (R_1 and R_2)
$\pi/2 + \pi/4 \cdot \sin(10\,\pi\, t)$	$\pi/2$	$0, \pi/2$	7.13 (R_1) – 7.14 (R_2)

Step response has the following characteristics:

- *with integral action* (R_1)*:*
  - *the response of the non-linear system doesn't have oscillations;*
  - *the response of the linearized system has oscillations;*
  - *both responses tend to zero*
  - *the response of the linearized systems with* $\bar{\vartheta} = \pm\pi/2$ *are closer to the response of the non-linear system;*
- *without integral action* (R_2)*:*
  - *both the responses of the non-linear and of the linearized systems have oscillations;*
  - *the response of the non-linear system is faster, and its oscillations are smaller and slower;*
  - *both responses don't generally tend to zero due to the lack of integral action (with the only exception of the responses of the linearized system with* $\bar{\vartheta} = \pm\pi/2$*, as* $P(s; \pm\pi/2)$ *has a pole in the origin);*

Sine response has the following characteristics:

- *at a low sine frequency* ($\pi\,\text{rad/s} < \omega_c$) *the difference* $\varepsilon_y(t)$ *is similar to the step response case, with both regulators;*
- *at a high sine frequency* ($10\pi\,\text{rad/s} > \omega_c$) *the system is not able to track the reference:*
  - *the period of the response signal is the same as that of the reference signal, but out of phase*
  - *the amplitude of the response of the linearized system with* $\bar{\vartheta} = \pi/2$ *is reduced to 43.1%* ($|F(j10\pi; \pi/2)| = 0.431$)
  - *the amplitude of the response of the non-linear system is reduced to 4.28%*
  - *the linearized system have better performances than the non-linear system, as the regulators have been tuned on the former*
- *with integral action* (R_1) *the response is centered on* $\pi/2$
- *without integral action* (R_2) *the response is centered below* $\pi/2$ *(except for the case with* $\bar{\vartheta} = \pm\pi/2$*)*

Disturbances. *We introduce a disturbance* $d_T(t) = 0.05 \sin(\omega t)$ *(with a* peak-to-peak *amplitude of 0.1) on the measurement of* $y = \vartheta$ *and we measure its effect on the system.*

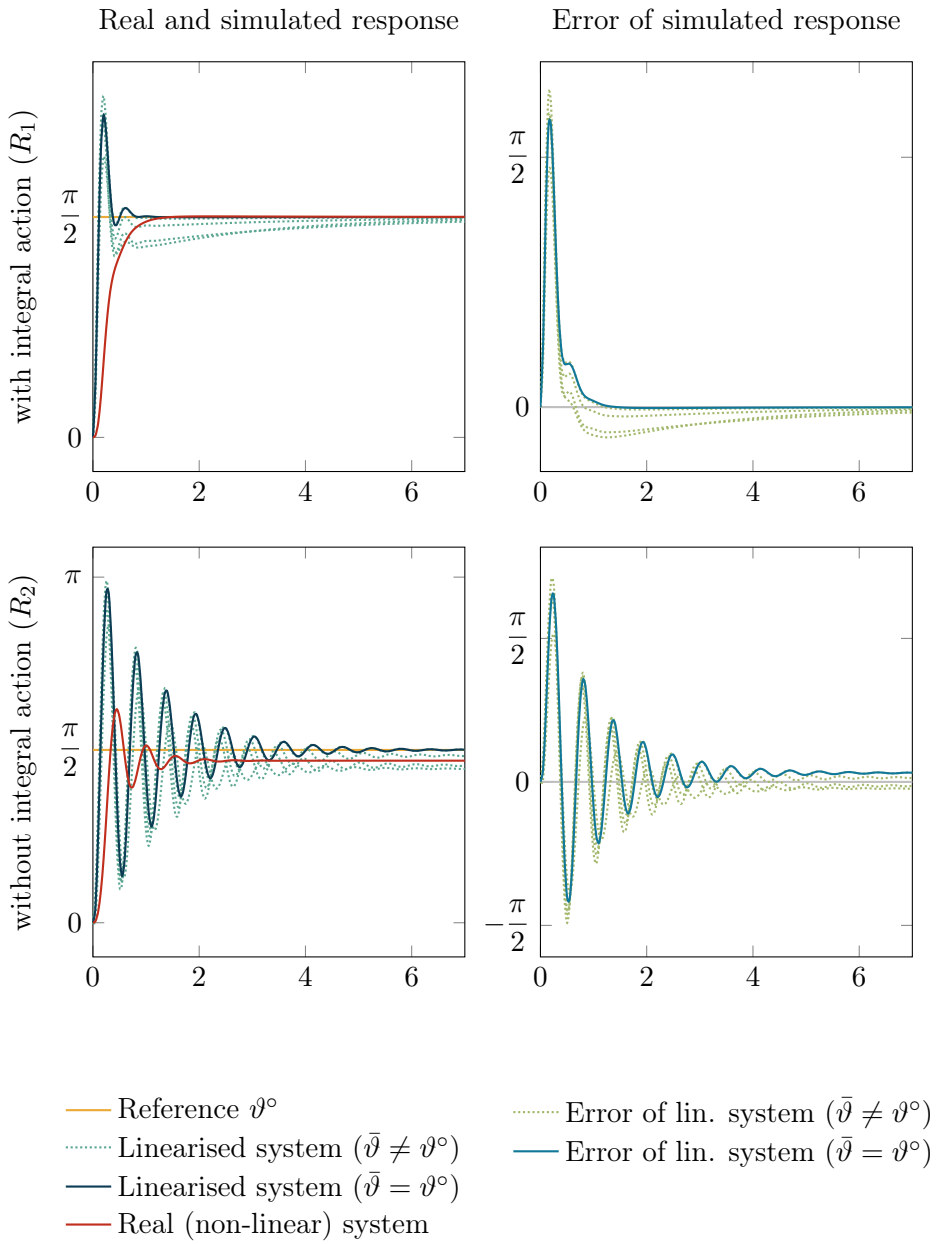


FIGURE 7.11: Step response

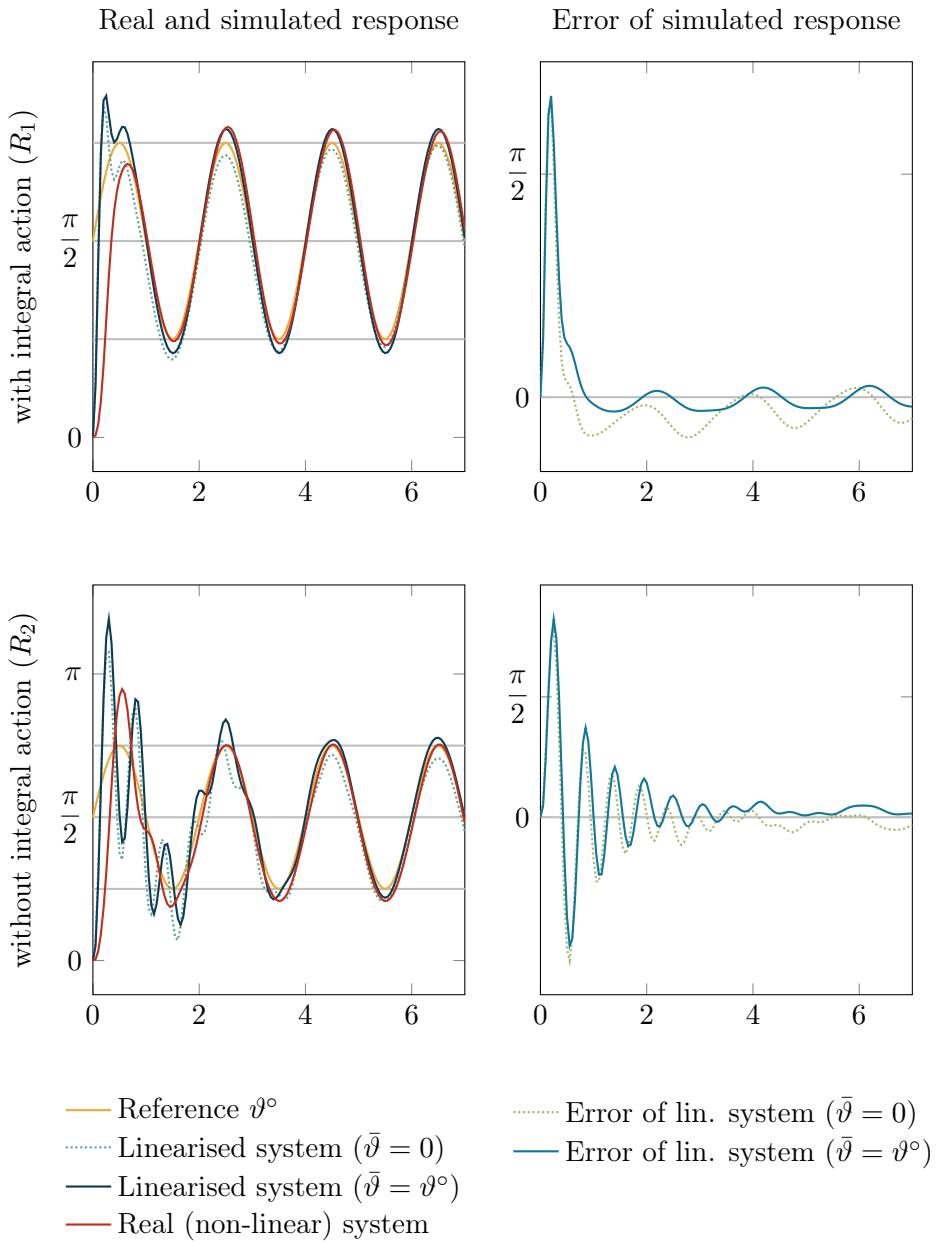


FIGURE 7.12: Sine response – $\vartheta^\circ(t) = \pi 2 + \pi/4 \cdot \sin(\pi\, t)$

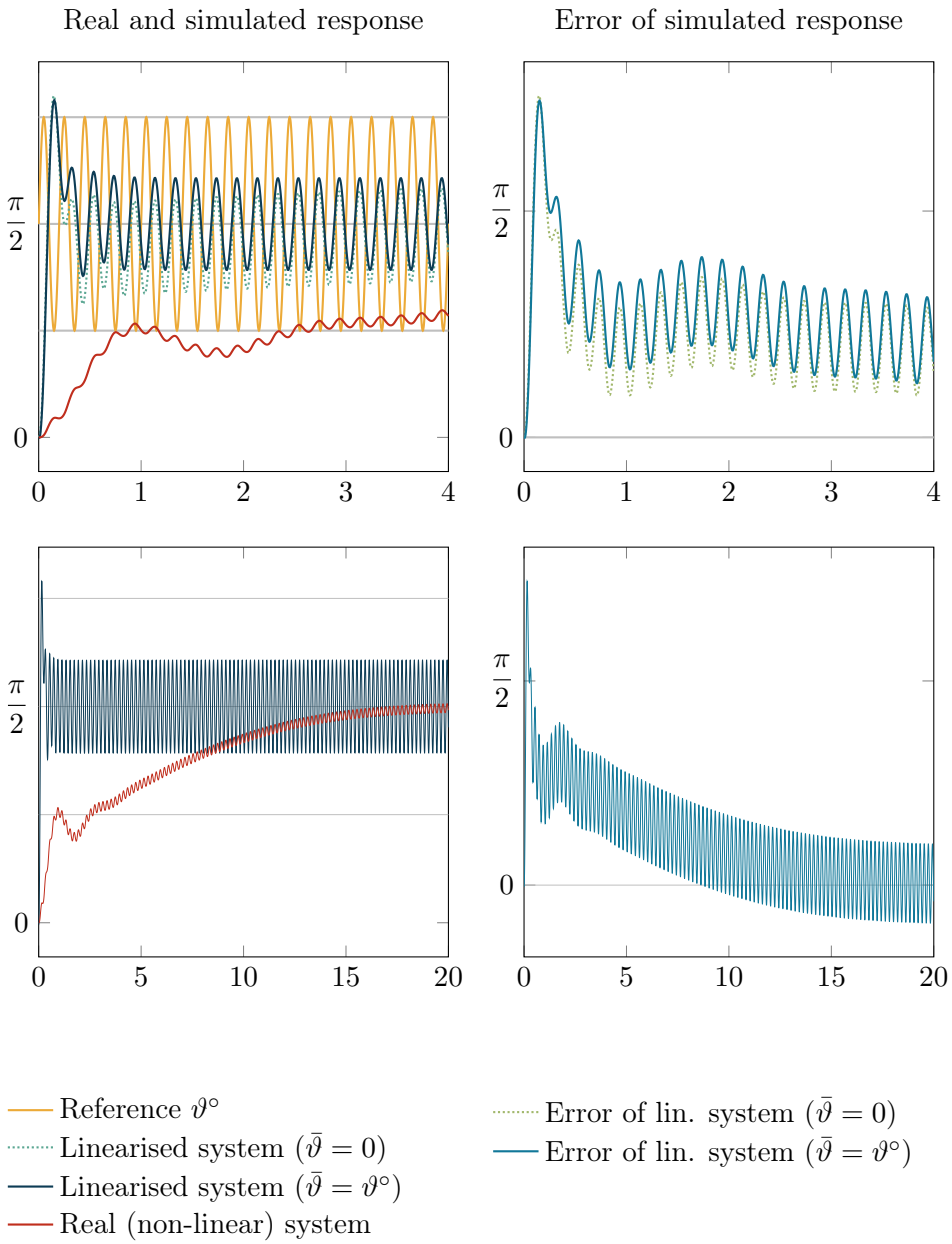


FIGURE 7.13: Sine response – $\vartheta^\circ(t) = \pi 2 + \pi/4 \cdot \sin(10\,\pi\,t)$ (with R_1)

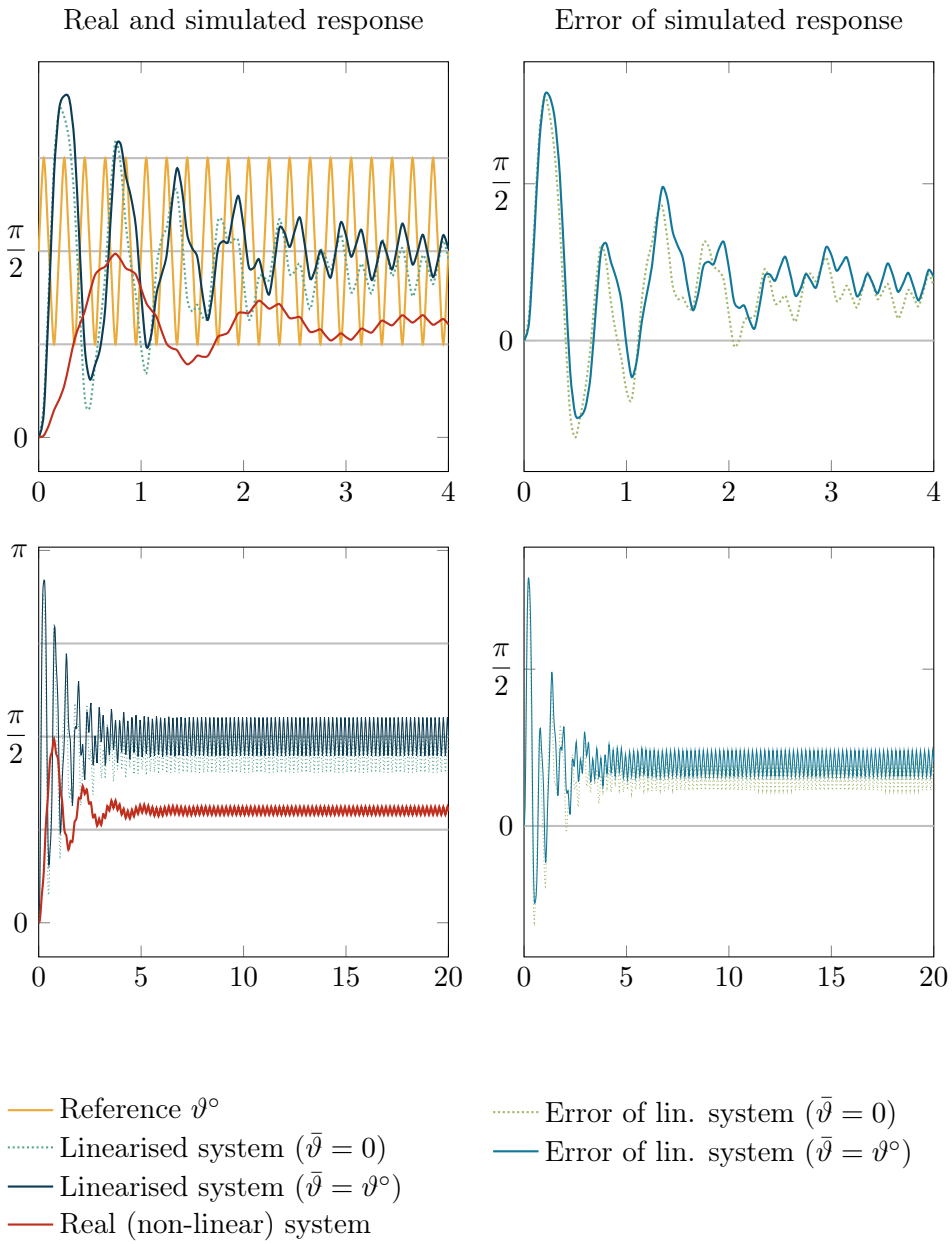


FIGURE 7.14: Sine response - $\vartheta^\circ(t) = \pi 2 + \pi/4 \cdot \sin(10\,t)$ (with R_2)

		Freq. response		linearized ($\pi/2$)		Non-linear	
	ω	$\lvert F(j\omega)\rvert$	$\lvert -S(j\omega)\rvert$	y_{PP}	$y_{s\,PP}$	y_{PP}	$y_{s\,PP}$
$R_1(s)$	1	1.0139	0.0217	0.1014	0.0022	0.1018	0.0023
	10	1.6045	0.9501	0.1604	0.0950	0.1355	0.0803
	50	0.1525	1.1508	0.0152	0.1151	0.0152	0.1149
	1000	0.0001	1.0000	10^{-5}	0.1000	$\approx 10^{-5}$	0.0999
$R_2(s)$	1	1.0104	0.0142	0.1010	0.0014	0.1000	0.0010
	10	4.3618	3.4828	0.4361	0.3483	0.2824	0.2234
	50	0.0724	1.0689	0.0072	0.1069	0.0072	0.1070
	1000	$\approx 10^{-5}$	1.0000	$\approx 10^{-6}$	0.1000	$\approx 10^{-6}$	0.0999

TABLE 7.1: Effect of disturbances on the output variable

	$R_1(s)$		$R_2(s)$	
ω	Δy_{PP}	$\Delta y_{s\,PP}$	Δy_{PP}	$\Delta y_{s\,PP}$
1	$\approx 0\%$	4.5%	1.0%	28.6%
10	7.4%	15.5%	35.2%	35.6%
50	$\approx 0\%$	$\approx 0\%$	$\approx 0\%$	$\approx 0\%$
1000	n.d.	$\approx 0\%$	n.d.	$\approx 0\%$

TABLE 7.2: Effect of disturbances on the output variable

Table 7.1 show the effect of the disturbance on both the position y and its measurement y_s, as its peak-to-peak amplitude. Table 7.2 compares the effect on the linearized system with the effect on the non linear system. At low frequencies, the effect is higher on the linearized system.

Chapter 8

Conclusions

The possible role of the models has been discussed in chapters 2 and 3, together with related issues.

Chapters 4 and 5 shows different techniques to detect abnormal behaviors in the system. Data-driven methods require to know the normal behavior of the plant, that is, the expected value of some monitored variables, and models can be useful to generate data when it is not possible to obtain them from the real system.

Chapter 6 shows how faults can be implemented and simulated in Modelica models. There are many ways to use data generated by simulations, as discussed in chapters 2 and 3. A possible use is Bayesian inference, that allow to identify the configuration of the system that led to its observed behavior.

The first part of Chapter 7 describes the structure of a feedback control system and the effect of additive disturbances and parametric faults. Block diagram elaboration rules allow to easily obtain the transfer function from a disturbance to any monitored variable. Many systems can be approximated to a first or second order transfer function, and this allow to understand the magnitude and the speed of the effect of a parametric fault. However, as discussed in the second part of the chapter, the effect of approximations introduced by transfer function models may be non negligible, and the behavior of a linearized model can be very different from the behavior of the real system, therefore any conclusion must be treated with great caution. For these reasons, transfer functions and block diagram may be very useful in order to understand how disturbances and parametric faults affect the system, but they not be used as digital twins as they are not reliable.

Other techniques that are based on transfer functions, such as residual generators, are very useful for fault isolation, but great care must be taken and the transfer function must be a good approximation of the model.

Appendices

Appendix A

Statistics

A.1 Random variables

Definition: probability (A.1.1). Probability is the measure $P(E) \in [0, 1]$ of the likelihood that an event E will occur. An event that never occurs has probability $P = 0$, while an event that certainly occurs has probability $P = 1$.

Definition: random variable (A.1.2). A random variable X (also called *aleatory variable* or *stochastic variable*) is a variable whose possible values are outcomes of a random phenomenon.

A random variable whose set of possible values can be written either as a finite sequence $x_1, x_2, \ldots, x_n$ (*e.g.* the result of a die roll) or as a countable infinite sequence $x_1, x_2, \ldots$ (*e.g.* $\mathbb{N}$ or the positive multiples of 3) is said to be *discrete*. A random variable that take values on a continuum of possible values (*e.g.* $\mathbb{R}$ or any subset $\mathcal{B} \subseteq \mathbb{R}$) is said to be *continuous*.

Definition: probability mass function (A.1.3). The probability mass function $p_X(x)$ of a discrete random variable X assuming values $x_1, x_2, \ldots$ is defined as the probability of X being equal to a:

$$p_X(x) = P\{X = x\} \quad \begin{cases} > 0\,, & x \in \{x_1, x_2, \ldots\} \\ = 0\,, & x \notin \{x_1, x_2, \ldots\} \end{cases} \tag{A.1}$$

The probability mass function has the following property:

$$\sum_{i=1,\,2\ldots} p_X(x_i) = 1 \tag{A.2}$$

Definition: probability density function (A.1.4). The probability density function (PDF) of a continuous random variable X is the function $f_X(x)$, defined for $x \in \mathbb{R}$, such that:

$$P\{X \in \mathcal{B}\} = \int_{\mathcal{B}} f_X(x)\, dx$$

It follows that:

$$P\{X = a\} = \int_a^a f_X(x)\,dx = 0\,, \qquad P\{X \in \mathbb{R}\} = \int_{-\infty}^{\infty} f_X(x)\,dx = 1$$

$$P\{a < X < b\} = P\{a \leq X \leq b\} = \int_a^b f_X(x)\,dx$$

Definition: distribution function (A.1.5). The (cumulative) distribution function F of the random variable X is defined for any $x \in \mathbb{R}$ as the probability that the value of a random variable X is less than or equal to x:

$$F_X(x) = P\{X \leq x\} \Longleftrightarrow X \sim F_X \tag{A.3}$$

We write $X \sim F$ to indicate that F is the distribution function of X.
For continuous random variables:

$$F_X(a) = P\{X \in (-\infty, a)\} = \int_{-\infty}^{a} f_X(x)\,dx \tag{A.4}$$

For discrete random variables:

$$F_X(a) = \sum_{x \leq a} p_X(x)$$

In both cases, $F_X(x)$ is monotonically non-decreasing, and:

$$F_X(x) \to 0 \text{ for } x \to -\infty\,, \quad F_X(x) \to 1 \text{ for } x \to \infty$$

Definition: expected value or mean (A.1.6). The expected value of a random variable X, also called *mean*, is a weighted average of all the values that the variable can assume. For a discrete random variable it's defined as:

$$\mathbb{E}[X] = \sum_i x_i\, P(X = x_i) \tag{A.5a}$$

and for a continuous random variable it's defined as:

$$\mathbb{E}[X] = \int_{-\infty}^{+\infty} x\, f_X(x)\,dx \tag{A.5b}$$

Definition: variance (A.1.7). The variance of a random variable X represents its spread and it is defined as the expected value of the squared deviation from its mean μ:

$$\sigma_X^2 = \mathbb{E}[(X - \mu)] \tag{A.6}$$

Definition: standard deviation (A.1.8). The standard deviation of a random variable X is the (positive) square root σ_X of its variance σ_X^2.

A.2 Probability distributions

A.2.1 Normal distribution

Definition (A.2.1). A random variable X is said to be *normally distributed* with mean μ and variance σ^2, and we write $X \sim \mathcal{N}(\mu, \sigma^2)$, if its PDF is:

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\dfrac{(x-\mu)^2}{2\sigma^2}} \tag{A.7}$$

The *standard normal distribution* is a special case with zero mean and unit variance; a random variable with standard normal distribution is usually denoted with the letter Z, and we write $Z \sim \mathcal{N}(0, 1)$.

Proposition: Standardization (A.2.1). If $Z \sim \mathcal{N}(0, 1)$ is a standard normal variable, then $X = \sigma^2 Z + \mu \sim \mathcal{N}(\mu, \sigma^2)$. Conversely, given a normal variable $X \sim \mathcal{N}(\mu, \sigma^2)$, then $Z = (X - \mu)/\sigma^2 \sim \mathcal{N}(0, 1)$ has a standard normal distribution, and it is called the *standardized form* of X.

Normal distribution table When $z \geq 0$, the probability $P(0 < Z \leq z)$ (corresponding to the shaded area in fig. A.1) can be read from table C.2. For example, $P(0 < Z \leq 1.96)$ is:

z	...	0.05	**0.06**	0.07	...
⋮		⋮	⋮	⋮	
1.8	...	0.46784	0.46856	0.46926	...
1.9	...	0.47441	**0.47500**	0.47558	...
2.0	...	0.47982	0.48030	0.48077	...
⋮		⋮	⋮	⋮	

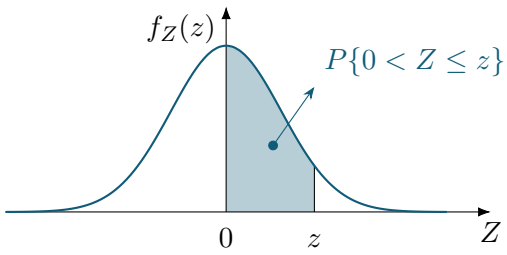


FIGURE A.1: Standard normal distribution

Since the normal distribution is symmetrical, this table can also be used to read $P(z < Z \leq 0)$ for $z \leq 0$. The following probabilities can be read from

dedicated tables, or computed from $P(0 < Z \leq z)$ as follows:

$$P\{Z > z\} = \begin{cases} 0.5 - P\{0 < Z \leq z\}, & z \geq 0 \\ 0.5 + P\{0 < Z \leq z\}, & z < 0 \end{cases}$$

$$P\{Z \leq z\} = \begin{cases} 0.5 + P\{0 < Z \leq z\}, & z \geq 0 \\ 0.5 - P\{0 < Z \leq z\}, & z < 0 \end{cases}$$

$$P\{|Z| \leq |z|\} = 2\,P\{0 < Z \leq z\}$$

$$P\{|Z| > |z|\} = 1 - 2\,P\{0 < Z \leq z\} = 2\,(0.5 - P\{0 < Z \leq z\})$$

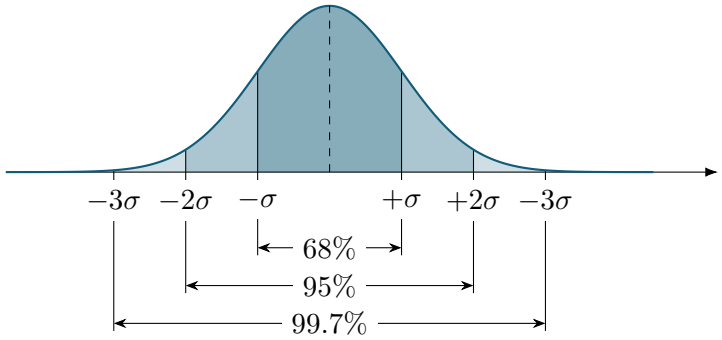


FIGURE A.2: Confidence intervals of the normal distributions

A.3 Conditional probability

Definition: conditional probability (A.3.1). The *conditional probability* of A given B, written as $P(A|B)$, is the measurement of the probability of the event A occurring given that the event F has occurred. If $P(A|B)$ then the events A and B are said to be *independent*. If $P(B) > 0$ we can write:

$$P(A|B) = \frac{P(AB)}{P(B)}$$

Theorem: Bayes' theorem (A.3.1). The conditional probability $P(A|B)$ can be written as:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

Supposing that $A_1, \ldots, A_n$ are mutually exclusive events of which one and only one must occur we can write:

$$P(A_j|B) = \frac{P(BA_j)}{P(B)} = \frac{P(B|A_j)\,P(A_j)}{\sum_{i=1}^{n} P(B|A_j)\,P(A_i)}$$

A.4 Summary Statistics

Definition: average or mean (A.4.1). The *average* of a group of data is a single numerical value that synthetically describes the whole group. The most common average is the *arithmetic mean*, or simply *mean*[1]. Other kinds of average are the *median*, the *mode* and the *geometric mean*. Given a set of n numerical values $x_1, \ldots, x_n$, the arithmetic mean is defined as $\frac{1}{n}\sum_{k=1}^{n} x_k$.

Definition: population mean and variance (A.4.2). Given a finite population of N elements, i.e. a collection of independent random variables $\{X_i\}_{i=1\ldots N}$ representing all its elements, the *population mean* μ_X is defined as its average value (i.e. the arithmetic mean), and the *population variance* σ_X^2 is defined as the average squared deviation from the mean:

$$\mu_X = \frac{1}{N}\sum_{i=1}^{N} X_i \tag{A.8}$$

$$\sigma_X^2 = \frac{1}{N}\sum_{i=1}^{N} (X_i - \mu_X)^2 \tag{A.9}$$

Definition: sample mean and variance (A.4.3). Given a population sample of size $n < N$, i.e. a subset of data $\{X_j\}_{j=1\ldots n}$, *sample mean and variance* are estimators of the population mean and variance, defined as:

$$\bar{X}_{(n)} = \frac{1}{n}\sum_{j=1}^{n} X_j \tag{A.10}$$

$$s^2 = \frac{1}{n-1}\sum_{j=1}^{n} (X_j - \bar{X}_{(n)})^2 \tag{A.11}$$

Proposition: variance of the sample mean (A.4.1). The sample mean $\bar{X}$is a random variables itself, which depend on the chosen subset of data, and have its own distribution. Its expected value $\mu_{\bar{X}}$ and variance $\sigma_{\bar{X}}^2$ are given by:

$$\bar{\bar{X}} = \mathbb{E}\big[\bar{X}_{(n)}\big] = \mathbb{E}\left[\frac{1}{n}\sum_{j=1}^{n} X_j\right] = \frac{1}{n}\sum_{j=1}^{n} \mathbb{E}[X_j] = \frac{1}{n}\sum_{j=1}^{n} \mu_X = \mu_X \tag{A.12}$$

$$\sigma_{\bar{X}}^2 = \mathrm{Var}\big[\bar{X}_{(n)}\big] = \mathrm{Var}\left[\frac{1}{n}\sum_{j=1}^{n} X_j\right] = \frac{1}{n^2}\sum_{j=1}^{n} \mathrm{Var}[X_j] = \frac{1}{n^2}\sum_{j=1}^{n} \sigma_X^2 = \frac{\sigma_X^2}{n} \tag{A.13}$$

[1] here the terms *(arithmetic) mean* and *average* are used without distinction

and the standard deviation of $\bar{X}_{(n)}$ is $\sigma_{\bar{X}} = \sigma_X/\sqrt{n}$. Therefore, the sample mean $\bar{X}_{(n)}$ is also centered on the population mean μ_X, but its spread $\sigma^2_{\bar{X}}$ decreases as the sample size n increases. If the population is normally distributed, then also the sample mean is, independently from the number of samples n.

Theorem: Central limit (A.4.2). For a large sample size (usually $n \geq 30$) the sample mean $\bar{X}_{(n)}$ is approximately normally distributed, regardless of the distribution of the population, that is, $\bar{X} \sim \mathcal{N}(\mu_X, \sigma^2_X/n)$.

A.5 Hypothesis testing

Hypothesis testing aims to make a decision about observed data, and specifically, to choose among two or more *statistical hypotheses* that usually concern the parameters of their distribution:

- the *null hypothesis* H_0, which is generally assumed to be true unless evidence indicates otherwise
- one or more *alternative hypotheses* H_i, $i = 1, 2, \ldots$ representing a situation that the test should be able to diagnose

Four possible situations may arise:

	H_0 is true		H_0 is false	
H_0 is not rejected	✓	correct decision	✗	*Type II* error (false positive)
H_0 is rejected	✗	*Type I* error (false negative)	✓	correct decision

For example, assuming to have a population sample $X_1, \ldots, X_n$ whose distribution's mean μ is unknown, we may want to test the following hypotheses on μ given the sample mean $\bar{X}$:

	Null hypothesis	Alternative hypotheses
(a)	$H_0 \colon \mu = \mu_0$	$H_1 \colon \mu \neq \mu_0$
(b)	$H_0 \colon \mu = \mu_0$	$H_1 \colon \mu > \mu_0$
(c)	$H_0 \colon \mu = \mu_0$	$H_1 \colon \mu = \mu_1$
(d)	$H_0 \colon \mu = \mu_0$	$H_1 \colon \mu = \mu_1$, $H_2 \colon \mu = \mu_2$

TABLE A.1: Examples of null and alternative hypotheses

We define the **significance level** $\alpha \in (0, 1)$ as the maximum acceptable probability of making a type I error (often expressed as percentage). It's

generally assumed $\alpha = 5\%$, unless differently specified. The set of values of X such that the null hypothesis is rejected is called **critical region**.

A test can be *two-tailed*, as in example (a), when the statistical significance is tested in two directions, or *two-tailed*, as in example (b), when it is tested in one direction. The tails correspond to the critical region: in two-tailed tests the area of each tail is $\alpha/2$, while in one-tailed tests the area of the single tail is α, as shown in fig. A.3.

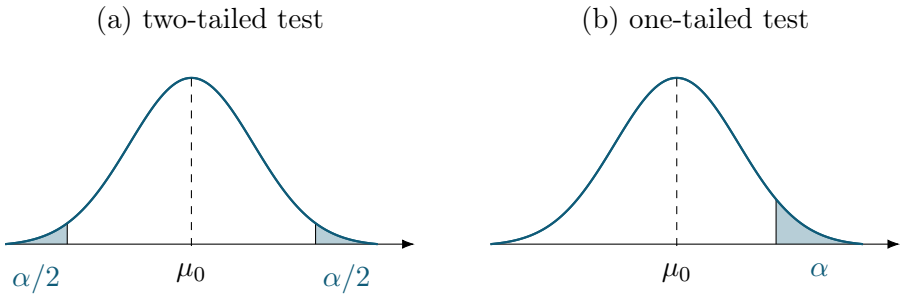


FIGURE A.3: One- and two-tailed tests

The probability of making a type II error is denoted with β, as shown in fig. A.4 for example (c). We refer to $(1 - \beta)$ as the *power* of the test.

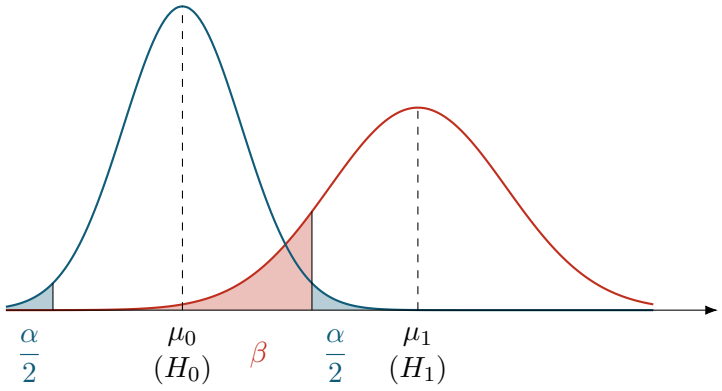
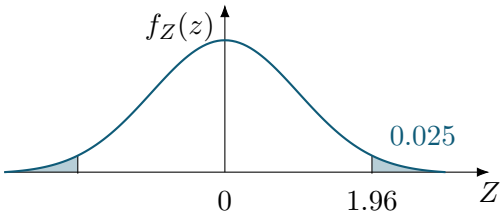


FIGURE A.4: Critical region (α) and power of the test (β)

In predictive maintenance the null hypothesis H_0 usually represents the absence of faults, and the alternative hypotheses (such as a different mean or a larger variance) the presence of faults. Rejecting the null hypothesis means detecting a fault, with a probability α of making a type I error.

The ***p*-value** is the probability that a given result would occur under the null hypothesis. If the p-value is smaller than the significance level, then the null hypothesis is rejected.

Normal distribution Given a normal distribution $X \sim \mathcal{N}(\mu, \sigma^2)$, in the case of a two-tailed test we want to find $x_{\alpha/2} > 0$ such that $P\{|X| > x_{\alpha/2}\} = \alpha$. Recalling that $Z = (X - \mu)/\sigma^2$, we can find from table C.2 the value $z_{\alpha/2}$ of z, called *z-score*, such that $P\{|Z| < z_{\alpha/2}\} = 1 - \alpha$, and then compute $x_{\alpha/2} = \sigma^2 z_{\alpha/2} + \mu$. If $\alpha = 0.05$ then $z_{0.025} = 1.96$.



Appendix B

Control theory

B.1 Transfer functions

The transfer function of a continuous-time linear time-invariant (LTI) system can be written in two forms:

$$G(s) = \frac{\rho \cdot \prod_i (s + z_i) \cdot \prod_i (s^2 + 2\,\zeta_i\,\alpha_{ni}\,s + \alpha_{ni}^2)}{s^g \cdot \prod_i (s + p_i) \cdot \prod_i (s^2 + 2\,\xi_i\,\omega_{ni}\,s + \omega_{ni}^2)}$$

$$G(s) = \frac{\mu \cdot \prod_i (1 + \tau_i s) \cdot \prod_i (1 + 2\,\zeta_i\,s/\alpha_{ni} + s^2/\alpha_{ni}^2)}{s^g \cdot \prod_i (1 + T_i s) \cdot \prod_i (1 + 2\,\xi_i\,s/\omega_{ni} + s^2/\omega_{ni}^2)}$$

$$\mu = \frac{\rho \cdot \prod_i z_i \cdot \prod_i \alpha_{ni}^2}{\prod_i p_i \cdot \prod_i \omega_{ni}^2}\,, \quad \rho = \frac{\mu \cdot \prod_i \tau_i \cdot \prod_i \omega_{ni}^2}{\prod_i T_i \cdot \prod_i \alpha_{ni}^2}\,, \qquad \tau_i = \frac{1}{z_i}\,, \quad T_i = \frac{1}{p_i}$$

$$(1 + 2\,\zeta_i\,s/\alpha_{ni} + s^2/\alpha_{ni}^2) = \\ = [s + \cos(\xi)\,\omega + j\,\sin(\xi)\,\omega] \cdot [s + \cos(\xi)\,\omega - j\,\sin(\xi)\,\omega]$$

(NOMENCLATURA - INDICARE TIPO g E SIGNIFICATO FDT STRETTAM PROPRIA)

se $g = 0$ e sistema alimentato da ingr costante u-bar e trasf U(s)=u-bar/s, l'uscita y-bar è costante e vale per il teor del val finale

$$\bar{y} = \lim_{t\to\infty} y(t) = \lim_{s\to 0} s\,G(s)\,\frac{\bar{u}}{s} = G(0)\,\bar{u}$$

con riferimento alla tf sopra $\bar{y} = \mu\,\bar{u}$

se $g \neq 0$ si def guadagno generalizzato $\mu = \lim_{s\to 0} s^g\,G(s)$

rho zero-pole gain?

Definition: Minimum phase transfer function (B.1.1). If $L(s)$ has a positive gain and no delay, and it doesn't have singularities (zero and poles) with real part, then it is said to be a minimum phase transfer function.

B.2 Closed loop systems

We consider the closed-loop system shown in fig. B.1, where $L(s)$ is a strictly proper transfer function representing a continuous-time LTI system.

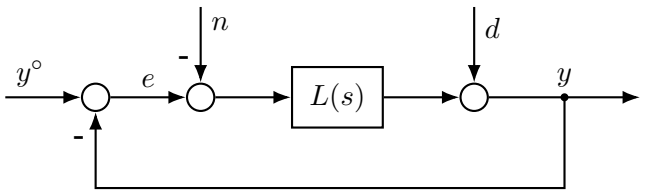


FIGURE B.1: Feedback system

Bode hypotheses We assume that:

1. $L(s)$ doesn't have poles with positive real part;
2. the magnitude plot $|L(j\omega)|_{\text{dB}}$ crosses the axis at 0 dB exactly once.[1]

Definition: Gain cross-over frequency and gain margin (B.2.1). Under the Bode hypotheses, we define:

- the gain cross-over frequency ω_c as the angular frequency such that $|L(j\omega_c)| = 1 = 0\,\text{dB}$;
- the critical margin $\varphi_c \triangleq \angle L(j\omega_c)$;
- the phase margin $\varphi_m \triangleq 180° - |\varphi_c|$.

Theorem: Bode stability criterion (B.2.1). Under the Bode hypotheses, the closed-loop system is asymptotically stable if and only if $\mu_L > 0$ and $\varphi_m > 0°$.

B.3 Low-pass filters

BREVISSIMO Filtri primo e secondo ordine (poli cc)
pag 14

[1] having assumed that $L(s)$ is a strictly proper transfer function, $|L(j\omega)|$ crosses the 0 dB axis with a negative slope.

Appendix C

Tables

n	**A2**	**d2**	**D3**	**D4**	**A3**	**c4**	**B3**	**B4**
2	1.880	1.128	0	3.267	2.659	0.798	0	3.267
3	1.023	1.693	0	2.574	1.954	0.886	0	2.568
4	0.729	2.059	0	2.282	1.628	0.921	0	2.266
5	0.577	2.326	0	2.114	1.427	0.94	0	2.089
6	0.483	2.534	0	2.004	1.287	0.952	0.030	1.970
7	0.419	2.704	0.076	1.924	1.182	0.959	0.118	1.882
8	0.373	2.847	0.136	1.864	1.099	0.965	0.185	1.815
9	0.337	2.970	0.184	1.816	1.032	0.969	0.239	1.761
10	0.308	3.078	0.223	1.777	0.975	0.973	0.284	1.716
11	0.285	3.173	0.256	1.744	0.927	0.975	0.321	1.679
12	0.266	3.258	0.283	1.717	0.886	0.978	0.354	1.646
13	0.249	3.336	0.307	1.693	0.850	0.979	0.382	1.618
14	0.235	3.407	0.328	1.672	0.817	0.981	0.406	1.594
15	0.223	3.472	0.347	1.653	0.789	0.982	0.428	1.572
16	0.212	3.532	0.363	1.637	0.763	0.983	0.448	1.552
17	0.203	3.588	0.378	1.622	0.739	0.985	0.466	1.534
18	0.194	3.640	0.391	1.608	0.718	0.985	0.482	1.518
19	0.187	3.689	0.403	1.597	0.698	0.986	0.497	1.503
20	0.180	3.735	0.415	1.585	0.680	0.987	0.510	1.490
21	0.173	3.778	0.425	1.575	0.663	0.988	0.523	1.477
22	0.167	3.819	0.434	1.566	0.647	0.988	0.534	1.466
23	0.162	3.858	0.443	1.557	0.633	0.989	0.545	1.455
24	0.157	3.895	0.451	1.548	0.619	0.989	0.555	1.445
25	0.153	3.931	0.459	1.541	0.606	0.990	0.565	1.435

TABLE C.1: Anti-biasing parameters for $\bar{X}$-R and $\bar{X}$-s charts

z	**0.00**	**0.01**	**0.02**	**0.03**	**0.04**	**0.05**	**0.06**	**0.07**	**0.08**	**0.09**
0.0	0.00000	0.00399	0.00798	0.01197	0.01595	0.01994	0.02392	0.02790	0.03188	0.03586
0.1	0.03983	0.04380	0.04776	0.05172	0.05567	0.05962	0.06356	0.06749	0.07142	0.07535
0.2	0.07926	0.08317	0.08706	0.09095	0.09483	0.09871	0.10257	0.10642	0.11026	0.11409
0.3	0.11791	0.12172	0.12552	0.12930	0.13307	0.13683	0.14058	0.14431	0.14803	0.15173
0.4	0.15542	0.15910	0.16276	0.16640	0.17003	0.17364	0.17724	0.18082	0.18439	0.18793
0.5	0.19146	0.19497	0.19847	0.20194	0.20540	0.20884	0.21226	0.21566	0.21904	0.22240
0.6	0.22575	0.22907	0.23237	0.23565	0.23891	0.24215	0.24537	0.24857	0.25175	0.25490
0.7	0.25804	0.26115	0.26424	0.26730	0.27035	0.27337	0.27637	0.27935	0.28230	0.28524
0.8	0.28814	0.29103	0.29389	0.29673	0.29955	0.30234	0.30511	0.30785	0.31057	0.31327
0.9	0.31594	0.31859	0.32121	0.32381	0.32639	0.32894	0.33147	0.33398	0.33646	0.33891
1.0	0.34134	0.34375	0.34614	0.34849	0.35083	0.35314	0.35543	0.35769	0.35993	0.36214
1.1	0.36433	0.36650	0.36864	0.37076	0.37286	0.37493	0.37698	0.37900	0.38100	0.38298
1.2	0.38493	0.38686	0.38877	0.39065	0.39251	0.39435	0.39617	0.39796	0.39973	0.40147
1.3	0.40320	0.40490	0.40658	0.40824	0.40988	0.41149	0.41308	0.41466	0.41621	0.41774
1.4	0.41924	0.42073	0.42220	0.42364	0.42507	0.42647	0.42785	0.42922	0.43056	0.43189
1.5	0.43319	0.43448	0.43574	0.43699	0.43822	0.43943	0.44062	0.44179	0.44295	0.44408
1.6	0.44520	0.44630	0.44738	0.44845	0.44950	0.45053	0.45154	0.45254	0.45352	0.45449
1.7	0.45543	0.45637	0.45728	0.45818	0.45907	0.45994	0.46080	0.46164	0.46246	0.46327
1.8	0.46407	0.46485	0.46562	0.46638	0.46712	0.46784	0.46856	0.46926	0.46995	0.47062
1.9	0.47128	0.47193	0.47257	0.47320	0.47381	0.47441	0.47500	0.47558	0.47615	0.47670
2.0	0.47725	0.47778	0.47831	0.47882	0.47932	0.47982	0.48030	0.48077	0.48124	0.48169
2.1	0.48214	0.48257	0.48300	0.48341	0.48382	0.48422	0.48461	0.48500	0.48537	0.48574
2.2	0.48610	0.48645	0.48679	0.48713	0.48745	0.48778	0.48809	0.48840	0.48870	0.48899
2.3	0.48928	0.48956	0.48983	0.49010	0.49036	0.49061	0.49086	0.49111	0.49134	0.49158
2.4	0.49180	0.49202	0.49224	0.49245	0.49266	0.49286	0.49305	0.49324	0.49343	0.49361
2.5	0.49379	0.49396	0.49413	0.49430	0.49446	0.49461	0.49477	0.49492	0.49506	0.49520
2.6	0.49534	0.49547	0.49560	0.49573	0.49585	0.49598	0.49609	0.49621	0.49632	0.49643
2.7	0.49653	0.49664	0.49674	0.49683	0.49693	0.49702	0.49711	0.49720	0.49728	0.49736
2.8	0.49744	0.49752	0.49760	0.49767	0.49774	0.49781	0.49788	0.49795	0.49801	0.49807
2.9	0.49813	0.49819	0.49825	0.49831	0.49836	0.49841	0.49846	0.49851	0.49856	0.49861
3.0	0.49865	0.49869	0.49874	0.49878	0.49882	0.49886	0.49889	0.49893	0.49896	0.49900
3.1	0.49903	0.49906	0.49910	0.49913	0.49916	0.49918	0.49921	0.49924	0.49926	0.49929
3.2	0.49931	0.49934	0.49936	0.49938	0.49940	0.49942	0.49944	0.49946	0.49948	0.49950
3.3	0.49952	0.49953	0.49955	0.49957	0.49958	0.49960	0.49961	0.49962	0.49964	0.49965
3.4	0.49966	0.49968	0.49969	0.49970	0.49971	0.49972	0.49973	0.49974	0.49975	0.49976
3.5	0.49977	0.49978	0.49978	0.49979	0.49980	0.49981	0.49981	0.49982	0.49983	0.49983
3.6	0.49984	0.49985	0.49985	0.49986	0.49986	0.49987	0.49987	0.49988	0.49988	0.49989
3.7	0.49989	0.49990	0.49990	0.49990	0.49991	0.49991	0.49992	0.49992	0.49992	0.49992
3.8	0.49993	0.49993	0.49993	0.49994	0.49994	0.49994	0.49994	0.49995	0.49995	0.49995
3.9	0.49995	0.49995	0.49996	0.49996	0.49996	0.49996	0.49996	0.49996	0.49997	0.49997
4.0	0.49997	0.49997	0.49997	0.49997	0.49997	0.49997	0.49998	0.49998	0.49998	0.49998

TABLE C.2: Standard Normal Distribution: $P(0 < Z \leq z)$

Bibliography

[Sus+15] Gian Antonio Susto et al. "Machine learning for predictive maintenance: A multiple classifier approach". In: *IEEE Transactions on Industrial Informatics* 11.3 (2015), pp. 812–820. DOI: 10.1109/TII.2014.2349359.

[HAA+17] Albokhary Mohammed Taher Bashir Hamid, Almoatassem Bellah Khalaf Allah Ahmed Ali, Masaad Shawgy Masaad Ali, et al. "Implementation of Preventive Maintenance Optimization Techniques". PhD thesis. Sudan University of Science and Technology, 2017.

[Bau18] Marco Baur. "A survey of PHM techniques of machinery equipment, with a focus on machine tools". Dec. 19, 2018.

[KG09] Kevin A. Kaiser and Nagi Z. Gebraeel. "Predictive maintenance management using sensor-based degradation models". In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 39.4 (2009), pp. 840–849. DOI: 10.1109/TSMCA.2009.2016429.

[Sca17] Riccardo Scattolini. *Lecture notes from "Safety in Automation Systems" course*. 2017.

[JLB06] Andrew K.S. Jardine, Daming Lin, and Dragan Banjevic. "A review on machinery diagnostics and prognostics implementing condition-based maintenance". In: *Mechanical systems and signal processing* 20.7 (2006), pp. 1483–1510. DOI: 10.1016/j.ymssp.2005.09.012.

[AK12] Rosmaini Ahmad and Shahrul Kamaruddin. "An overview of time-based and condition-based maintenance in industrial application". In: *Computers & Industrial Engineering* 63.1 (2012), pp. 135–149. DOI: 10.1016/j.cie.2012.02.002.

[SFP03] Silvio Simani, Cesare Fantuzzi, and Ron J. Patton. *Model-based fault diagnosis in dynamic systems using identification techniques*. Springer Science & Business Media, 2003. ISBN: 9781849968959. DOI: 10.1007/978-1-4471-3829-7.

[PDZ10] Ying Peng, Ming Dong, and Ming Jian Zuo. "Current status of machine prognostics in condition-based maintenance: a review". In: *The International Journal of Advanced Manufacturing Technology* 50.1-4 (2010), pp. 297–313. DOI: 10.1007/s00170-009-2482-0.

[Com58] Western Electric Company. *Statistical Quality Control handbook*. 2nd ed. Western Electric Co., Inc., 1958.

[Ger98] Janos Gertler. *Fault detection and diagnosis in engineering systems*. Marcel Dekker, 1998. ISBN: 9780824794279.

[CRB01] Leo H. Chiang, Evan L. Russell, and Richard D. Braatz. *Fault detection and diagnosis in industrial systems*. Springer, 2001. ISBN: 9781852333270.

[Sim06] Silvio Simani. *Fault diagnosis of dynamic systems using model-based and filtering approaches*. 2006.

[dKle+13] Johan de Kleer et al. "Fault augmented modelica models". In: *The 24th International Workshop on Principles of Diagnosis*. Citeseer. 2013, pp. 71–78.

[vdLin14] Franciscus van der Linden. "General fault triggering architecture to trigger model faults in Modelica using a standardized blockset". In: *Proceedings of the 10 th International Modelica Conference; March 10-12; 2014; Lund; Sweden*. 96. Linköping University Electronic Press. 2014, pp. 427–436. ISBN: 9789175193809. DOI: 10.3384/ecp14096427.

[Gun+19] Julia Gundermann et al. "The Fault library-A new Modelica library allows for the systematic simulation of non-nominal system behavior". In: *Proceedings of the 2nd Japanese Modelica Conference Tokyo, Japan, May 17-18, 2018*. 148. Linköping University Electronic Press. 2019, pp. 161–168. DOI: 10.3384/ecp18148161.

[Min+14] Raj Minhas et al. "Using fault augmented modelica models for diagnostics". In: *Proceedings of the 10 th International Modelica Conference; March 10-12; 2014; Lund; Sweden*. 96. Linköping University Electronic Press. 2014, pp. 437–445. ISBN: 9789175193809. DOI: 10.3384/ecp14096437.

[Ros04] Sheldon M. Ross. *Introduction to probability and statistics for engineers and scientists*. Elsevier Academic Press, 2004. ISBN: 9780125980579.

Acronyms

d.o.f. degrees of freedom.

ABC Approximate Bayesian Computation.

CBM Condition-based Maintenance.

CM Corrective Maintenance.

CM Condition monitoring.

DAE differential algebraic equation.

FMEA Failure Mode and Effect Analysis.

FMECA Failure Mode Effect and Critical Analysis.

FPM first-principles model.

LCL Lower Control Limit.

LTI linear time-invariant.

MIMO Multi-Input, Multi-Output.

MPC Model Predictive Control.

MSL Modelica Standard Library.

MTTF Mean Time To Failure.

OOP object-oriented programming.

P&ID Piping and Instrumentation Diagram.

PCA Principal Component Analysis.

PDF probability density function.

PdM Predictive Maintenance.

PM Process monitoring.

PSC passive sign convention.

PvM Preventive Maintenance.

RTF Run-to-failure.

RUL Remaining Useful Life.

SPC Statistical Process Control.

TBM Time-based Maintenance.

UCL Upper Control Limit.

WER Western Electric Rules.

List of Figures

List of Tables

Alphabetical Index