Machine-In-The-Loop control optimization

van der Meulen, S.H.

Published: 01/01/2004

Document Version
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the author’s version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
Machine-In-the-Loop Control Optimization: A Literature Survey

S.H. van der Meulen

CTB 534-04-1502
DCT 2004-114

Literature Survey

Supervisors: dr.ir. R.L. Tousain†
prof.ir. O.H. Bosgra‡
prof.dr.ir. M. Steinbuch†

†PHILIPS APPLIED TECHNOLOGIES
DEPARTMENT OF MECHATRONICS TECHNOLOGIES
ACOUSTICS AND CONTROL GROUP

‡EINDHOVEN UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF MECHANICAL ENGINEERING
CONTROL SYSTEMS TECHNOLOGY GROUP

Eindhoven, April 2005
Abstract

Control design is an important element in the development of new motion stage concepts at Philips Applied Technologies. In order to meet the increasing performance requirements, it is often necessary to extend the traditional two-degree-of-freedom control structure. These extensions typically consist of additional control-related functionality, such as disturbance feedforward filters and compensation tables, for instance.

As a result of this increasing flexibility and complexity of the motion control system, it is necessary to develop adequate design and tuning procedures for an increasing number of design parameters. The approaches currently being applied can roughly be divided into manual control design and off-line model-based control design. However, it appears that these methods have several disadvantages. For example, both approaches do not adapt to changes in the system over time.

It is believed that these shortcomings can be overcome by Machine-In-the-Loop control optimization. The main idea behind this approach is to formulate the control design problem as an optimization problem, in which the objective function is evaluated and optimized on the actual system. The objective of the literature survey is to investigate the Machine-In-the-Loop control optimization concept in more detail. For this reason, two questions are formulated.

The first question is to find out which theoretical aspects and building blocks are important with respect to Machine-In-the-Loop control optimization. Three main building blocks are considered. These are discrete time signals and discrete time signal analysis, single input single output linear time-invariant systems and system modeling and, finally, optimization techniques.

The second question is to find out which control techniques, related to Machine-In-the-Loop control optimization, are already available. In this context, four control concepts are considered. These are the self-tuning regulator, model reference adaptive control, iterative feedback tuning and lifted iterative learning control.

It is concluded that none of the four considered control concepts satisfies all desired properties of Machine-In-the-Loop control optimization. Furthermore, each concept provides a solution to a specific control design problem, thereby imposing specific assumptions. Nevertheless, it is observed that three control concepts make use of optimization techniques in order to solve the problem. Hence, a recommendation for future research is to investigate whether a general approach, denoted Machine-In-the-Loop control optimization, can be developed from an optimization point of view. Hereby, it is desired that the general approach results in algorithms that have several appealing properties.
## Contents

Abstract iii

Contents v

Abbreviations and Acronyms ix

1 Introduction 1
   1.1 Background ........................................ 1
   1.2 Motivation ........................................ 2
   1.3 Problem Statement .................................. 4
   1.4 Approach and Overview ............................. 4

2 Discrete Time Signals 7
   2.1 Introduction ........................................ 7
   2.2 Sampling Continuous Time Signals .................. 7
   2.3 Signal Analysis ..................................... 9
      2.3.1 Deterministic Signals .......................... 9
      2.3.2 Discrete Fourier Transform .................... 11
      2.3.3 Stochastic Processes ........................... 12
   2.4 Signal Norms ....................................... 14

3 Linear Time-Invariant Systems 17
   3.1 Introduction ........................................ 17
   3.2 Impulse Responses ................................... 17
   3.3 Disturbances ....................................... 18
   3.4 Transfer Functions .................................. 19
   3.5 Frequency Function .................................. 21
   3.6 Quasi-Stationarity .................................. 21
   3.7 Transformation of Signal Spectra ................... 23
   3.8 System Norms ....................................... 24
      3.8.1 Continuous Time Case ........................... 24
      3.8.2 Discrete Time Case ............................. 25

4 Prediction 27
   4.1 Introduction ........................................ 27
   4.2 One-Step-Ahead Prediction of $v$ .................... 27
   4.3 One-Step-Ahead Prediction of $y$ .................... 28
4.4 $k$-Step-Ahead Prediction of $y$ ........................................... 29

5 Models of Linear Time-Invariant Systems 31
  5.1 Introduction .................................................. 31
  5.2 Linear Models and Sets of Linear Models ......................... 31
  5.3 Transfer Function Models ...................................... 32
    5.3.1 Equation Error Model Structure ........................... 32
    5.3.2 Linear Regressions ...................................... 33
    5.3.3 Additional Equation-Error-Type Model Structures ......... 34
    5.3.4 Output Error Model Structure ............................ 35
    5.3.5 General Model Structure .................................. 36

6 Parameter Estimation Methods 39
  6.1 Introduction .................................................. 39
  6.2 Minimization of Prediction Errors .............................. 39
  6.3 Criterion Function Minimization Approach ....................... 40
    6.3.1 Least Squares Method ................................... 41
    6.3.2 Maximum Likelihood Method .............................. 44
  6.4 Correlation Approach ......................................... 45
    6.4.1 Instrumental Variable Method ............................ 46
  6.5 Excitation Conditions ........................................ 48
    6.5.1 Persistent Excitation .................................... 48

7 Optimization Techniques 51
  7.1 Introduction .................................................. 51
  7.2 Fundamentals of Optimization .................................. 51
    7.2.1 Generalized Optimization Problem ......................... 51
    7.2.2 Feasibility ............................................. 52
    7.2.3 Optimality ............................................. 53
    7.2.4 Convexity ............................................. 54
    7.2.5 Generalized Optimization Algorithm ....................... 56
  7.3 Unconstrained Optimization .................................... 57
    7.3.1 Newton's Method for Nonlinear Equations ................ 57
    7.3.2 Optimality Conditions .................................. 59
    7.3.3 Newton's Method for Minimization ......................... 60
    7.3.4 Search Direction ...................................... 61
    7.3.5 Line Search ........................................... 62
    7.3.6 Steepest Descent Method ................................. 64
    7.3.7 Quasi-Newton Methods .................................. 65

8 Self-Tuning Regulator 69
  8.1 Introduction .................................................. 69
  8.2 General Description ......................................... 69
  8.3 Pole Placement Design ....................................... 71
  8.4 Minimum Variance Control .................................... 74
# CONTENTS

9 Model Reference Adaptive Control
- 9.1 Introduction ........................................ 77
- 9.2 General Description ................................. 77
- 9.3 Gradient Based Approach ......................... 78
- 9.4 Stability Based Approach ......................... 79

10 Iterative Feedback Tuning .......................... 83
- 10.1 Introduction ........................................ 83
- 10.2 One-Degree-of-Freedom Control ............... 84
  - 10.2.1 Control Design Criterion ..................... 84
  - 10.2.2 Criterion Minimization ....................... 85
- 10.3 Two-Degree-of-Freedom Control ............... 88
  - 10.3.1 Control Design Criterion ..................... 88
  - 10.3.2 Criterion Minimization ....................... 89
- 10.4 Implementation Issues and Design Choices ..... 92
  - 10.4.1 Convergence .................................. 92
  - 10.4.2 Nonminimum Phase Controllers .......... 93
  - 10.4.3 Design Choices Iterative Algorithm ...... 93
  - 10.4.4 Design Parameters ......................... 94

11 Lifted Iterative Learning Control ................. 97
- 11.1 Introduction ........................................ 97
- 11.2 Lifted System Representation .................... 98
- 11.3 Control Design Problem ......................... 100
- 11.4 Internal Model Principle and Convergence .... 100
- 11.5 Learning Matrix ................................ 102
  - 11.5.1 Multi-Objective Design Method .......... 102

12 Evaluation of Control Concepts .................. 105
- 12.1 Introduction ........................................ 105
- 12.2 Evaluation ......................................... 105

13 Conclusions and Recommendations ............... 109
- 13.1 Conclusions ........................................ 109
- 13.2 Recommendations ................................ 110

A Iterative Feedback Tuning: Gradient Computations 113
- A.1 Process Output Gradient ......................... 113
- A.2 Process Input Gradient ......................... 114

Bibliography ........................................... 115

Distribution List ...................................... 119
Abbreviations and Acronyms

AR AutoRegressive
BFGS Broyden Fletcher Goldfarb Shanno
BIBO Bounded Input Bounded Output
BJ Box-Jenkins
D/A Digital-to-Analog
DFT Discrete Fourier Transform
DTFT Discrete Time Fourier Transform
FIR Finite Impulse Response
FRF Frequency Response Function
IC Integrated Circuit
IFT Iterative Feedback Tuning
ILC Iterative Learning Control
IV Instrumental Variable
LQG Linear Quadratic Gaussian
LQR Linear Quadratic Regulator
LSE Least Squares Estimate
LTI Linear Time-Invariant
MA Moving Average
MAP Maximum A Posteriori
MIL Machine-In-the-Loop
MIMO Multi Input Multi Output
ML Maximum Likelihood
MLE Maximum Likelihood Estimator
MRAC Model Reference Adaptive Control
MV Minimum Variance
OE Output Error
PDF Probability Density Function
PE Persistently Exciting
PEM Prediction Error Method
PID Proportional Integral Derivative
RMS Root Mean Square
SISO Single Input Single Output
STR Self-Tuning Regulator
X eXternal input
Chapter 1

Introduction

1.1 Background

At Philips Applied Technologies\(^1\), the development of state-of-the-art motion stage concepts for application in advanced production systems is one of the main activities. A typical example of such an advanced production system is the ASML TwinScan wafer scanner, see [2], which is used for mass production of integrated circuits (IC’s). More specifically, the wafer scanner is responsible for the lithographic printing of the IC pattern onto a silicon disk, the so-called wafer. In order to do this, use is made of a so-called reticle, containing the IC pattern. The reticle is placed on a motion stage, the reticle stage, that is positioned above an advanced lens system. The silicon disk, containing a light-sensitive layer, is placed on a motion stage as well, the so-called wafer stage. This wafer stage is positioned below the lens system. In this way, it is possible to project the IC pattern onto the wafer by making use of a short wavelength light source. Due to the fact that a complete IC consists of several layers with different patterns, this procedure needs to be performed multiple times for one wafer. Moreover, only a small part of the entire wafer can be exposed at the same time, due to limitations of the lens system. Consequently, in order to realize high throughputs it is required that the reticle stage and the wafer stage are positioned as fast as possible. In addition, it is necessary that both stages are positioned sufficiently accurate in order to project each pattern exactly on top of the previous pattern. Furthermore, during the exposure it is required that the motion stages move as little as possible. If these requirements are not fulfilled, this will result in a faulty IC.

Obviously, a dedicated control system design is of major importance in order to meet these performance requirements. Often, it appears that the traditional two-degree-of-freedom control structure, consisting of feedback and feedforward control, is insufficient and additional control-related functionality is required. In this context, one can think of setpoint feedforward tables, disturbance feedforward filters and cogging tables, for instance. However, the increasing flexibility and complexity of these software-based motion control systems calls for adequate design and tuning procedures. A familiar approach to control design is manual loopshaping. Manual loopshaping amounts to making suitable series connections of basic filters and tuning the parameters based on Nyquist and Bode plots, using measured frequency response functions (FRF’s) of the process. However, this method does not guarantee that the optimal controller is found for the intended goal and does not incorporate adaptation mechanisms. Moreover, manual tuning becomes infeasible if the control problem becomes

\(^1\)Formerly Philips Centre for Industrial Technology (CFT).
more complicated and is time-consuming, since besides the feedback controller many other parameters need to be tuned. To make the design process of feedback and feedforward filters more systematic and to achieve a higher performance, sometimes use is made of $H_{\infty}$ control optimization and $\mu$-synthesis. In these methods, the controller is the solution of an optimization problem. A process model and frequency-dependent weighting filters are used to translate the actual control problem into a meaningful mathematical problem that can be solved by available algorithms. The main disadvantage of these methods is the requirement of an accurate process model. The determination of a process model is often time-consuming and expensive and the model will only be an approximation of the actual system. Furthermore, it is hard to see how such methods could be used to optimize other crucial elements of the motion control system, such as static decoupling matrices and disturbance compensation tables. Hence, it can be concluded that the methods currently being used in the design of control systems for motion stages have several disadvantages.

1.2 Motivation

In order to overcome the mentioned disadvantages, the use of Machine-In-the-Loop (MIL) control optimization is an intuitive and appealing approach. The main idea behind this approach to designing and/or tuning controllers and control-related functions in motion systems is to formulate the control design problem as an optimization problem, in which the objective function is evaluated and optimized on the real system. This is a rather general concept, which can be used to optimize many different elements of a typical motion control system. One can think of the feedback controller, the feedforward controller and coggings tables, for instance. Moreover, an attractive feature of the Machine-In-the-Loop control optimization approach is that it can easily be extended to adaptive schemes, where the control parameters are continuously, or discretely, adapted to slow changes in the motion system dynamics. In order to demonstrate the intended nature of Machine-In-the-Loop control optimization, an example problem is considered.

Example 1. Consider the block diagram of a motion system with feedback and feedforward control, depicted in Figure 1.1. Here, $P$ denotes the process, $K_{fb}$ denotes the feedback controller and $K_{ff}$ denotes the feedforward controller. The reference signals are given by the position $r$, the velocity $v$, the acceleration $a$, the jerk $j$ and the snap$^2$ $s$. Here, the velocity, the acceleration, the jerk and the snap are the first, second, third and fourth order derivative of the position, respectively. Furthermore, $u$ is the process input, $y$ is the process output, $e$ is the tracking error and $ff$ is the feedforward signal. In this case, the feedforward signal consists of four independent parts. Each part is constructed by multiplication of the constant scalar feedforward coefficient and its corresponding reference signal.

The low-order feedforward controller depicted in Figure 1.1 is well-known and widely used in industry. The main advantages of low-order feedforward control are its simplicity, its ease of implementation and its flexibility with respect to application of different motion profiles. Disadvantages of low-order feedforward control are the time and effort spent on tuning the feedforward coefficients. This is mainly done by manual tuning, which implies that optimal performance is not guaranteed. Another disadvantage of low-order feedforward control is that the controller does not adapt to changes in the motion system dynamics. This may lead to a substantial decrease in performance.

$^2$Snap is an equivalent name for derivative of jerk.
1.2 Motivation

Figure 1.1: Block diagram of a motion system with feedback and feedforward control.

It is obvious that Machine-In-the-Loop control optimization can be used to overcome these disadvantages of low-order feedforward control. One of the basic properties of Machine-In-the-Loop control optimization is that the feedforward coefficients are updated in an iterative way, which makes it possible to adapt to slow changes in the motion system dynamics. The actual update is determined by using an objective function, typically defined as the sum of squares of the tracking errors. Minimization of the objective function to the feedforward coefficients leads to an expression for the update of the coefficients that is evaluated by making use of measurement data.

Hence, it can be concluded that application of Machine-In-the-Loop control optimization to tuning of the feedforward coefficients has several advantages. Due to the fact that measurement data is used, it is possible to adapt to changes in the motion system dynamics. Furthermore, optimization of an objective function guarantees that optimal feedforward control performance is achieved without manual tuning.

With this example problem in mind, it is attempted to formulate a general definition of Machine-In-the-Loop control optimization.

Definition 1 (Machine-In-the-Loop control optimization). Machine-In-the-Loop control optimization is a control approach in which a certain fixed control structure is defined that incorporates one or more control parameters $\theta$, the values of which are unknown. The values of the control parameters $\theta$ are determined through on-line optimization of a certain objective function, where use is made of a certain optimization algorithm.

From Example 1 and Definition 1, it follows that Machine-In-the-Loop control optimization is characterized by several specific properties. In addition to these specific properties, it is desired that the Machine-In-the-Loop control optimization approach incorporates several additional appealing properties. An overview of these properties is given next. This overview is useful with respect to the evaluation of the control concepts that are related to Machine-In-the-Loop control optimization. Remark that the Machine-In-the-Loop control optimization concept and its properties are based on discussions between and with [34], [8] and [33].
• Adaptation to changes in motion system dynamics
• Generic method with large application area, e.g., feedforward control design and machine calibration
• Optimal performance
• Automated procedure
• Application of optimized controller to variety of tasks (and not just the tasks that are performed during optimization)
• Demonstrable / guaranteed convergence
• Model knowledge limited to simple, ‘cheap’ models

1.3 Problem Statement

The problem statement of the literature survey is as follows:

Make a critical overview of Machine-In-the-Loop control optimization methodologies described in literature and a comprehensive description of the most relevant theoretical aspects to and building blocks for Machine-In-the-Loop control optimization.

Taking this problem statement into account, two questions can be distinguished with respect to the literature survey:

• What are the most relevant theoretical aspects to and building blocks for Machine-In-the-Loop control optimization that are available in related fields?
• Which related control techniques are already available and to what extent are they useful with respect to Machine-In-the-Loop control optimization?

1.4 Approach and Overview

The report is organized as follows. Part I (Chapters 2 to 7) is devoted to the first question, while the second question is answered in Part II (Chapters 8 to 12). Several parts of the contents will be motivated by referring to Example 1 in Section 1.2.

In Chapter 2, discrete time signals and discrete time signal analysis are discussed. The discussion of continuous time signals is largely omitted, since all of signal processing performed by computers has to be done in discrete time. In view of the example problem, signals and signal analysis are important with respect to the measurement of the tracking error and the determination of a mathematical model of the system based on observed data (signals) from the system.

The last mentioned item, the determination of a mathematical model of the dynamical system, also requires knowledge about systems and system modeling. This is the subject of Chapter 3. Only linear time-invariant systems are considered, due to the fact that they form the most important class of dynamical systems considered in theory and practice.
An important property with respect to the evaluation of models as representatives of systems is the capability of a model to predict future output values. For this reason, an analysis is performed in Chapter 4 how general models can be used for prediction purposes.

So far, only general model descriptions have been considered. In order to obtain an adequate model description of the dynamical system, it is necessary to parameterize this general model set. There are several different ways of parameterizing the model set, leading to different classes of models, also called model structures. In Chapter 5, several classes of models for linear time-invariant systems are discussed.

After parameterization of the model set, a model structure is obtained that incorporates a finite number of coefficients. These coefficients often cannot be determined a priori and, thus, enter the model structure as unknown parameters. In order to determine the unknown coefficients, use is made of parameter estimation methods, which is the subject of Chapter 6. In view of the example problem, parameter estimation methods are relevant with respect to the determination of the feedforward coefficients.

In many control concepts that are related to Machine-In-the-Loop control optimization, it appears that the control parameters are updated in such a way that a certain objective function is minimized. This also follows from the example problem. In order to adequately solve the minimization problem, it is necessary to have a basic knowledge of optimization techniques. These fundamentals of optimization are discussed in Chapter 7.

Four control techniques that are related to Machine-In-the-Loop control optimization are investigated. Each technique will now be discussed briefly, in order to reflect the relation with Machine-In-the-Loop control optimization.

The self-tuning regulator (STR) is considered in Chapter 8. This is an adaptive control method that updates the control parameters by first estimating the process parameters and subsequently solving a control design problem, using the estimated process parameters according to the certainty equivalence principle.

Another adaptive control method is given by model reference adaptive control (MRAC). The objective of this method is to minimize the difference between the output of the adaptive control system and the output of a specific reference model. To this end, an adaptation mechanism is used to adjust the control parameters, which is discussed in Chapter 9.

In Chapter 10, iterative feedback tuning (IFT) is considered. This is a gradient based iterative approach to tuning a fixed control structure by using measurement data only, in such a way that a closed loop performance criterion is minimized.

Lifted iterative learning control (ILC) is an iterative approach to feedforward tuning, which is discussed in Chapter 11. It updates the feedforward signal for a repeating setpoint following task iteratively, in such a way that the tracking error decreases.

An evaluation is performed in Chapter 12, in order to investigate the relation between these control concepts and Machine-In-the-Loop control optimization in more detail. To this end, criteria are defined, where the desired properties of Machine-In-the-Loop control optimization from Section 1.2 are taken as a starting point. Evaluation of these criteria gives more insight to what extent the related control concepts are useful with respect to Machine-In-the-Loop control optimization.

Finally, several conclusions and recommendations are given in Chapter 13.
Chapter 2

Discrete Time Signals

2.1 Introduction

Many subjects in the field of systems and control engineering are concerned with signals and signal analysis. In this context, one can think of system identification [37] and adaptive control [7], for instance. Due to this importance, several basic concepts with respect to signals and signal analysis are discussed in this chapter.

A signal is a phenomenon, arising in some environment, that changes over time due to external causes. These changes over time provide information that can be of interest. Consequently, the analysis of a signal in order to obtain this information is referred to as signal analysis, see [36, Section 1.1].

Signals can be classified according to various criteria, see [18, Section 1.1]. In this survey, attention is focused on discrete time signals, since all of signal processing performed by computers has to be done in discrete time. Moreover, specific attention will be given to both deterministic and stochastic signals.

2.2 Sampling Continuous Time Signals

Many physical signals, such as electrical voltages produced by measuring devices, are essentially continuous time signals. Computers and related devices operate in a digital1 (discrete) environment. Therefore, continuous time signals that are to be processed by such devices first need to be converted to discrete time signals. One way of doing this is sampling.

In this context, the sampling and reconstruction theorem of Shannon will be discussed in this section. It considers the question under which conditions a sampled signal can exactly represent an infinite time continuous time signal from which it is constructed. Basically, the operation of sampling a continuous time signal $x_c(t)$ consists of observing the signal value during short time intervals $\tau$. In this case, only so-called equidistant sampling is considered, which implies that the time intervals between the samples are constant and equal to $T$. In order to simplify the analysis, it is assumed that $\tau \ll T$, which is a reasonable assumption, see [36, Section 3.1]. Moreover, it is assumed that the value of $x_c(t)$ is averaged over the time interval $\tau$, in order to attenuate the influence of noise. This leads to a sampled sequence $x_d(k) = x_c(kT)$, which is a discrete sequence defined for integer values $k \in \mathbb{Z}$, see [36, Section 3.2].

1In this respect, also the notion of quantization is of interest, see [24, Section 2.3].
As a first step, the Fourier transforms of the continuous time and discrete time signals are related to each other, in order to investigate the consequences of sampling a continuous time signal. A profound treatment of the corresponding derivation is given in [36, Section 3.1] and [37, Section 2.4.3]. The result, under the condition that \( x_d(k) = x_c(kT) \), is given by the following relation:

\[
X_d(\omega) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c \left( \omega - k \frac{2\pi}{T} \right). \tag{2.1}
\]

From (2.1), it can be concluded that the Fourier transform \( X_d(\omega) \) of the sampled signal \( x_d(k) \) is constructed as a shifted summation of the Fourier transform \( X_c(\omega) \) of the continuous time signal \( x_c(t) \). Consequently, the Fourier transform \( X_d(\omega) \) becomes periodic in the frequency domain with period \( \omega_s = \frac{2\pi}{T} \). Moreover, the summation is multiplied by \( \frac{1}{T} \), leading to a different scaling of the amplitude spectrum.

The fact that the Fourier transform \( X_d(\omega) \) becomes periodically repeated, is directly related to the possibility for reconstructing the original signal \( x_c(t) \) from its sampled version \( x_d(k) \). Obviously, loss of information occurs in case the shifted versions of \( X_c(\omega) \) are folded on top of each other in \( X_d(\omega) \). This effect is called aliasing and in this situation it is not possible to reconstruct \( x_c(t) \) from \( x_d(k) \). Aliasing can be avoided by using a sampling frequency \( \omega_s \), which is larger than two times the maximum frequency \( \omega_{\text{max}} \) present in the signal \( x_c(t) \). This is formally stated by the sampling theorem of Shannon:

\[
\omega_s > 2\omega_{\text{max}}. \tag{2.2}
\]

If this relation holds, \( X_c(\omega) \) can be exactly recovered from \( X_d(\omega) \) by applying a low-pass filter to \( x_d(k) \) that exactly extracts the frequency region \( [0, \frac{\omega_s}{2}] \). The frequency \( \frac{\omega_s}{2} \) is also referred to as the Nyquist frequency. Furthermore, continuous time signals \( x_c(t) \) whose frequency content is zero outside a given interval \( [0, \omega_{\text{max}}] \) are also called band-limited signals, see [24, Section 9.2].

Summarizing the results, it can be stated that in case of sampling a continuous time signal, one has to take care that the signal that is sampled does not contain any frequency components exceeding the Nyquist frequency \( \frac{\omega_s}{2} \). Only in case this condition is satisfied, all information of the continuous time signal can be recovered from the sampled signal. For this reason, often use is made of continuous time anti-aliasing filters before a sampling operation.

Next, attention is paid to the reconstruction of a continuous time signal from its sampled signal. It is assumed that the condition (2.2) is satisfied. As a result, it is possible to determine \( X_c(\omega) \) by application of a low-pass filter \( H(\omega) \):

\[
X_c(\omega) = TX_d(\omega)H(\omega), \tag{2.3}
\]

where the multiplication by \( T \) is necessary to obtain the correct amplitude spectrum. The low-pass filter \( H(\omega) \) is defined by:

\[
H(\omega) = \begin{cases} 
1 & -\frac{\omega_s}{2} \leq \omega \leq \frac{\omega_s}{2} \\
0 & |\omega| > \frac{\omega_s}{2}
\end{cases}. \tag{2.4}
\]

In practice, the construction of such a filter is not straightforward, see [36, Section 3.2]. Remark that the filter \( H(\omega) \) is anti-causal, which implies that on-line application is not feasible.
2.3 Signal Analysis

Subsequently, it is possible to obtain the signal $x_c(t)$ from (2.3). A detailed description of the corresponding procedure can be found in [36, Section 3.2] and [37, Section 2.4.3]. The result is given by the following relation\(^2\):

$$ x_c(t) = \sum_{k=-\infty}^{\infty} x_c(kT) \text{sinc} \left( \frac{\omega_s}{2} (t - kT) \right). \quad (2.5) $$

Expression (2.5) shows how the continuous time signal $x_c(t)$ can be computed from the corresponding sampled signal $x_c(kT)$. Note that the computation requires an infinite number of data points in order to obtain the underlying continuous time signal, even in the case it takes values during a finite time period only. More details on this subject can be found in [36, Chapter 3].

2.3 Signal Analysis

Almost all practical applications of signal analysis and signal processing are concerned with discrete time signals, as a result of the growing importance of digital computers. For this reason, it is explained in the previous section how discrete time signals are obtained from continuous time signals through sampling. In order to analyze the obtained discrete time signals, a theoretical framework is required, which will be discussed in this section. One can think of the decomposition of a signal into its basic harmonics, the evaluation of the frequency content of a signal and the evaluation of the distribution of energy and/or power over frequency. A distinction will be made between deterministic signals and stochastic processes.

2.3.1 Deterministic Signals

From Section 2.2, it follows that for discrete time signals the notation $x_d(k) = x_c(kT)$, $k = 1, 2, \ldots$ is adopted. Here, $x_c(t)$ is the continuous time signal that possibly underlies the discrete time signal and $T$ is the sampling period.

In specifying relevant signal properties, the energy and power of a signal are considered.

**Definition 2 (Energy).** The energy of a signal is defined by:

$$ E_x = \sum_{k=-\infty}^{\infty} x_d^2(k). \quad (2.6) $$

In case $E_x$ is finite, the signal has finite energy and is referred to as an energy signal.

**Definition 3 (Power).** The power of a signal is defined by:

$$ P_x = \lim_{N \to \infty} \frac{1}{2N + 1} \sum_{k=-N}^{N} x_d^2(k). \quad (2.7) $$

In case this limit exists, the signal is referred to as a power signal.

\(^2\)The sinc function is defined by $\text{sinc}(t) = \frac{\sin(t)}{t}$ for $t \neq 0$ and $\text{sinc}(0) = 1$. 
In the analysis of the frequency content of a discrete time signal, the decomposition of arbitrary discrete time signals into a number of harmonic components is a very important tool. This basic tool is referred to as discrete time Fourier analysis. In case pure periodic signals are considered, the Fourier series can be used to show that any periodic signal can be written as a summation of harmonic functions (sine waves). In practice, however, one normally has to deal with nonperiodic signals, such as transient signals or steady state signals. The Fourier transform is a generalization of the Fourier series that can also handle these nonperiodic signals.

**Definition 4 (Discrete time Fourier series).** Let \( x_d(k) \) be a periodic signal, which means that \( x_d(k+N) = x_d(k) \) for all \( k \in \mathbb{Z} \), with \( N \) the (smallest) period of the signal. The discrete time Fourier series (complex form) of the signal \( x_d(k) \) is then given by:

\[
\begin{align*}
x_d(k) &= \sum_{n=0}^{N-1} c_ne^{i2\pi k^n} \\
c_n &= \frac{1}{N} \sum_{k=0}^{N-1} x_d(k)e^{-i2\pi k/n},
\end{align*}
\]

with \( c_n \) the so-called Fourier coefficients. It can be shown that the Fourier coefficients are periodic as well: \( c_{n+iN} = c_n \) for \( i \in \mathbb{N} \). More detailed information with respect to the discrete time Fourier series can be found in [36, Section 4.1].

**Definition 5 (Discrete time Fourier transform).** The discrete time Fourier transform (DTFT) for a sampled (discrete time), possibly nonperiodic signal \( x_d(k) \) is given by the following Fourier transform pair:

\[
\begin{align*}
X_d(\omega) &= \sum_{k=-\infty}^{\infty} x_d(k)e^{-i\omega kT} \\
x_d(k) &= \frac{T}{2\pi} \int_{-\pi}^{\pi} X_d(\omega)e^{i\omega kT} d\omega.
\end{align*}
\]

Relation (2.10) transforms a discrete sequence \( x_d(k) \) into a continuous, complex function of a real variable \( \omega \). The transform \( X_d(\omega) \), since \( k \) is integer-valued, is a periodic function with period \( \omega_s = \frac{2\pi}{T} \). Consequently, the integral in (2.11) is taken over any range of \( \omega \) with length \( \frac{2\pi}{T} \), being the period length of the integrand. Recall that an infinite sequence \( x_d(k) \) is required in order to compute the transform \( X_d(\omega) \) in (2.10). Moreover, a sufficient condition to show the existence of the spectrum \( X_d(\omega) \) is that the sum of the discrete sequence is finite: \( \sum_k |x_d(k)| < \infty \). More information can be found in [18, Section 12.3].

The energy/power spectral density of a signal characterizes the distribution of energy/power of a signal over the different frequency components. These spectral densities are naturally obtained by utilizing Parseval’s relationship for energy signals:

\[
\sum_{k=-\infty}^{\infty} x_d^2(k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} |X_d(\omega)|^2 d\omega.
\]

The Parseval relationship states that the energy of a signal \( x_d(k) \), defined here by summation of \( x_d^2(k) \), can also be found by integrating with respect to \( |X_d(\omega)|^2 \) in the frequency domain, see [18, Section 12.5.5]. This directly leads to the following definition of the energy spectral density function.
2.3 Signal Analysis

Definition 6 (Energy spectral density function). Let \( x_d(k) \) be a finite energy sampled (discrete time) signal, sampled with sampling interval \( T \). Then, the energy \( E_x \) of the signal is given by:

\[
E_x = \frac{T}{2\pi} \int_{-\pi}^{\pi} \Psi_x(\omega) d\omega
\]

(2.13)

\[
= \frac{T}{2\pi} \int_{-\pi}^{\pi} |X_d(\omega)|^2 d\omega,
\]

(2.14)

where \( \Psi_x(\omega) \) is referred to as the energy spectral density.

With the same line of reasoning, a similar expression can be given for the situation of power signals.

Definition 7 (Power spectral density function). Let \( x_d(k) \) be a finite power sampled (discrete time) signal, sampled with sampling interval \( T \). Then, the power \( P_x \) of the signal is given by:

\[
P_x = \frac{T}{2\pi} \int_{-\pi}^{\pi} \Phi_x(\omega) d\omega
\]

(2.15)

\[
= \frac{T}{2\pi} \int_{-\pi}^{\pi} \frac{1}{N} |X_N(\omega)|^2 d\omega,
\]

(2.16)

where \( \Phi_x(\omega) \) is referred to as the power spectral density. Details can be found in [37, Appendix 2].

For finite power signals, the quantity \( \frac{1}{N} |X_N(\omega)|^2 \) is referred to as the periodogram of the finite time discrete time signal. This periodogram determines the distribution of power over frequency.

2.3.2 Discrete Fourier Transform

In Definition 5, the discrete time Fourier transform (DTFT) is introduced. It is also remarked that an infinite sequence \( x_d(k) \) is required for the computations. However, in most situations only finite time signals are available. For this reason, attention is focused on finite time sequences in this section.

Consider the finite sequence \( x_d(k) \), for \( k = 0, 1, \ldots, N - 1 \). The Fourier transform pair corresponding to this finite interval signal is given by:

\[
X_N(\omega) = \sum_{k=0}^{N-1} x_d(k)e^{-i\omega k T}
\]

(2.17)

\[
x_d(k) = \frac{1}{N} \sum_{n=0}^{N-1} X_N \left( \omega \frac{n}{N} \right) e^{i \omega \frac{n}{N} k},
\]

(2.18)

which actually is a finite time DTFT. For a proof, the reader is referred to [37, Appendix 2].

In many situations, discrete time signals are analyzed without taking into account that they are obtained by sampling continuous time signals, see [37, Section 2.4.31]. This implies

\[\text{Here, } X_N(\omega) \text{ is the notation for finite time sequences, defined according to } X_N(\omega) = \sum_{k=0}^{N-1} x_d(k)e^{-i\omega k T}.\]
that in this case the expressions for the Fourier transform pair can be simplified by substitution of $T = 1$:

$$
X_N(\omega) = \sum_{k=0}^{N-1} x_d(k) e^{-i\omega k}
$$

(2.19)

$$
x_d(k) = \frac{1}{N} \sum_{n=0}^{N-1} X_N \left( \frac{2\pi n}{N} \right) e^{i\frac{2\pi n}{N} k}.
$$

(2.20)

From this Fourier transform pair, it can be concluded that $X_N(\omega)$ takes its values on a continuous region of $\omega$, while only $N$ discrete values of $X_N$ are necessary for reconstruction of the original signal $x_d(k)$. Furthermore, it is clear that $X_N(\omega)$ is periodic with period $2\pi$. The values of $X_N(\omega)$ obtained for $\omega = \frac{2\pi n}{N}$, $n = 1, \ldots, N$, form the well-known discrete Fourier transform (DFT) of the sequence $x_d(k)$, $k = 0, 1, \ldots, N-1$. A detailed analysis can be found in [36, Section 4.1]. Obviously, the DFT constitutes a one-to-one mapping from an $N$-length sequence of time domain samples to an $N$-length sequence of frequency domain samples. From the inverse DFT, defined by (2.20), it follows that the reconstructed signal $x_d(k)$ also is defined outside the interval $[0, \ldots, N-1]$. Actually, the inverse DFT induces a periodic extension of the original sequence $x_d(k)$, resulting in a reconstructed signal that is periodic with period $N$. Also, since $x_d(k)$ is real, it holds that $X_N(-\omega) = \overline{X_N(\omega)}$, where the overbar denotes the complex conjugate. As a result, the Fourier transform $X_N(\omega)$ is uniquely defined by its values over the interval $[0, \pi]$. This implies that the one-to-one mapping between time and frequency domain actually takes place between $N$ real-valued time domain samples and $\frac{N}{2}$ complex-valued frequency domain samples.

### 2.3.3 Stochastic Processes

One of the main reasons for using control is the presence of disturbances. Different types of disturbances can be distinguished, such as load disturbances, which influence the process variables and measurement errors, which enter in the sensors. In order to describe disturbances, extensive use is made of stochastic, or random, concepts. By such an approach, it is possible to describe a wide class of disturbances. For this reason, a brief summary of the theory of stochastic processes is provided in this section. First, some basic concepts of random processes are given, followed by a discussion of discrete time white noise.

#### Theoretical Concepts

A stochastic process, also referred to as random process or random function, can be regarded as a family of stochastic variables $\{x_d(k), k \in \mathbb{Z}\}$. The stochastic variables are indexed with the parameter $k$, which belongs to the set of integer numbers $\mathbb{Z}$, the index set. Consequently, the stochastic process is referred to as a discrete parameter process or a discrete time process. Notice that the sampling period is chosen as the time unit.

A random process may be considered as a function of two variables, denoted by $x_d(k, \zeta)$. For fixed $k = k_0$, the function $x_d(k_0, \cdot)$ is thus a random variable and for fixed $\zeta = \zeta_0$, the function $x_d(\cdot, \zeta_0)$ is a function of time, which is called a realization of the process.

The values of a random process at $n$ distinct times, denoted by $x_d(k_1), \ldots, x_d(k_n)$, are $n$-dimensional random variables. The finite-dimensional distribution function of the random
process is then given by:

\[ F(\xi_1, \ldots, \xi_n; k_1, \ldots, k_n) = P\{x_d(k_1) \leq \xi_1, \ldots, x_d(k_n) \leq \xi_n\}, \tag{2.21} \]

where \( P \) denotes probabilities, see [3, Section 2.2]. A random process is called normal, or Gaussian, if all finite-dimensional distributions are normal, see [17, Section 2.2]. It is well-known that a Gaussian random process is completely characterized by its mean value function and its covariance function.

The mean value function of a random process \( x_d(k) \) is defined by:

\[ m(k) = E x_d(k) \]

\[ = \int_{-\infty}^{\infty} \xi dF(\xi, k). \tag{2.23} \]

It can be concluded that the mean value function is an ordinary function of the parameter \( k \), i.e., the mean value is defined for any value of \( k \). Higher moments are defined in a similar way.

The covariance function of a random process is defined by:

\[ r_{xx}(s, t) = \text{cov}[x_d(s), x_d(t)] \]

\[ = E[x_d(s) - m(s)][x_d(t) - m(t)]^T \tag{2.24} \]

\[ = \int \int [\xi_1 - m(s)][\xi_2 - m(t)]^T dF(\xi_1, \xi_2; s, t). \tag{2.25} \]

Analogously, the cross-covariance function of two stochastic processes is defined by:

\[ r_{xy}(s, t) = \text{cov}[x_d(s), y_d(t)]. \tag{2.27} \]

A stochastic process is called stationary if the finite-dimensional distribution of \( x_d(k_1), \ldots, x_d(k_n) \) is identical to the distribution of \( x_d(k_1 + \tau), \ldots, x_d(k_n + \tau) \), for all \( \tau \) such that \( k_i \in \mathbb{Z} \) and \( k_i + \tau \in \mathbb{Z}, i = 1, \ldots, n \). The process is called weakly stationary if the first two moments of the distributions are the same, see [3, Section 2.3]. With some abuse of notation, a weakly stationary stochastic process is also called a stationary stochastic process, despite the difference in definition.

From (2.23), it can be concluded that the mean value function of a (weakly) stationary process is constant. Furthermore, the cross-covariance function of weakly stationary processes is a function of the difference \( s - t \) only. With some abuse of notation, the function can be written as follows:

\[ r_{xy}(\tau) = \text{cov}[x_d(t + \tau), y_d(t)]. \tag{2.28} \]

When the processes \( x_d(t) \) and \( y_d(t) \) are the same, \( r_{xx}(\tau) \) is referred to as the auto-covariance function.

The cross-spectral density of (weakly) stationary processes is the Fourier transform of its covariance function. This is reflected by the following transform pair:

\[ \Phi_{xy}(\omega) = \sum_{k=-\infty}^{\infty} r_{xy}(k)e^{-j\omega k} \tag{2.29} \]

\[ r_{xy}(k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \Phi_{xy}(\omega)e^{j\omega k} d\omega, \tag{2.30} \]
where use is made of Definition 5. Usually, $\Phi_{xx}$ is referred to as the auto-spectral density and $\Phi_{xy}$ is referred to as the cross-spectral density.

Next, the interpretation of covariances and spectra is discussed, because it is useful to understand how the properties of a stochastic process are reflected by these functions.

The value $r_{xx}(0)$ of the covariance function at the origin is the variance $\lambda$ of the process. It reflects the size of the fluctuations. Normalization of the covariance function by $r_{xx}(0)$ leads to the so-called correlation function:

$$\rho_{xx}(\tau) = \frac{r_{xx}(\tau)}{r_{xx}(0)},$$

which has a magnitude less than or equal to one. Basically, the value $\rho_{xx}(\tau)$ gives the correlation between values of the process with a spacing $\tau$.

The auto-spectral density $\Phi_{xx}(\omega)$ represents the distribution of signal power over frequency. As a consequence, integration over a certain frequency band results in the signal power present in that specific frequency band. Furthermore, the total area under the spectral density curve is proportional to the total variance of the signal.

More information with respect to the concepts of stochastic processes can be found in [6, Section 6.5] and [3, Chapter 21.

Discrete Time White Noise

After discussion of the general theory of stochastic processes, one specific type of random process is now introduced. Consider a stationary discrete time stochastic process $x_d(t)$, such that $x_d(t)$ and $x_d(s)$ are independent if $t \neq s$. As a result, the stochastic process can be considered as a sequence $x_d(k, \zeta)$, $k \in \mathbb{Z}$ of independent, equally distributed random variables. The covariance function $r_{xx}(\tau)$ is given by:

$$r_{xx}(\tau) = \begin{cases} \lambda & \tau = 0 \\ 0 & \tau = \pm 1, \pm 2, \ldots \end{cases}$$

A stochastic process with this covariance function is called discrete time white noise. From (2.29), it follows that the auto-spectral density is given by $\Phi_{xx}(\omega) = \lambda$. Hence, it follows that the spectral density is constant for all frequencies.

The concept of white noise plays an important role in stochastic control theory. All stochastic processes that are needed can be generated by filtering white noise. This is shown in Section 3.3, where the modeling of disturbances is considered.

2.4 Signal Norms

It is often useful to have a measure of the size of a signal. For example, the performance of a tracking control system can be described by the size of the error signal. A measure of the signal’s size is provided by the signal norm. There are several different ways of defining norms for signals. Which norm is appropriate depends on the specific situation that is considered.

Consider a discrete time signal $x_d(k)$, defined for $k \in \mathbb{Z}$. This signal may be scalar-valued (with values in $\mathbb{R}$ or $\mathbb{C}$) or vector-valued (with values in $\mathbb{R}^n$ or $\mathbb{C}^n$). Consequently, the signals are elements of a so-called vector space. Given such a signal $x_d(k)$, its norm is a mathematically well-defined notion that is a measure for the size of the signal.
2.4 Signal Norms

Definition 8 (Norms). Let $X$ denote a vector space over the real $\mathbb{R}$ or complex $\mathbb{C}$ numbers. Then, a function $\| \cdot \| : X \to \mathbb{R}$, that maps $X$ into the set of real numbers $\mathbb{R}$, is a norm if it satisfies the following properties:

i. $\| x \| \geq 0$ for all $x \in X$; nonnegativity

ii. $\| x \| = 0$ if and only if $x = 0$; positive-definiteness

iii. $\| \gamma x \| = |\gamma| \| x \|$ for every scalar $\gamma$ and all $x \in X$; homogeneity with respect to scaling

iv. $\| x + y \| \leq \| x \| + \| y \|$ for all $x \in X$ and all $y \in X$; triangle inequality,

see [15, Section 2.1].

In Definition 8, the general notion of a norm is defined. Next, several frequently used norms will be discussed. First, the $p$-norm of vectors in $\mathbb{R}^n$ and $\mathbb{C}^n$ is considered.

Definition 9 (Norms of vectors in $\mathbb{R}^n$ and $\mathbb{C}^n$). Let $p$ be a real number, such that $1 \leq p \leq \infty$. Then, the $p$-norm of an $n$-dimensional real-valued or complex-valued vector $x = [x_1, x_2, \ldots, x_n]$ is defined as:

$$
\| x \|_p = \begin{cases} 
\left( \sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}} & \text{for } 1 \leq p < \infty, \\
\max_{1 \leq i \leq n} |x_i| & \text{for } p = \infty.
\end{cases}
$$

see [24, Section 2.4].

Important special cases are, see also [19, Section 2.3.1]:

- 1-Norm:
  $$
  \| x \|_1 = \sum_{i=1}^{n} |x_i|.
  $$

- 2-Norm:
  $$
  \| x \|_2 = \left( \sum_{i=1}^{n} |x_i|^2 \right)^{\frac{1}{2}}.
  $$

The 2-norm is often called the Euclidean norm, because it corresponds to the usual notion of distance in Euclidean space. In $\mathbb{R}^2$ and $\mathbb{R}^3$, for instance, $\| x \|_2$ is actually the length of the vector with coordinates $x$.

- $\infty$-Norm:
  $$
  \| x \|_\infty = \max_{1 \leq i \leq n} |x_i|.
  $$

The $\infty$-norm can be viewed as a limiting case as $p \to \infty$.

The $p$-norm in Definition 9 for constant vectors may easily be generalized to vector-valued signals.
Definition 10 (\(\ell_p\)-norm of a signal). For any \(1 \leq p \leq \infty\), the \(p\)-norm or \(\ell_p\)-norm \(\|x_d\|_{\ell_p}\) of a discrete time scalar-valued signal \(x_d(k)\) is defined by:

\[
\|x_d\|_{\ell_p} = \left\{ \begin{array}{ll}
\left(\sum_{k=-\infty}^{\infty} |x_d(k)|^p\right)^{\frac{1}{p}} & \text{for } 1 \leq p < \infty, \\
\sup_{k \in \mathbb{Z}} |x_d(k)| & \text{for } p = \infty
\end{array} \right.
\]  

(2.37)

Here, "sup" denotes the supremum or least upper bound. That is, \(\sup_{k \in \mathbb{Z}} |x_d(k)|\) is the smallest real number \(\alpha\) such that \(|x_d(k)| \leq \alpha\) for all \(k \in \mathbb{Z}\).

If \(x_d(k)\) is vector-valued, with values in \(\mathbb{R}^n\) or \(\mathbb{C}^n\), this definition is generalized to:

\[
\|x_d\|_{\ell_p} = \left\{ \begin{array}{ll}
\left(\sum_{k=-\infty}^{\infty} \|x_d(k)\|^p\right)^{\frac{1}{p}} & \text{for } 1 \leq p < \infty, \\
\sup_{k \in \mathbb{Z}} \|x_d(k)\| & \text{for } p = \infty
\end{array} \right.
\]  

(2.38)

where \(\|x_d(k)\|\) is any norm on the \(n\)-dimensional space \(\mathbb{R}^n\) or \(\mathbb{C}^n\). See also [24, Section 2.4] and [9, Section 4.3.1].

Again, several frequently used special cases are mentioned, see also [24, Section 2.4]:

- **\(\ell_1\)-Norm:**
  \[
  \|x_d\|_{\ell_1} = \sum_{k=-\infty}^{\infty} \|x_d(k)\|. 
  \]  
  (2.39)

  The \(\ell_1\)-norm is also called the action of the signal.

- **\(\ell_2\)-Norm:**
  \[
  \|x_d\|_{\ell_2} = \left(\sum_{k=-\infty}^{\infty} \|x_d(k)\|^2\right)^{\frac{1}{2}}. 
  \]  
  (2.40)

  The square of the \(\ell_2\)-norm, denoted by \(\|x_d\|_{\ell_2}^2\), is often called the energy of the signal \(x_d(k)\).

- **\(\ell_\infty\)-Norm:**
  \[
  \|x_d\|_{\ell_\infty} = \sup_{k \in \mathbb{Z}} \|x_d(k)\|. 
  \]  
  (2.41)

  The \(\ell_\infty\)-norm is called the amplitude of the signal \(x_d(k)\), because it is the largest magnitude the signal assumes.

Several additional quantities that are used to represent certain aspects of the signal, such as power, RMS value and mean of a signal, can be found in [24, Section 2.4].
Chapter 3

Linear Time-Invariant Systems

3.1 Introduction

Linear time-invariant systems form the most important class of dynamical systems considered in theory and in practice. Due to this importance, first some basic concepts will be illustrated. In general, the discussion will be limited to single input single output (SISO) systems.

3.2 Impulse Responses

Consider a system with a scalar input signal $u(t)$ and a scalar output signal $y(t)$, such as depicted in Figure 3.1. The system is said to be time-invariant if its response to a certain input signal does not depend on absolute time. It is said to be linear if its output response to a linear combination of inputs is the same linear combination of the output responses of the individual inputs. Furthermore, it is said to be causal if the output at a certain time depends on the input up to that time only.

Figure 3.1: System definition.

It is well-known, see e.g. [37, Section 2.2], that a linear time-invariant, causal system, operating on continuous time signals, can be described by its impulse response $g(\tau)$ as follows:

$$y(t) = \int_{\tau=0}^{\infty} g(\tau) u(t - \tau) d\tau.$$  \hspace{1cm} (3.1)

Knowing $\{g(\tau)\}_{\tau=0}^{\infty}$ and knowing $u(s)$ for $s \leq t$, it is possible to compute the corresponding output $y(s)$, $s \leq t$ for any input. The impulse response is thus a complete characterization of the system.

In this survey, observations of inputs and outputs are almost exclusively set in discrete time, since this is the typical data-acquisition mode used in practice. Therefore, it is assumed that $y(t)$ is observed at the sampling instants $t(k) = kT$, for $k = 1, 2, \ldots$:

$$y(kT) = \int_{\tau=0}^{\infty} g(\tau) u(kT - \tau) d\tau.$$  \hspace{1cm} (3.2)
The interval $T$ will be called the sampling interval. Most often, in computer control applications, the input signal $u(t)$ is kept constant between the sampling instants (zero-order hold) as a result of digital-to-analog (D/A) conversion:\footnote{If this assumption is not made, the resulting discrete time system will in general not be causal and will generally not have a finite-dimensional representation, see [37, Section 2.6.1].}

$$u(t) = u(kT), \quad kT \leq t < (k + 1)T.$$ (3.3)

Inserting (3.3) into (3.2) and performing some simplifications leads to the following result:

$$y(kT) = \sum_{\ell=1}^{\infty} \left[ \int_{\tau=(\ell-1)T}^{\ell T} g(\tau) d\tau \right] u((k - \ell)T) = \sum_{\ell=1}^{\infty} g_T(\ell)u((k - \ell)T).$$ (3.4)

The relationship (3.4) describes a sampled-data system and the sequence $\{g_T(\ell)\}_{\ell=1}^{\infty}$ will be called the impulse response of that system. Recall that the sampled-data system originates from sampling continuous time signals, leading to the use of the sampling interval $T$. It is, however, also possible to discard the connection with a sampling mechanism, without losing any information, see [37, Section 2.4.3]. This is achieved by inserting $T = 1$ into (3.4) and using $t$ to enumerate the sampling instants, leading to the following discrete time system:

$$y(t) = \sum_{k=1}^{\infty} g(k)u(t - k), \quad t = 0, 1, 2, \ldots.$$ (3.5)

In the sequel, this notation will be employed.

### 3.3 Disturbances

According to (3.5), the output can be exactly calculated once the input is known. In most cases this is unrealistic. There are always signals beyond our control that also affect the system. Within the linear framework, it is assumed that such effects can be lumped into an additive term $v(t)$ at the output, see Figure 3.2:

$$y(t) = \sum_{k=1}^{\infty} g(k)u(t - k) + v(t).$$ (3.6)

![Figure 3.2: System with additive disturbance located at the output.](image)
The most characteristic feature of a disturbance is that its value is not known beforehand. This raises the question how to model the disturbance signal $v(t)$. It should be able to reflect all disturbance terms that one expects to be present in a given situation. In this case, the disturbance $v(t)$ is modeled as a zero mean stationary stochastic process:

$$v(t) = \sum_{k=0}^{\infty} h(k)e(t-k),$$

(3.7)

where \{e(t)\} is white noise, i.e., a sequence of independent, identically distributed, random variables with a certain probability density function (PDF). Identically distributed means that the PDF of $e(t)$ is independent of $t$ and independent means that the joint PDF of $e(t_1)$ and $e(t_2)$ is the product of the two separate PDF's. Although this description does not allow completely general characterizations of all possible probabilistic disturbances, it is versatile enough for most practical purposes. In Section 4.2, it is shown how the description (3.7) allows predictions about future disturbances.

It should be made clear that the specification of different probability density functions for \{e(t)\} may result in very different characteristic features of the disturbance. However, instead of specifying the PDF for \{e(t)\}, attention is most often limited to the specification of the so-called second-order properties of the sequence \{e(t)\}, i.e., the mean and the variance. In the sequel, $e(t)$ is assumed to have zero mean and variance $\lambda$. Using the description of $v(t)$ according to (3.7), its mean is computed as:

$$Ev(t) = \sum_{k=0}^{\infty} h(k)Ev(t-k) = 0$$

(3.8)

and its covariance function $R_v(\tau)$ as:

$$R_v(\tau) = Ev(t)v(t-\tau) = \lambda \sum_{k=0}^{\infty} h(k)h(k-\tau).$$

(3.9)

The second-order properties of $v(t)$ are now specified by (3.8) and (3.9), which are uniquely defined by the sequence \{h(k)\} and the variance $\lambda$ of $e(t)$. Since both (3.8) and (3.9) do not depend on $t$, the process is said to be stationary.

### 3.4 Transfer Functions

In order to allow a more convenient notation for sums like (3.6) and (3.7), two operators are introduced:

$$qu(t) = u(t+1)$$

(3.10)

$$q^{-1}u(t) = u(t-1),$$

(3.11)

where $q$ is referred to as the forward shift operator and $q^{-1}$ as the backward shift operator. Using both operators, it is possible to rewrite (3.5) as:

$$y(t) = \left[ \sum_{k=1}^{\infty} g(k)q^{-k} \right] u(t) = G(q)u(t).$$

(3.12)

\footnote{To guarantee a unique representation, it is assumed that $h(0) = 1$, which is no loss of generality since the variance of $e(t)$ can be adjusted.}
With slight abuse of notation, the operator $G(q)$ is referred to as the transfer function of the linear system (3.5). Strictly speaking, however, the transfer function is defined by the function $G(z)$:

$$G(z) = \sum_{k=1}^{\infty} g(k)z^{-k},$$

where $z$ is a complex indeterminate. Subsequently, (3.7) is transformed in a similar way:

$$v(t) = \left[ \sum_{k=0}^{\infty} h(k)q^{-k} \right] e(t) = H(q)e(t).$$

This leads to the following basic description for a linear system with additive disturbance:

$$y(t) = G(q)u(t) + H(q)e(t),$$

where $\{e(t)\}$ is a sequence of independent random variables with zero mean values and variances $\lambda$.

**Remark.** The transfer function $G(q)$ is said to be stable if the following expression holds:

$$\sum_{k=1}^{\infty} |g(k)| < \infty.$$  

Definition (3.16) coincides with the system theoretic definition of bounded input bounded output (BIBO) stability, see [18, Section 16.1]. This definition states that if an input signal with bounded amplitude is applied, then an output signal with bounded amplitude results.

**Remark.** A filter $H(q)$ is called monic if its zeroth coefficient is one:

$$H(q) = \sum_{k=0}^{\infty} h(k)q^{-k}, \ h(0) = 1.$$  

In order to guarantee a unique representation of (3.15), $H(q)$ is restricted to be monic. This is necessary since the signal $e(t)$ cannot be measured and, thus, its variance cannot be uniquely determined. As a result, the level of noise $v(t)$ is allowed to depend on both the variance of $e(t)$ and the gain of $H(q)$. By restricting $H(q)$ to be monic, this freedom is removed, leading to a unique representation.

**Remark.** In preparation of Chapter 4, it is imposed that the noise model (3.14) is invertible. This is necessary in order to predict the value of $v(t)$ based on past observations. Invertibility of the noise model implies that if $v(s), s \leq t$ are known, it is possible to compute $e(t)$ according to:

$$e(t) = \tilde{H}(q)v(t) = \sum_{k=0}^{\infty} \tilde{h}(k)v(t-k),$$

with $\tilde{H}(q)$ a stable filter, defined by $\tilde{H}(q) = H^{-1}(q)$. The only necessary requirement for this to hold is that the function $\frac{1}{\tilde{H}(z)}$ be analytic in $|z| \geq 1$, i.e., it has no poles on or outside the unit circle. For a proof, the reader is referred to [26, Section 3.2].
3.5 Frequency Function

Suppose that the system (3.5) is excited with a sinusoidal input signal:

\[ u(t) = \cos \omega t = \text{Re}\{e^{j\omega t}\}. \]  

(3.19)

According to (3.5), the output of the discrete time system is given by:

\[ y(t) = \sum_{k=1}^{\infty} g(k)u(t-k) \]
\[ = \text{Re}\{e^{j\omega t}G(e^{j\omega})\} \]
\[ = |G(e^{j\omega})|\cos(\omega t + \varphi), \]  

(3.20)

(3.21)

(3.22)

where \( \varphi = \arg G(e^{j\omega}) \). From (3.22), it follows that the output to (3.19) will be a cosine of the same frequency, but with an amplitude magnified by \( |G(e^{j\omega})| \) and a phase shift of \( \arg G(e^{j\omega}) \) radians. The complex-valued function \( G(e^{j\omega}) \) is referred to as the frequency function of the discrete time system (3.5). It evaluates the transfer function in the complex plane over the unit circle \( z = e^{j\omega} \) and gives full information as to what will happen in stationarity, when the input signal is a sinusoid of frequency \( \omega \). A graphical representation of the frequency function is given by a Bode plot and a Nyquist plot.

3.6 Quasi-Stationarity

In practical engineering situations of analyzing signals, it is often inconvenient to describe measurement signals as being either deterministic or stochastic. A more convenient approach is to consider the input sequence as being deterministic, or at least partly deterministic, while disturbances on the system are described by random variables. In this way, the system output becomes a stochastic process with deterministic components. In the case of Machine-In-the-Loop control optimization, it is expected that the relevant signals, typically the tracking error, can be described as a stochastic process with deterministic components as well.

However, if \( y(t) \) in (3.15) is considered to be a stochastic process, with \( u(t) \) deterministic and \( e(t) \) stochastic, this process will generally be nonstationary. This can be understood by realizing that:

\[ Ey(t) = G(q)u(t), \]  

(3.23)

which is not a stationary process due to the dependence on \( t \).

To deal with these types of signals in a way that facilitates its analysis, a generalized expectation operation is introduced, denoted by \( \overline{E} \) and defined by\(^3\):

\[ \overline{E}f(t) = \lim_{N \to \infty} \frac{1}{N} \sum_{t=1}^{N} Ef(t). \]  

(3.24)

This generalized expectation operation combines a statistical expectation with a time-averaging. Two special cases can be distinguished. First, if \( f(t) \) is a deterministic sequence then:

\[ \overline{E}f(t) = \lim_{N \to \infty} \frac{1}{N} \sum_{t=1}^{N} f(t). \]  

(3.25)

\(^3\)Note that the expression \( \overline{E}f(t) \) is not a function of \( t \) anymore, since an averaging over \( t \) is involved.
Second, if \( f(t) \) is a stationary stochastic process then:

\[
\overline{E}f(t) = Ef(t).
\]

Using this generalized expectation operator, the following notion of quasi-stationarity is defined.

**Definition 11 (Quasi-stationary signal).** A signal \( \{s(t)\} \) is called quasi-stationary if there exist \( c_1, c_2 \in \mathbb{R} \) such that:

1. \(|Es(t)| \leq c_1 \) for all \( t \)
2. \( R_s(\tau) = \overline{Es}(t)s(t - \tau) \) satisfies \(|R_s(\tau)| \leq c_2 \) for all \( \tau \).

Here, expectation \( E \) is taken over the stochastic components in the signal \( s(t) \), while the time-averaging is taken over the deterministic components. For a deterministic sequence \( \{s(t)\} \) to be quasi-stationary, it has to be bounded in the sense that the limit:

\[
\lim_{N \to \infty} \frac{1}{N} \sum_{t=1}^{N} s(t)s(t - \tau)
\]

is finite for all \( \tau \). If \( \{s(t)\} \) is a stationary stochastic process, the conditions as mentioned in the definition are automatically satisfied, since then \( Es(t)s(t - \tau) = R_s(\tau) \) does not depend on \( t \).

Similar to the situation of stationary stochastic processes, two signals \( \{s(t)\} \) and \( \{w(t)\} \) are called jointly quasi-stationary if they both are quasi-stationary and if \( R_{sw}(\tau) = \overline{Es}(t)w(t - \tau) \) exists. Furthermore, jointly quasi-stationary signals \( \{s(t)\} \) and \( \{w(t)\} \) are uncorrelated if \( R_{sw}(\tau) \) is identically zero.

With slight abuse of notation, the following terms will be used:

- **Correlation function or covariance function:**
  \[
  R_s(\tau) = \overline{Es}(t)s(t - \tau)
  \]

- **Cross-correlation function or cross-covariance function:**
  \[
  R_{sw}(\tau) = \overline{Es}(t)w(t - \tau)
  \]

- **Power spectral density or power spectrum:**
  \[
  \Phi_s(\omega) = \sum_{\tau=-\infty}^{\infty} R_s(\tau)e^{-i\omega \tau}
  \]

- **Power cross-spectral density or cross spectrum:**
  \[
  \Phi_{sw}(\omega) = \sum_{\tau=-\infty}^{\infty} R_{sw}(\tau)e^{-i\omega \tau}.
  \]
Using these respective terms, one has to realize that they are formally correct only in the case of stationary stochastic processes.

The generalized expectation operator also induces a generalized notion of power of a signal, which is expressed as \( \overline{E}s^2(t) \). In the case of a stationary stochastic process, this equals the ensemble-average power of the process, while for a deterministic sequence it reflects the time-average power of the signal. It follows directly from the relations above and the inverse DFT that:

\[
\overline{E}s^2(t) = R_s(0) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \Phi_s(\omega) d\omega,
\]

which implies that the power spectral density \( \Phi_s(\omega) \) reflects the distribution of the power of \( s(t) \) over frequency.

Next, the concept of quasi-stationarity is further explored in two examples.

**Example 2 (Stationary stochastic process).** Let \( \{v(t)\} \) be a stationary stochastic process with covariance function (3.9). As a result, the second condition in Definition 11 equals (3.9) and the definition of power spectrum in (3.30) coincides with the conventional one. Given the process \( v(t) \) in (3.7) and its covariance function in (3.9), the corresponding power spectrum thus equals:

\[
\Phi_v(\omega) = \lambda |H(e^{i\omega})|^2,
\]

see [26, Section 2.3]. It can be concluded that the stochastic process described by \( v(t) = H(q)e(t) \), where \( \{e(t)\} \) is a sequence of independent random variables with zero mean values and variances \( \lambda \), has the power spectrum given in (3.33).

**Example 3 (Combination of deterministic and stochastic signals).** Consider the signal \( s(t) = u(t) + v(t) \), where \( \{u(t)\} \) is a deterministic signal with spectrum \( \Phi_u(\omega) \) and \( \{v(t)\} \) is a stationary stochastic process with zero mean value and spectrum \( \Phi_v(\omega) \). Then it holds that:

\[
\overline{E}u(t)v(t - \tau) = \lim_{N \to \infty} \frac{1}{N} \sum_{t=1}^{N} u(t)Ev(t - \tau) = 0,
\]

because \( v(t) \) is zero mean. As a direct result, it follows that:

\[
\Phi_s(\omega) = \Phi_u(\omega) + \Phi_v(\omega).
\]

### 3.7 Transformation of Signal Spectra

As signals are filtered through linear systems, their properties will change. In (3.33), for instance, it is shown how a stationary stochastic process is created by white noise. For spectra, the following general result can be stated.

**Theorem 1.** Let \( \{w(t)\} \) be a quasi-stationary signal with spectrum \( \Phi_w(\omega) \) and let \( G(q) \) be a stable transfer function. Furthermore, let \( s(t) = G(q)w(t) \). Then \( \{s(t)\} \) is also quasi-stationary and the following relations hold:

\[
\Phi_s(\omega) = |G(e^{i\omega})|^2 \Phi_w(\omega),
\]

\[
\Phi_{sw}(\omega) = G(e^{i\omega})\Phi_w(\omega).
\]
The proof can be found in [26, Appendix 2A].

**Corollary 1.** Let \( \{y(t)\} \) be given by 
\[ y(t) = G(q)u(t) + H(q)e(t), \]
where \( \{u(t)\} \) is a quasi-stationary, deterministic signal with spectrum \( \Phi_u(\omega) \) and \( \{e(t)\} \) is white noise with variance \( \lambda \). Let \( G \) and \( H \) be stable filters. Then \( \{y(t)\} \) is quasi-stationary and the following relations hold:

\[
\begin{align*}
\Phi_y(\omega) &= |G(e^{j\omega})|^2 \Phi_u(\omega) + \lambda |H(e^{j\omega})|^2, \\
\Phi_{yu}(\omega) &= G(e^{j\omega}) \Phi_u(\omega).
\end{align*}
\]

(3.38) (3.39)

The corollary follows from Theorem 1, using Examples 2 and 3.

### 3.8 System Norms

This section is concerned with the notion of system norms. System norms are often referred to as operator norms, because a system represented by a transfer function is an operator, which maps functions (input signals) into other functions (output signals). These norms measure the amplification, or at least the greatest possible amplification, of this mapping. In other words, a system norm is a measure of the "gain" of a transfer function, expressed by a single number.

In this section, two different norms for a stable transfer function \( M \) are derived. These are the \( H_\infty \) norm and the \( H_2 \) norm. Furthermore, attention is restricted to linear time-invariant single input single output finite-dimensional systems. As a result, the transfer function is rational with real coefficients. Although discrete time transfer functions are of main interest, continuous time transfer functions are considered first, because they appear more often in relation to system norms.

#### 3.8.1 Continuous Time Case

In the continuous time case, the transfer function is denoted by \( M(s) \). It is assumed that the considered transfer functions are strictly proper (degree of denominator > degree of numerator) and stable, leading to a finite 2-norm and a finite \( \infty \)-norm. Next, these norms are introduced for the transfer function \( M(s) \).

- **\( \infty \)-Norm:**
  
  One possibility to measure the gain of the considered system is given by the \( \infty \)-norm:
  
  \[
  \|M\|_\infty = \sup_{\omega} |M(j\omega)|.
  \]
  
  (3.40)

  This norm is finite, since \( M(s) \) is proper and has no poles on the imaginary axis, see [15, Section 2.2]. The \( \infty \)-norm of \( M(s) \) equals the distance in the complex plane from the origin to the farthest point on the Nyquist plot of \( M(s) \). It also appears as the peak value on the Bode magnitude plot of \( M(s) \). An important property of the \( \infty \)-norm is that it is submultiplicative:

  \[
  \|MN\|_\infty \leq \|M\|_\infty \|N\|_\infty.
  \]
  
  (3.41)
3.8 System Norms

• 2-Norm:
Another possibility to measure the gain of the considered system is given by the 2-norm. The 2-norm of a transfer function is the expected root mean square (RMS) value of the output when the input is a realization of unit variance white noise. It is defined as follows:

\[ \| M \|_2 = \int_0^\infty |M(t)|^2 dt \]
\[ = \frac{1}{2\pi} \int_{-\infty}^{\infty} |M(j\omega)|^2 d\omega, \]  
(3.42)
(3.43)

where the second equality holds by Parseval's theorem. The first equality implies that the 2-norm of a system can be interpreted as the \( L_2 \)-norm, see [9, Section 4.3.1], of its response to a unit impulse, see [11, Section 1]. This norm is finite as well, since \( M(s) \) is strictly proper and has no poles on the imaginary axis, see [15, Section 2.2]. Notice that the 2-norm is not submultiplicative. It is possible to rewrite the 2-norm as follows:

\[ \| M \|_2 = \frac{1}{2\pi j} \int_{-j\infty}^{j\infty} M^*(s)M(s)ds \]  
\[ = \frac{1}{2\pi j} \oint M^*(s)M(s)ds, \]  
(3.44)
(3.45)

where \( M^*(s) = \overline{M(-s)} \). Here, \( \overline{M(s)} \) denotes the complex conjugate of \( M(s) \). This notation is convenient with respect to the comparison of the 2-norm in the continuous time and the discrete time case.

Ways to compute the \( \infty \)-norm and the 2-norm can be found in [15, Section 2.2].

3.8.2 Discrete Time Case
In the discrete time case, the transfer function is denoted by \( M(z) \). It is assumed that \( M(z) \) is proper and has no poles on the unit circle. Again, definitions of the \( \infty \)-norm and the 2-norm are given.

• \( \infty \)-Norm:
The \( \infty \)-norm of the discrete time transfer function \( M(z) \) is given by:

\[ \| M \|_\infty = \sup_{\omega} |M(e^{j\omega})|. \]  
(3.46)

The remarks that are made about the \( \infty \)-norm in the continuous time case remain valid in the discrete time case.

• 2-Norm:
The definition of the 2-norm of \( M(z) \) also is very similar to the definition in the continuous time case, given by (3.45). The 2-norm of \( M(z) \) is defined according to:

\[ \| M \|_2 = \frac{1}{2\pi} \int_0^{2\pi} M^*(e^{j\omega})M(e^{j\omega})d\omega \]
\[ = \frac{1}{2\pi j} \oint M^*(z)M(z)z^{-1}dz, \]  
(3.47)
(3.48)
see [11, Section 1]. It can be concluded that the 2-norm of $M(z)$ is finite. Furthermore, the 2-norm remains finite even when a direct feedthrough term is present in the transfer function. This does not hold in the continuous time case.

In the discrete time case, it can easily be shown that the 2-norm of a transfer function is bounded from above by its $\infty$-norm:

$$
\|M\|_2^2 = \frac{1}{2\pi} \int_0^{2\pi} |M(e^{j\omega})|^2 d\omega \leq \frac{1}{2\pi} \int_0^{2\pi} \|M\|_\infty^2 d\omega = \|M\|_\infty^2.
$$

(3.49)

More information about norms for discrete time transfer functions can be found in [11].
Chapter 4

Prediction

4.1 Introduction

In the Machine-In-the-Loop control optimization concept, the modeling of systems may play an important role. In this context, one can think of model-based feedforward control, where the feedforward model should approximate the inverse of the process sufficiently well in order to achieve a good performance. Consequently, it is desired to evaluate to what extent the feedforward model approximates the process inverse.

The capability of a model to predict future values of signals is an important property with respect to the evaluation of models as candidate representatives of systems to be modeled. Therefore, in this chapter an analysis is performed how general models can be used to predict future output values. Furthermore, the idea of how to predict future output values will turn out to be most essential for the development of identification methods. Throughout the chapter, it is assumed that the system description is given in the form (3.15).

4.2 One-Step-Ahead Prediction of $v$

Consider the description of $v(t)$ according to:

$$v(t) = H(q)e(t),$$

which is the same as in (3.14). The one-step-ahead prediction problem then can be formulated as to predict the value of $v(t)$ based on the observations of $v(s)$ for $s \leq t - 1$. Since $H(q)$ is assumed to be monic, $v(t)$ can be rewritten as:

$$v(t) = \sum_{k=0}^{\infty} h(k)e(t-k) = e(t) + \sum_{k=1}^{\infty} h(k)e(t-k).$$

Because $H(q)$ is also assumed to be an inversely stable filter, see (3.18), the knowledge of $v(s)$, $s \leq t - 1$ implies the knowledge of $e(s)$, $s \leq t - 1$. Therefore, the second term of (4.2) is known at time $t - 1$. However, the first term of (4.2), $e(t)$, is unknown at time $t - 1$. In spite of this, several possibilities can be chosen to obtain a prediction of $v(t)$.

Suppose that the probability density function of $e(t)$ is given by $f_e(x)$, according to:

$$P(x \leq e(t) \leq x + \Delta x) \approx f_e(x)\Delta x.$$
This distribution is independent of the other values of \( e(s), s \neq t \), since \( \{e(t)\} \) is a sequence of independent random variables. Using the distribution \( f_e(x) \), it can be stated that the posterior probability density function of \( v(t) \), given \( u \) and \( y \) up to time \( t - 1 \), is represented by:

\[
f_v(x) = f_e \left( x - \sum_{k=1}^{\infty} h(k)e(t-k) \right).\tag{4.4}
\]

This is the most complete statement that can be made about \( v(t) \) at time \( t - 1 \).

One possible choice is now to pick that value of \( v(t) \) for which \( f_v(x) \) has its maximum value. This is the most probable value of \( v(t) \), which also is called the maximum a posteriori (MAP) prediction. Another predictor, that will be used throughout the chapter, is the conditional expectation of \( v(t) \), denoted by \( \hat{v}(t|t-1) \). The conditional expectation, i.e., the mean value of the distribution in question, is defined by:

\[
\hat{v}(t|t-1) = Ev(t) = \sum_{k=1}^{\infty} h(k)e(t-k) = [1 - H^{-1}(q)]v(t).\tag{4.5}
\]

\[
\hat{v}(t|t-1) = Ev(t) = \sum_{k=1}^{\infty} h(k)e(t-k) = [1 - H^{-1}(q)]v(t).\tag{4.6}
\]

### 4.3 One-Step-Ahead Prediction of \( y \)

Consider the following situation:

\[
y(t) = G(q)u(t) + v(t),\tag{4.7}
\]

with \( v(t) = H(q)e(t) \) as before. Analogously to Section 4.2, the one-step-ahead prediction problem can now be formulated as to predict \( y(t) \), given observations \( y(s) \) and \( u(s) \) for \( s \leq t - 1 \). According to (4.7), \( v(s) \) are also known for \( s \leq t - 1 \). Using this information, the conditional expectation of \( y(t) \) is as follows:

\[
\hat{y}(t|t-1) = G(q)u(t) + \hat{v}(t|t-1).\tag{4.8}
\]

Note that the terms on the right hand side contain signals that are available at time \( t - 1 \), provided that \( G(q) \) is strictly proper, i.e., it does not have a constant (direct feedthrough) term. Inserting (4.6) into (4.8) and performing some simplifications leads to the following predictor:

\[
\hat{y}(t|t-1) = H^{-1}(q)G(q)u(t) + [1 - H^{-1}(q)]y(t).\tag{4.9}
\]

An additional property of \( \hat{y}(t|t-1) \) is that it is the best one-step-ahead prediction of \( y(t) \) if a quadratic error criterion is considered.

The one-step-ahead predictor is derived for the situation in which the mechanism that generates the data \( y(t) \) is described by a known model, given by \( G(q) \) and \( H(q) \). Obviously, if \( y(t) \) satisfies this condition, it holds that:

\[
y(t) - \hat{y}(t|t-1) = e(t).\tag{4.10}
\]

For general \( y(t) \), however, one can denote the one-step-ahead prediction error as:

\[
e(t) = y(t) - \hat{y}(t|t-1).\tag{4.11}
\]
This prediction error can only be calculated a posteriori, when the measurement \( y(t) \) has become available. Substituting (4.9) for the one-step-ahead prediction, the following expression for the prediction error is obtained:

\[
\varepsilon(t) = H^{-1}(q)[y(t) - G(q)u(t)].
\]  

This prediction error is exactly that component of \( y(t) \) that cannot be predicted from past data. For this reason, it is also called the innovation at time \( t \).

The one-step-ahead prediction of \( y(t) \) can be calculated when the whole data record from time minus infinity to \( t - 1 \) is available. Indeed, all these data appear explicitly in (4.9). In practice, however, it is usually the case that only data over the interval \([0, t - 1]\) are known. Then, (4.9) still can provide a prediction of \( y(t) \), by assuming zero initial conditions. It should be realized that this is only an approximation of the actual conditional expectation of \( y(t) \). The exact prediction, which holds in case of nonzero initial conditions, involves time-varying filter coefficients and can be computed using the Kalman filter, see [17]. Nevertheless, the predictor (4.9) will be used in the sequel, thereby implicitly assuming that the initial conditions are zero.

It should be remarked that in many cases one does not work with a full description of the properties of disturbances as in (4.1). Instead, a noise-free or deterministic model is used:

\[
y(t) = G(q)u(t).
\]  

The assumption (4.13) can also be used for prediction purposes. As a result of the fact that no noise model is present, several possibilities are available to do this. The concept of observers is a key issue for these calculations. A discussion of this concept can be found in [26, Section 3.3].

### 4.4 \( k \)-Step-Ahead Prediction of \( y \)

With a similar line of reasoning, an expression can be derived for the \( k \)-step-ahead \((k > 1)\) prediction of \( y(t) \). The predictor is given by:

\[
\hat{y}(t|t-k) = W_k(q)G(q)u(t) + [1 - W_k(q)]y(t),
\]

with the following definitions:

\[
W_k(q) = \tilde{H}_k(q)H^{-1}(q)
\]

\[
\tilde{H}_k(q) = \sum_{\ell=0}^{k-1} h(\ell)q^{-\ell}.
\]

For more details on this, the reader is referred to [26, Section 3.2].
Chapter 5

Models of Linear Time-Invariant Systems

5.1 Introduction

A model of a system is a partial description of its properties, suitable for certain purposes. In Machine-In-the-Loop control optimization, models of systems may be used in several different control design problems, for example in model-based feedforward control. As a result, it is necessary to be able to construct or select a model of a dynamical system in such a way that it serves its purpose. The construction or selection of models of dynamical systems is the subject of system identification. A first step in the procedure is to determine a class of models within which the search for the most suitable model is to be conducted. Therefore, it is necessary to have a certain knowledge with respect to classes of models. In this chapter, several classes of models for linear time-invariant systems will be discussed. Attention is restricted to transfer function models, because they represent the most intuitive way of parameterization. Furthermore, state-space models are especially useful in case of first principles modeling, which will not be pursued in Machine-In-the-Loop control optimization. For more details on state-space models, the reader is referred to [26, Section 4.3].

5.2 Linear Models and Sets of Linear Models

A complete description of a linear time-invariant model is given by:

\[ y(t) = G(q)u(t) + H(q)e(t) \]

\[ = \left[ \sum_{k=1}^{\infty} g(k)q^{-k} \right] u(t) + \left[ 1 + \sum_{k=1}^{\infty} h(k)q^{-k} \right] e(t). \]
finite number of numerical values has an important consequence. Often, these coefficients are not known a priori, implying that they have to be estimated. This means that the coefficients in question enter the model (5.1) as parameters to be determined. These parameters are denoted by the vector \( \theta \), leading to the following model description:

\[
y(t) = G(q, \theta)u(t) + H(q, \theta)e(t),
\]

(5.3)

with \( f_e(x, \theta) \) representing the PDF of \( e(t) \) and \( \{e(t)\} \) being white noise. The parameter vector \( \theta \) then ranges over a subset of \( \mathbb{R}^d \), where \( d \) is the dimension of \( \theta \) according to \( \theta \in D_M \subset \mathbb{R}^d \). Notice that (5.3) no longer is a single model. It is a set of models and the member in the set that appears to be most suitable for the purpose in question is to be determined by an estimation procedure. Using (4.9), the one-step-ahead prediction for (5.3) can be computed. It is denoted by \( \hat{y}(t|\theta) \) to emphasize its dependence on \( \theta \):

\[
\hat{y}(t|\theta) = H^{-1}(q, \theta)G(q, \theta)u(t) + [1 - H^{-1}(q, \theta)]y(t).
\]

(5.4)

Since this predictor is not dependent on \( f_e(x, \theta) \), the prediction properties of the model are only determined by \( G \) and \( H \). Models that are specified by only these two rational functions are called predictor models. Furthermore, a parameterized set of models will be called a model structure and will be denoted by \( M \). The particular model associated with the parameter value \( \theta \) will be denoted by \( M(\theta) \). In the following section, different ways of describing (5.3) in terms of \( \theta \), i.e., different ways of parameterizing the model set, will be discussed.

### 5.3 Transfer Function Models

A direct way of parameterizing \( G \) and \( H \) is to represent them as rational functions and let the parameters be the numerator and denominator coefficients. Various ways of carrying out such parameterizations are described in this section. Notice that such model structures are also known as black-box models.

#### 5.3.1 Equation Error Model Structure

The most simple input-output relationship is obtained by describing it as a linear difference equation:

\[
y(t) + a_1 y(t-1) + \cdots + a_{na} y(t-na) = b_1 u(t-1) + \cdots + b_{nb} u(t-nb) + e(t).
\]

(5.5)

This model is often called an equation error model structure, since the white noise term \( e(t) \) enters as a direct error in the difference equation. The adjustable parameters are in this case given by the vector:

\[
\theta = [a_1 \ldots a_{na} \quad b_1 \ldots b_{nb}]^T.
\]

(5.6)

Introduction of \( A(q) \) and \( B(q) \) according to:

\[
A(q) = 1 + a_1 q^{-1} + \cdots + a_{na} q^{-na}
\]

(5.7)

\[
B(q) = b_1 q^{-1} + \cdots + b_{nb} q^{-nb},
\]

(5.8)

makes it possible to rewrite (5.5) as follows:

\[
A(q) y(t) = B(q) u(t) + e(t).
\]

(5.9)
Comparison of (5.9) and (5.3) reveals that the following equalities hold:

\[
G(q, \theta) = \frac{B(q)}{A(q)} \\
H(q, \theta) = \frac{1}{A(q)}.
\]  

The model (5.5) is also called an ARX model, where AR refers to the autoregressive part \( A(q) y(t) \) and X to the extra input \( B(q)u(t) \). In the special case where \( n_a = 0 \), it holds that \( A(q) = 1 \) and \( y(t) \) is modeled as a finite impulse response (FIR).

A block diagram of the ARX model is depicted in Figure 5.1. From this block diagram, it can be concluded that the model (5.5) is not the most natural one from a physical point of view: the white noise is assumed to go through the denominator dynamics of the system before being added to the output. Nevertheless, the equation error model set has a very important property that makes it a prime choice in many applications: the predictor defines a linear regression.

\[
\begin{align*}
\text{e} & \xrightarrow{1/A} y(t) \\
\text{u} & \xrightarrow{B/A} y(t)
\end{align*}
\]

Figure 5.1: ARX model structure.

### 5.3.2 Linear Regressions

The predictor for the ARX model structure is obtained by inserting (5.10) and (5.11) into (5.4):

\[
\hat{y}(t|\theta) = B(q)u(t) + [1 - A(q)]y(t). 
\]  

Next, the vector \( \varphi(t) \) is introduced:

\[
\varphi(t) = [ -y(t-1) \ldots -y(t-n_a) \ u(t-1) \ldots u(t-n_b) ]^T.
\]  

Then, (5.12) can be rewritten as follows:

\[
\hat{y}(t|\theta) = \varphi^T(\theta) = \varphi^T(t)\theta.
\]  

This is the important property of (5.5), referred to in Section 5.3.1. The predictor \( \hat{y}(t|\theta) \) is a scalar product between a known data vector \( \varphi(t) \) and the parameter vector \( \theta \). Such a model is called a linear regression model and the vector \( \varphi(t) \) is known as the regression vector. “Regress” here alludes to the fact that \( \hat{y}(t|\theta) \) is described by going back to \( \varphi(t) \). It is of importance, since powerful and simple estimation methods can be applied for the determination of \( \theta \), see [7, Section 2.2]. The estimation of \( \theta \) in linear regressions will be treated in Section 6.3.1.
5.3.3 Additional Equation-Error-Type Model Structures

A basic disadvantage of the input-output relationship in (5.5) is that there is insufficient freedom in describing the properties of the disturbance term \(e(t)\). In order to add flexibility to that, the equation error is described as a moving average of white noise. This results in the following model:

\[
y(t) + a_1 y(t-1) + \cdots + a_{n_a} y(t-n_a) = \]
\[
= b_1 u(t-1) + \cdots + b_{n_b} u(t-n_b) + e(t) + c_1 e(t-1) + \cdots + c_{n_c} e(t-n_c). \tag{5.15}
\]

The corresponding parameter vector is given by:

\[
\theta = \begin{bmatrix} a_1 & \cdots & a_{n_a} & b_1 & \cdots & b_{n_b} & c_1 & \cdots & c_{n_c} \end{bmatrix}^T. \tag{5.16}
\]

Introduction of \(C(q)\) according to:

\[
C(q) = 1 + c_1 q^{-1} + \cdots + c_{n_c} q^{-n_c}, \tag{5.17}
\]

makes it possible to rewrite (5.15) as follows:

\[
A(q)y(t) = B(q)u(t) + C(q)e(t). \tag{5.18}
\]

Obviously, (5.18) corresponds to (5.3) with:

\[
 G(q, \theta) = \frac{B(q)}{A(q)} \tag{5.19}
\]

\[
H(q, \theta) = \frac{C(q)}{A(q)}. \tag{5.20}
\]

The difference between the models (5.18) and (5.9) is found in the term \(C(q)e(t)\), which is called the moving average (MA). Consequently, the model (5.18) is referred to as ARMAX.

The predictor for the ARMAX model (5.18) is obtained by inserting (5.19) and (5.20) into (5.4). This gives:

\[
\hat{y}(t|\theta) = \frac{B(q)}{C(q)} u(t) + \left[1 - \frac{A(q)}{C(q)}\right] y(t). \tag{5.21}
\]

Manipulation of (5.21) and introduction of additional definitions leads to the following expression for the predictor:

\[
\hat{y}(t|\theta) = \varphi^T(t, \theta) \theta, \tag{5.22}
\]

see [26, Section 4.2]. This expression is similar to (5.14), although it is no linear regression, due to the nonlinear effect of \(\theta\) in the vector \(\varphi(t, \theta)\). Therefore, (5.22) is called a pseudolinear regression.

Instead of modeling the equation error as a moving average, see (5.15), it can also be described as an autoregression. This results in the model structure:

\[
A(q)y(t) = B(q)u(t) + \frac{1}{D(q)} e(t), \tag{5.23}
\]

\[
\quad A(q) = 1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a}, \quad B(q) = b_1 q^{-1} + \cdots + b_{n_b} q^{-n_b}, \quad C(q) = 1 + c_1 q^{-1} + \cdots + c_{n_c} q^{-n_c}.
\]
5.3 Transfer Function Models

with the definition:

\[ D(q) = 1 + d_1 q^{-1} + \cdots + d_{n_d} q^{-n_d}. \] (5.24)

Analogously to the previous terminology, (5.23) is called an ARARX model.

An even more general structure is obtained when an ARMA description of the equation error is used:

\[ A(q)y(t) = B(q)u(t) + \frac{C(q)}{D(q)} e(t). \] (5.25)

This structure is called an ARARMAX model, which contains (5.9), (5.18) and (5.23) as special cases.

5.3.4 Output Error Model Structure

It can be concluded from Sections 5.3.1 and 5.3.3 that the equation error model structures all correspond to descriptions where the transfer functions \( G \) and \( H \) have the polynomial \( A \) as a common factor in the denominator. From a physical point of view, it may seem more natural to parameterize these transfer functions independently.

In order to do this, the relation between input \( u \) and undisturbed output \( w \) is written as a linear difference equation:

\[ w(t) + f_1 w(t-1) + \cdots + f_{n_f} w(t-n_f) = b_1 u(t-1) + \cdots + b_{n_b} u(t-n_b). \] (5.26)

This relation is similar to (5.5) with the term \( e(t) \) removed. Subsequently, it is assumed that the disturbances consist of white measurement noise, which results in the following description:

\[ y(t) = w(t) + e(t). \] (5.27)

Introduction of \( F(q) \) according to:

\[ F(q) = 1 + f_1 q^{-1} + \cdots + f_{n_f} q^{-n_f}, \] (5.28)

makes it possible to rewrite (5.27) as:

\[ y(t) = \frac{B(q)}{F(q)} u(t) + e(t). \] (5.29)

A block diagram of this model is depicted in Figure 5.2.

![Figure 5.2: Output error model structure.](image)

From Figure 5.2, it can be concluded that the disturbance term \( e(t) \) directly enters in the output \( y(t) \). For this reason, (5.29) is called an output error (OE) model structure. The corresponding parameter vector is given by:

\[ \theta = \begin{bmatrix} b_1 & \cdots & b_{n_b} & f_1 & \cdots & f_{n_f} \end{bmatrix}^T. \] (5.30)
As a result of the fact that \( w(t) \) in (5.27) is not observed, it should carry an index \( \theta \), since it is determined from \( u \) using (5.26). This leads to the following notation:

\[
    w(t, \theta) + f_1 w(t-1, \theta) + \cdots + f_{n_f} w(t-n_f, \theta) = b_1 u(t-1) + \cdots + b_{n_b} u(t-n_b). \tag{5.31}
\]

Next, the predictor is computed, using (5.4):

\[
    \hat{y}(t|\theta) = \frac{B(q)}{F(q)} u(t) = w(t, \theta). \tag{5.32}
\]

From (5.32) and (5.31), it follows that the predictor \( \hat{y}(t|\theta) \) is constructed from past inputs only. This also appears from the vector \( \varphi(t, \theta) \):

\[
    \varphi(t, \theta) = \begin{bmatrix} u(t-1) & \cdots & u(t-n_b) & -w(t-1, \theta) & \cdots & -w(t-n_f, \theta) \end{bmatrix}^T, \tag{5.33}
\]

which can be used to rewrite (5.32) as follows:

\[
    \hat{y}(t|\theta) = \varphi^T(t, \theta) \theta, \tag{5.34}
\]

which is similar to the ARMAX model predictor (5.22). Notice that in (5.33), \( w(t-k, \theta) \) for \( k = 1, \ldots, n_f \) are not observed. However, they can be obtained from (5.32), because it holds that \( \hat{y}(t-k|\theta) = w(t-k, \theta) \), for \( k = 1, \ldots, n_f \).

A more general description of the output error model (5.29) is obtained by application of an ARMA description of the output error. This gives the model:

\[
    y(t) = \frac{B(q)}{F(q)} u(t) + \frac{C(q)}{D(q)} e(t). \tag{5.35}
\]

This structure is referred to as the Box-Jenkins (BJ) model structure. Notice that the transfer functions \( G \) and \( H \) are independently parameterized rational functions. The relationship to (5.3) as well as the expression for the predictor are straightforward.

### 5.3.5 General Model Structure

In Sections 5.3.1, 5.3.3 and 5.3.4, several different model structures are discussed. These model structures can be obtained by using five different polynomials, denoted by \( A, B, C, D \) and \( F \). In this context, it is convenient to set up a generalized model structure as follows:

\[
    A(q)y(t) = \frac{B(q)}{F(q)} u(t) + \frac{C(q)}{D(q)} e(t). \tag{5.36}
\]

The block diagram of this generalized model structure is depicted in Figure 5.3. Notice that this structure is too general for most practical purposes. In applications, one or several of the five polynomials will be fixed to unity. The most common special cases of (5.36) are summarized in Table 5.1.
Figure 5.3: Generalized model structure.

Table 5.1: Black-box SISO model structures.

<table>
<thead>
<tr>
<th>Polynomials used in (5.36)</th>
<th>Name of model structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>AR</td>
</tr>
<tr>
<td>B</td>
<td>FIR (Finite Impulse Response)</td>
</tr>
<tr>
<td>C</td>
<td>MA</td>
</tr>
<tr>
<td>AB</td>
<td>ARX</td>
</tr>
<tr>
<td>AC</td>
<td>ARMA</td>
</tr>
<tr>
<td>ABC</td>
<td>ARMAX</td>
</tr>
<tr>
<td>ABD</td>
<td>ARARX</td>
</tr>
<tr>
<td>ABCD</td>
<td>ARARMAX</td>
</tr>
<tr>
<td>BF</td>
<td>OE (Output Error)</td>
</tr>
<tr>
<td>BFCD</td>
<td>BJ (Box-Jenkins)</td>
</tr>
</tbody>
</table>
Chapter 6

Parameter Estimation Methods

6.1 Introduction

In the context of adaptive control and Machine-In-the-Loop control optimization, on-line determination of process parameters is considered to be a key element. In a self-tuning regulator, for instance, a recursive parameter estimator is incorporated as an explicit component, see Chapter 8. Parameter estimation also occurs implicitly in a model reference adaptive controller, see Chapter 9. With respect to Machine-In-the-Loop control optimization, it is expected that the optimization of unknown control parameters in certain control structures bears a strong resemblance to the estimation of parameters in model structures.

In Chapter 5, several different parameterizations of a general model set are described. These parameterizations, also called model structures, incorporate a finite number of coefficients that often cannot be determined a priori from knowledge of the physical mechanisms that govern the system's behaviour. As a result, these coefficients enter the model structure as parameters to be determined. The determination of the unknown parameters needs to be performed by estimation procedures, which is the subject of this chapter.

There are many different ways to estimate the unknown parameters. In this chapter, attention is focused on so-called prediction error methods (PEM’s), see Section 6.2. These methods are based on the importance of the capability of a model to predict future values of the output signal. The model is said to be good in case it produces small prediction errors when applied to observed data. In this context, two approaches are discussed that can be used to estimate the unknown parameters. The first possibility is to minimize a scalar-valued norm or criterion function that measures the size of the prediction error. The second possibility is to demand that the prediction error be uncorrelated with a given data sequence. Both approaches are discussed in Section 6.3 and Section 6.4, respectively. Finally, Section 6.5 deals with excitation conditions, which may be of crucial importance in parameter estimation methods.

6.2 Minimization of Prediction Errors

Suppose a certain model structure $\mathcal{M}$ has been selected and it is parameterized using the parameter vector $\theta$, leading to a particular model $\mathcal{M}(\theta)$. The search for the best model then simplifies to the problem of estimating the parameter vector $\theta$.

For each model $\mathcal{M}(\theta)$, with $\theta$ unknown, it is possible to derive a predictor, see Chapter 4,
for instance. Furthermore, it is possible to obtain a batch of data from the system:

\[
Z^N = \begin{bmatrix} y(1) & u(1) & y(2) & u(2) & \ldots & y(N) & u(N) \end{bmatrix}.
\]  

(6.1)

Subsequently, the question raises how to use the information contained in \(Z^N\) to select a proper value \(\hat{\theta}_N\) of the parameter vector, leading to an adequate model \(\mathcal{M}(\hat{\theta}_N)\) of the dynamical system under consideration. The way on which this is done is called a parameter estimation method.

As already mentioned in Section 6.1, the capability of a model to predict future values of signals is used as a performance criterion in prediction error methods. In case the data sequence \(Z^N\) is available, the prediction error given by a specific model \(\mathcal{M}(\theta_*)\) can be determined at every time instant \(t = 1, \ldots, N\) by comparing the actual measurement \(y(t)\) and the prediction \(\hat{y}(t|\theta_*)\). Consequently, a different prediction error \(\varepsilon(t, \theta_*)\) is obtained for a different parameter vector \(\theta_*\):

\[
\varepsilon(t, \theta_*) = y(t) - \hat{y}(t|\theta_*).
\]  

(6.2)

Intuitively, a model is considered to be good if it produces prediction errors that are sufficiently small. Furthermore, due to the fact that various predictor functions \(\hat{y}(t|\theta_*)\) can be chosen, a considerable freedom in defining "accurate" models in terms of prediction performance is obtained. However, it remains to be discussed what to understand by "small" prediction errors. Two different approaches that can be used to qualify what "small" should mean are considered in this chapter. The first approach is to form a scalar-valued norm or criterion function that measures the size of \(\varepsilon(t, \hat{\theta}_N)\). The second approach is to require that \(\varepsilon(t, \hat{\theta}_N)\) be uncorrelated with a specific data sequence. Both possibilities are discussed in Section 6.3 and Section 6.4, respectively.

### 6.3 Criterion Function Minimization Approach

The sequence of prediction errors in (6.2) can be seen as a vector in \(\mathbb{R}^N\). Consequently, the size of this vector can be measured using any norm, see Section 2.4, which leaves a substantial amount of choices. However, the freedom in the evaluation of the size of the prediction error sequence will be restricted by only considering the following procedure. First, let the prediction error be filtered through a stable linear filter \(L(q)\):

\[
\varepsilon_F(t, \theta) = L(q) \varepsilon(t, \theta),
\]  

(6.3)

for \(1 \leq t \leq N\). Subsequently, the following norm is defined:

\[
V_N(\theta, Z^N) = \frac{1}{N} \sum_{t=1}^{N} \ell(\varepsilon_F(t, \theta)),
\]  

(6.4)

where \(\ell(\cdot)\) is a scalar-valued, typically positive function. Hence, the function \(V_N(\theta, Z^N)\) is a well-defined scalar-valued function of the parameter vector \(\theta\) for given \(Z^N\). Next, it is possible to define the estimate \(\hat{\theta}_N\) by minimization of (6.4):

\[
\hat{\theta}_N = \arg\min_\theta V_N(\theta, Z^N).
\]  

(6.5)
This way of estimating the parameter vector $\theta$ contains many different procedures that are referred to by the general term prediction error methods (PEM’s). In Sections 6.3.1 and 6.3.2, attention will be focused on two methods that belong to the class of prediction error methods. First, however, some remarks are given on the choices of $L$ and $\ell$ in (6.3) and (6.4).

**Remark (Choice of $L$).** By making an adequate choice for the linear filter $L$, it is possible to influence the criterion function in a specific way. In particular, the choice of $L$ offers the possibility to enhance or suppress the relative importance of specific frequency regions in the criterion function. In other words, the filter $L$ acts like frequency weighting.

Substitution of (6.2) for the prediction error into (6.3) leads to the following expression:

$$
\varepsilon_F(t, \theta) = L(q)H^{-1}(q, \theta)[y(t) - G(q, \theta)u(t)].
$$

From (6.6), it follows that the effect of prefiltering the prediction error in (6.3) is identical to changing the noise model $H(q, \theta)$ of the models in the model set from $H(q, \theta)$ to $L^{-1}(q)H(q, \theta)$. For this reason, $L(q)$ is usually chosen equal to $L(q) = 1$.

**Remark (Choice of $\ell$).** In order to motivate a particular choice for $\ell$, consider the following proposition that is obtained from [37, Section 5.5].

**Proposition 1.** Consider the cost function $V(\theta) = \bar{E}\varepsilon^2(t, \theta)$ with the following definitions:

$$
\varepsilon(t, \theta) = H^{-1}(q, \theta)[y(t) - G(q, \theta)u(t)],
$$

$$
y(t) = G_0(q)u(t) + H_0(q)e(t),
$$

where $e(t)$ is white noise with variance $\lambda$. Then $V(\theta) \geq \lambda$ for all $\theta$, with equality for $\hat{\theta}$ if the following relations hold:

$$
G(q, \hat{\theta}) = G_0(q),
$$

$$
H(q, \hat{\theta}) = H_0(q).
$$

For a proof, the reader is referred to [37, Section 5.5].

This proposition shows that minimization of the power of the prediction error signal results in correct estimates of the system dynamics. Therefore, a reasonable choice of $\ell$ is a quadratic norm:

$$
\ell(\varepsilon(t, \theta)) = \frac{1}{2}\varepsilon^2(t, \theta).
$$

This is a standard choice, which is convenient both for computation and analysis. One disadvantage of the quadratic norm, however, is its relatively high sensitivity to outliers in the data. It is also possible to introduce several generalizations of the quadratic norm, which may be convenient in case one has to deal with measurement data that is considered to be of varying reliability over the length of the time interval. Detailed information about this can be found in [26, Section 7.2].

### 6.3.1 Least Squares Method

**Linear Regressions**

In case the chosen model structure has a corresponding one-step-ahead predictor that is linear in the unknown parameter vector $\theta$, this appears to have important consequences for
the determination of this parameter vector. With respect to the model structures discussed in Chapter 5, this concerns the ARX and FIR model structures, see Section 5.3.2. In these cases, the specific property of the predictor being linear in the unknown parameters can be exploited in solving the parameter estimation problem.

In Section 5.3.2, it is shown that the one-step-ahead predictor for the ARX model structure is given by:

\[ \hat{y}(t|\theta) = \varphi^T(t)\theta. \] (6.12)

Here, the regression vector \( \varphi(t) \) and the parameter vector \( \theta \) are given by the following expressions:

\[
\varphi(t) = \begin{bmatrix} -y(t-1) & \ldots & -y(t-n_a) & u(t-1) & \ldots & u(t-n_b) \end{bmatrix}^T
\] (6.13)

\[
\theta = \begin{bmatrix} a_1 & \ldots & a_{n_a} & b_1 & \ldots & b_{n_b} \end{bmatrix}^T.
\] (6.14)

Substitution of (6.12) into (6.2) leads to the following expression for the prediction error:

\[ \varepsilon(t, \theta) = y(t) - \varphi^T(t)\theta. \] (6.15)

Next, the criterion function is computed, using (6.4):

\[ V_N(\theta, Z^N) = \frac{1}{N} \sum_{t=1}^{N} \frac{1}{2} |y(t) - \varphi^T(t)\theta|^2, \] (6.16)

where \( L \) and \( \ell \) are chosen equal to \( L(q) = 1 \) and \( \ell(\varepsilon) = \frac{1}{2} \varepsilon^2 \). A motivation for these choices is given by the remarks above. The criterion function (6.16) is known as the least squares criterion for the linear regression (6.12). It follows that this criterion is quadratic in both the prediction error \( \varepsilon(t, \theta) \) and the parameter vector \( \theta \). As a result, (6.16) can be minimized analytically, leading to:

\[
\hat{\theta}_N^{LS} = \arg \min_{\theta} V_N(\theta, Z^N) = \left[ \frac{1}{N} \sum_{t=1}^{N} \varphi(t)\varphi^T(t) \right]^{-1} \frac{1}{N} \sum_{t=1}^{N} \varphi(t)y(t),
\] (6.17)

provided that the indicated inverse exists. The expression is referred to as the least squares estimate (LSE). Obviously, the least squares estimate is very appealing in the context of Machine-In-the-Loop control optimization. In case of a control structure with a linear parameterization and a quadratic criterion, typically the tracking error squared, (6.17) provides a simple expression for the optimal parameter vector. Introduction of the \( d \times d \) matrix \( R(N) \) and the \( d \)-dimensional column vector \( f(N) \) according to:

\[
R(N) = \frac{1}{N} \sum_{t=1}^{N} \varphi(t)\varphi^T(t)
\] (6.18)

\[
f(N) = \frac{1}{N} \sum_{t=1}^{N} \varphi(t)y(t),
\] (6.19)

\(^{1}\) The solution to the quadratic optimization problem can be found by setting the first derivative of (6.16) to zero and calculating the parameter vector that corresponds to this zero first derivative, see [37, Section 5.6.2] and [7, Section 2.2].
makes it possible to rewrite (6.17) as follows:

$$\hat{\theta}_{N}^{LS} = R^{-1}(N)f(N).$$

(6.20)

In the case (6.13), \(\varphi(t)\) contains lagged input and output variables. Consequently, the entries of (6.18) and (6.19) will consist of estimates of the covariance functions of \(\{y(t)\}\) and \(\{u(t)\}\). Hence, the LSE can be computed using only such estimates and is, therefore, related to correlation analysis, see [26, Section 7.3].

Remark. The notion of the one-step-ahead predictor being linear in the unknown parameter vector is not related to a notion of linearity/nonlinearity of the model that is identified. In other words, the solution to the linear regression problem can also directly be used for identification of models with nonlinear dynamics, provided that the structure of the nonlinearity is specified beforehand. Hence, the notion of the predictor being linear in the parameters only reflects a property of the model structure.

A geometric and a statistical interpretation of the least squares problem can be found in [7, Section 2.2]. Furthermore, in adaptive controllers the observations are obtained sequentially in real time. It is then desirable to make the computations recursively to save computation time. A recursive least squares parameter estimation algorithm is presented in [7, Section 2.2].

Properties of Least Squares Estimate

Suppose that data is obtained from the following data generating system:

$$y(t) = \varphi^{T}(t)\theta_0 + v_0(t),$$

(6.21)

where \(\theta_0\) denotes the parameter vector of the system and \(v_0(t)\) is some sequence.

Next, the notion of consistency is investigated. Consistency implies that an estimate converges to the true parameter value as the number of observations increases toward infinity. The parameter estimate \(\hat{\theta}_{N}^{LS}\) is obtained by substitution of (6.21) into (6.17):

$$\hat{\theta}_{N}^{LS} = R^{-1}(N)\frac{1}{N} \sum_{t=1}^{N} \varphi(t)[\varphi^{T}(t)\theta_0 + v_0(t)]$$

(6.22)

$$= R^{-1}(N)\frac{1}{N} \sum_{t=1}^{N} \varphi(t)v_0(t) + \theta_0.$$

(6.23)

Consistency of the estimate follows if \(\lim_{N \to \infty} \hat{\theta}_{N}^{LS} = \theta_0\). In order to achieve this, two conditions have to be satisfied. These conditions are stated by making use of the following relations:

$$R^* = \lim_{N \to \infty} R(N) = E\varphi(t)\varphi^{T}(t)$$

(6.24)

$$f^* = \lim_{N \to \infty} \frac{1}{N} \sum_{t=1}^{N} \varphi(t)v_0(t) = E\varphi(t)v_0(t),$$

(6.25)

which are valid provided that \(v_0(t)\) and \(\varphi(t)\) are quasi-stationary, see [26, Section 7.3].

- \(R^*\) should exist and be nonsingular. This condition will be satisfied if the input signal \(u(t)\) is sufficiently exciting the process. In this way, all process dynamics are present in the measurement data. More information with respect to excitation conditions can be found in Section 6.5.
The sequence $v_0(t)$ is a white noise sequence. Then, $v_0(t)$ will not depend on what happened up to time $t-1$ and, hence, it holds that $E\varphi(t)v_0(t) = 0$.

- The input sequence $u(t)$ is independent of the zero mean sequence $v_0(t)$ and $n_a = 0$ in (6.13). Then, $\varphi(t)$ only contains terms of $u$ and it holds that $E\varphi(t)v_0(t) = 0$. This condition allows $v_0(t)$ to be coloured.

The variance of an estimate is a measure for the variation that occurs in the estimate in case different realizations of the noise process $e$ are taken. The variance of the least squares parameter estimate can be written as follows:

$$\text{cov}(\hat{\theta}_N^{LS}) = E((\hat{\theta}_N^{LS} - \theta_0)(\hat{\theta}_N^{LS} - \theta_0)^T),$$

where use is made of the fact that $E\hat{\theta}_N^{LS} = \theta_0$, which implies that the estimate is unbiased. By making use of several relations, it can be shown that (6.26) can be transformed into the following approximation:

$$\text{cov}(\hat{\theta}_N^{LS}) \approx \frac{\lambda}{N}(R^*)^{-1},$$

see [37, Section 5.6.4]. Here, $\lambda$ is the variance of the white noise process $e$. From (6.27), it follows that the variance of $\hat{\theta}_N^{LS}$ tends to zero in case $N$ tends toward infinity. This is a direct consequence of the consistency property of the parameter estimate. Furthermore, (6.27) shows that the variance of the parameter estimate is influenced by three elements. These are the noise power $\lambda$, the length of the measurement interval $N$ and the input signal properties, via $R^*$.

### 6.3.2 Maximum Likelihood Method

A different way to obtain an estimate for the parameter vector is given by the so-called maximum likelihood (ML) method. This method originates from a statistical background. The principle of maximum likelihood will be highlighted in this section. For more details, the reader is referred to [26, Section 7.4].

The area of parameter estimation is concerned with the problem of extracting information from measurement data that is corrupted with noise. In this case, the observations are described as realizations of stochastic processes. Assume that the observations are represented by the following random vector:

$$y^N = [y(1) \ y(2) \ \ldots \ y(N)]^T,$$

taking values in $\mathbb{R}^N$. The probability density function, see [13, Section 2.7.3], of the vector $y^N$ is assumed to be as follows:

$$f(\theta; x_1, x_2, \ldots, x_N) = f_y(\theta; x^N),$$

where no restrictions apply. Consequently, it holds that:

$$P(y^N \in A) = \int_{x^N \in A} f_y(\theta; x^N) dx^N,$$
6.4 Correlation Approach

as a result of the definition of the probability density function, see [13, Section 2.7.3]. Recall that in (6.30), \( P \) denotes probabilities. Moreover, the variable \( \theta \) in (6.29) and (6.30) denotes a \( d \)-dimensional parameter vector that describes properties of the observed variable. Obviously, the parameter vector \( \theta \) is unknown and needs to be estimated by making use of the observations \( y^N \). In order to achieve this, an estimator \( \hat{\theta}(y^N) \) is used, which is a function from \( \mathbb{R}^N \) to \( \mathbb{R}^d \).

In this section, attention is focused on the maximum likelihood estimator, although other estimator functions are possible. The maximum likelihood estimator maximizes the probability of the observed event. Its operating principle will be discussed briefly. The joint probability density function for the observed random vector, denoted by \( y^*_N \), is given by (6.29). The probability that the observation indeed should take the value \( y^*_N \) is thus proportional to \( f_y(\theta; y^*_N) \). The function \( f_y(\theta; y^*_N) \) is called the likelihood function and is a deterministic function of \( \theta \) after substitution of \( y^*_N \). It reflects the "likelihood" that the observed event should indeed take place. Next, it is possible to choose the parameter vector \( \theta \) in such a way that the observed event becomes as likely as possible. Formally, this can be stated by:

\[
\hat{\theta}^{ML}(y^*_N) = \arg \max_{\theta} f_y(\theta; y^*_N),
\]

where the maximization is performed for fixed \( y^*_N \). This function is known as the maximum likelihood estimator (MLE). A discussion about the relation between the maximum likelihood method and the prediction error criterion (6.5) can be found in [26, Section 7.4].

### 6.4 Correlation Approach

In Section 6.3.1, it appeared that the least squares method is easy to apply in case of linear regression problems. However, it is shown that the parameter estimates are consistent only under restrictive conditions, which is considered to be a substantial drawback. For this reason, an approach based on correlation issues is considered in this section. During the derivation of the method, it will appear that the least squares method can be seen as a special case.

A different characterization of a good model is that the prediction error \( \varepsilon(t, \theta) \) should be independent of past data, denoted by \( Z^{t-1} \). Otherwise, that is if \( \varepsilon(t, \theta) \) is correlated with \( Z^{t-1} \), there is more information available in the measurements \( Z^{t-1} \) about \( y(t) \) than accounted for by the predictor \( \hat{y}(t|\theta) \). In other words, the predictor is not ideal. Hence, it can be concluded that a good model is characterized as a model that produces prediction errors \( \varepsilon(t, \theta) \) that are uncorrelated with past data \( Z^{t-1} \).

In order to investigate the correlation between \( \varepsilon(t, \theta) \) and \( Z^{t-1} \), a certain vector sequence \( \{\xi(t)\} \) derived from \( Z^{t-1} \) and a certain transformation of \( \{\varepsilon(t, \theta)\} \) are constructed. Both sequences are uncorrelated in case the following condition is satisfied:

\[
\frac{1}{N} \sum_{t=1}^{N} \xi(t) \sigma(\varepsilon(t, \theta)) = 0. \tag{6.32}
\]

Subsequently, it is possible to derive the estimate \( \hat{\theta}_N \) that satisfies (6.32). Notice that the transformation is denoted by \( \sigma(\varepsilon) \).

A generalization of the concept described above is given by the following algorithm.
Algorithm 1. Choose a stable linear filter $L(q)$ and let the prediction error sequence be filtered according to:

$$\varepsilon_F(t, \theta) = L(q)\varepsilon(t, \theta).$$

(6.33)

Next, a choice is made for $\zeta(t)$, which may depend on properties of the system, represented by $\theta$. For this reason, choose a sequence of correlation vectors according to:

$$\zeta(t, \theta) = \zeta(t, Z^{t-1}, \theta).$$

(6.34)

Finally, choose a transformation function $\alpha(\varepsilon)$. Then calculate the estimate $\hat{\theta}_N$:

$$\hat{\theta}_N = \text{sol}_{\theta}[f_N(\theta, Z^N) = 0]$$

(6.35)

$$f_N(\theta, Z^N) = \frac{1}{N} \sum_{t=1}^{N} \zeta(t, \theta) \alpha(\varepsilon_F(t, \theta)).$$

(6.36)

Normally, the dimension of $\zeta(t, \theta)$ is chosen in such a way that $f_N(\theta, Z^N)$ is a $d$-dimensional vector\textsuperscript{2}, which implies that (6.35) has as many equations as unknowns. In some cases, however, an augmented sequence $\zeta(t, \theta)$ of higher dimension than $d$ may be chosen, resulting in (6.35) to be an overdetermined set of equations. Solution strategies to this kind of extended estimation problems can be found in [32, Section 8.1].

Obviously, Algorithm 1 is a very general procedure. Different results will be obtained depending on which model structures it is applied to and on the particular choices of $\zeta(t, \theta)$. In the next section, attention will be focused on the instrumental variable (IV) method, which is a specific case of Algorithm 1.

6.4.1 Instrumental Variable Method

The instrumental variable method is an application of Algorithm 1 to the linear regression model defined by:

$$\hat{y}(t|\theta) = \varphi^T(t)\theta.$$

(6.37)

In addition, with respect to the design choices in Algorithm 1 it is required that $L(q) = 1$, $\zeta(t, \theta) = \zeta(t)$ and $\alpha(\varepsilon) = \varepsilon$. Then, the elements of $\zeta(t)$ are called instruments or instrumental variables. This leads to the following parameter estimate:

$$\hat{\theta}_{IV} = \text{sol}_{\theta} \left[ \frac{1}{N} \sum_{t=1}^{N} \zeta(t) [y(t) - \varphi^T(t)\theta] = 0 \right],$$

(6.38)

which can alternatively be written as:

$$\hat{\theta}_{IV} = \left[ \frac{1}{N} \sum_{t=1}^{N} \zeta(t)\varphi^T(t) \right]^{-1} \frac{1}{N} \sum_{t=1}^{N} \zeta(t)y(t),$$

(6.39)

provided that the indicated inverse exists. Next, it is possible to investigate the relation between the least squares estimate and the instrumental variable estimate. Comparison of

\textsuperscript{2}In order to achieve this, $\zeta(t, \theta)$ is required to have dimension $d \times p$ if the output has dimension $p$. 
6.4 Correlation Approach

(6.39) and (6.17) reveals that in case $\zeta(t)$ is chosen equal to $\varphi(t)$ for all $t$, the same estimates result for both methods. The objective is, however, to choose a different vector sequence $\zeta(t)$ in order to avoid the restrictive conditions with respect to consistency of the least squares estimate. The reason for the conditions is the correlation between $v_0(t)$ and $\varphi(t)$, which is obvious if the data generating system is assumed to be represented by:

$$y(t) = \varphi^T(t)\theta_0 + v_0(t).$$

(6.40)

Another consistency analysis, see [26, Section 5.6.5], reveals that in case of the instrumental variable method the following two conditions must hold:

- $\overline{E}\zeta(t)\varphi^T(t)$ should be nonsingular. This condition implies that the instruments should be correlated with the variables in the regression vector $\varphi(t)$.
- $\overline{E}\zeta(t)v_0(t)$ should be zero. This condition implies that the instruments should be uncorrelated with the disturbance term $v_0(t)$.

Next, several possible choices for the instrumental variables are discussed.

Assume that (6.40) is an ARX model:

$$y(t) + a_1 y(t-1) + \cdots + a_n y(t-n_a) = b_1 u(t-1) + \cdots + b_n u(t-n_b) + v(t).$$

(6.41)

In the situation that $v(t)$ is a white noise process, the choice $\zeta(t) = \varphi(t)$ satisfies both conditions. However, if $v(t)$ is not a white noise process then the entries of the regression vector $\varphi(t)$ that depend on the output $y(t)$ have to be replaced by other (instrumental) signals that are not correlated with the disturbance term $v(t)$. A straightforward choice for the instruments is:

$$\zeta(t) = \begin{bmatrix} -x(t-1) & -x(t-2) & \ldots & -x(t-n_a) & u(t-1) & \ldots & u(t-n_b) \end{bmatrix}^T.$$

(6.42)

Here, $x(t)$ is generated from the input through a linear system:

$$N(q)x(t) = M(q)u(t),$$

(6.43)

where:

$$N(q) = 1 + n_1 q^{-1} + \cdots + n_n q^{-n_a},$$

(6.44)

$$M(q) = m_0 + m_1 q^{-1} + \cdots + m_m q^{-m_b}.$$  

(6.45)

With respect to the conditions above, it can be concluded that the first condition holds in general, since both $\varphi$ and $\zeta$ are generated from the same input sequence. Moreover, the second condition holds if the input is generated in open loop, so that it does not depend on the noise $v(t)$ in the system. Different strategies are required for systems that operate in closed loop and for systems without inputs, see [32, Section 10.2] and [32, Complement C8.1], respectively.

---

3In this case, sample means is replaced by expectation.

4Notice that strictly speaking the coefficients must be indexed by "zero".

5A simple and appealing approach is to first apply the least squares method to (6.41) and then use the estimated model for $N$ and $M$. 
6.5 Excitation Conditions

The properties of the measurement data used in parameter estimation are crucial with respect to the quality of the estimates. For example, it is obvious that no information on the dynamic behaviour of the system is present in the output signal in case the input signal is constant. Only the static behaviour of the system can be uniquely determined in this situation. Hence, it is necessary to impose conditions on the character of the input signal that is applied during the experiments. These conditions are referred to as excitation conditions and are the subject of this section.

6.5.1 Persistent Excitation

One way to characterize process inputs is the notion of persistent excitation. A signal is called persistently exciting if it satisfies certain conditions, which follows from the following definition.

**Definition 12 ( Persistently exciting signals).** A signal $u$ is called persistently exciting (PE) of order $n$ if the following limit exists:

$$c(k) = \lim_{t \to \infty} \frac{1}{t} \sum_{i=1}^{t} u(i)u(i-k),$$

for $k = 0, \ldots, n-1$ and if the matrix $C_n$ is positive definite. Here, $C_n$ is defined according to:

$$C_n = \begin{bmatrix} c(0) & c(1) & \cdots & c(n-1) \\ c(1) & c(0) & \cdots & c(n-2) \\ \vdots & \vdots & \ddots & \vdots \\ c(n-1) & c(n-2) & \cdots & c(0) \end{bmatrix},$$

which is a symmetric Toeplitz matrix. In (6.47), $c(k)$ are the empirical covariances of the input as defined according to (6.46). Essentially, the property of persistent excitation reflects the "variation" that is present in the signal $u$.

**Remark.** In literature, an alternative definition of persistent excitation is often used. The signal $u$ is said to be persistently exciting of order $n$ if an integer $m$ exists for all $t$, such that:

$$\rho_1 I > \sum_{k=t}^{t+m} \varphi(k)\varphi^T(k) > \rho_2 I,$$

where $\rho_1, \rho_2 > 0$. The vector $\varphi(t)$ is given by:

$$\varphi(t) = \begin{bmatrix} u(t-1) & u(t-2) & \cdots & u(t-n) \end{bmatrix}.$$

Next, Definition 12 is illustrated by application of a least squares criterion to a FIR model structure.

**Example 4 (Least squares identification of FIR model structure).** The prediction error for the FIR model structure is given by:

$$e(t, \theta) = y(t) - \varphi^T(t)\theta,$$
6.5 Excitation Conditions

see (6.15). Here, the regression vector \( \varphi(t) \) and the parameter vector \( \theta \) are given by the following expressions:

\[
\varphi(t) = \begin{bmatrix} u(t-1) & \ldots & u(t-n_b) \end{bmatrix}^T
\]

\[
\theta = \begin{bmatrix} b_1 & \ldots & b_{n_b} \end{bmatrix}^T.
\]

Computation of the least squares criterion leads to:

\[
V_N(\theta, Z^N) = \frac{1}{N} \sum_{t=1}^{N} \frac{1}{2} [y(t) - \varphi^T(t)\theta]^2,
\]

see (6.16). Minimization of the least squares criterion results in the least squares estimate:

\[
\hat{\theta}_N^{LS} = \arg \min_{\theta} V_N(\theta, Z^N) = \left( \frac{1}{N} \sum_{t=1}^{N} \varphi(t)\varphi^T(t) \right)^{-1} \frac{1}{N} \sum_{t=1}^{N} \varphi(t)y(t).
\]

Obviously, the least squares estimate is the solution to the following equation:

\[
\begin{bmatrix}
    c(0) & c(1) & \ldots & c(n_b - 1) \\
    c(1) & c(0) & c(n_b - 2) & \ldots \\
    \vdots & \ddots & \ddots & \vdots \\
    c(n_b - 1) & c(n_b - 2) & \ldots & c(0)
\end{bmatrix}
\begin{bmatrix}
    \hat{b}_1 \\
    \hat{b}_2 \\
    \vdots \\
    \hat{b}_{n_b}
\end{bmatrix}
= \begin{bmatrix}
    \frac{1}{N} \sum_{t=1}^{N} u(t - 1)y(t) \\
    \frac{1}{N} \sum_{t=1}^{N} u(t - 2)y(t) \\
    \vdots \\
    \frac{1}{N} \sum_{t=1}^{N} u(t - n_b)y(t)
\end{bmatrix}.
\]

Hence, from (6.55) it follows that uniqueness of the estimated parameter vector \( \hat{\theta}_N^{LS} \) is obtained if and only if the symmetric Toeplitz matrix has full rank. Alternatively, it can be stated that the order of persistent excitation of the input signal is equal to the number of FIR parameters that can be identified uniquely by using the criterion (6.53).

An equivalent formulation of the concept of persistent excitation is given in the following theorem.

**Theorem 2 (Persistently exciting signals).** The signal \( u \) that satisfies (6.46) is persistently exciting of order \( n \) if and only if the following inequality holds:

\[
\lim_{t \to \infty} \frac{1}{t} \sum_{k=1}^{t} (A(q)u(k))^2 > 0,
\]

for all nonzero polynomials \( A(q) \) of degree \( n - 1 \) or less. Here, the polynomial \( A(q) \) is defined according to:

\[
A(q) = a_0q^{n-1} + a_1q^{n-2} + \cdots + a_{n-1}.
\]

For a proof, the reader is referred to [7, Section 2.4].

Next, two examples are given in which it is investigated whether specific signals are persistently exciting. Additional examples can be found in [7, Section 2.4].

**Example 5 (White noise).** White noise is a sequence of zero mean independent random variables, see Section 2.3.3. From this definition, it follows that white noise is persistently exciting of any finite order, since for these signals it holds that \( C_n = I_n \) for all \( 1 \leq n \in \mathbb{N} \).
Example 6 (Sinusoid). Let $u(t) = \sin \omega t$. Subsequently, choose the polynomial $A(q)$ as follows:

$$A(q) = q^2 - 2q \cos \omega + 1,$$

for which $n = 3$. Then, it can be concluded that the following equality holds:

$$A(q)u(t) = (q^2 - 2q \cos \omega + 1)u(t) = 0.$$  

Hence, taking into account (6.56), it follows that a sinusoid can at most be PE of order 2. Computation of $C_2$ according to (6.47) leads to the following matrix:

$$C_2 = \frac{1}{2} \begin{bmatrix} 1 & \cos \omega \\ \cos \omega & 1 \end{bmatrix}.$$  

Consequently, a sinusoid is persistently exciting of order 2.

A frequency domain interpretation of PE can be found in [7, Section 2.4] and [37, Section 5.7.3]. Furthermore, in [7, Section 2.4], system identification under closed loop conditions is considered. In this case, the process input signal is generated by feedback, which may give rise to certain difficulties.
Chapter 7

Optimization Techniques

7.1 Introduction

In many control concepts, see Chapters 9 and 10 for instance, the controller parameters are updated in such a way that a certain objective function is minimized. The objective function typically contains terms concerning the tracking error and the control effort. In order to solve the minimization problem in an adequate way, dedicated algorithms are used. For this reason, it is necessary to have a basic knowledge of optimization techniques, which is the subject of this chapter.

Basically, the chapter is divided into two sections. In Section 7.2, a number of fundamental topics in optimization are discussed. The generalized optimization problem is considered in Section 7.2.1. Feasibility and optimality are highlighted in Sections 7.2.2 and 7.2.3, respectively. This is followed by convexity in Section 7.2.4 and the generalized optimization algorithm in Section 7.2.5. In Section 7.3, basics of unconstrained optimization are treated, because this type of optimization problem is considered to be most important with respect to Machine-In-the-Loop control optimization. After highlighting Newton's method for nonlinear equations in Section 7.3.1, several optimality conditions are derived in Section 7.3.2. Subsequently, Newton's method is applied to the minimization problem in Section 7.3.3. This is followed by a discussion about the search direction in Section 7.3.4 and the step length in Section 7.3.5. Finally, the steepest descent method is derived in Section 7.3.6, while quasi-Newton methods are discussed in Section 7.3.7.

Many textbooks on fundamentals of optimization are available, see [28] and [30], for instance. Apart from different points of view, they all more or less contain the same information. For this reason, the discussion in this chapter closely follows [28].

7.2 Fundamentals of Optimization

7.2.1 Generalized Optimization Problem

The generalized optimization problem consists of three basic elements. These basic elements will be discussed briefly, after which a formal definition of the generalized optimization problem will be given.

The first basic element is formed by the design variables $x_1, \ldots, x_n$. These design variables are collected in one vector $x$, according to $x = [x_1, \ldots, x_n]^T$. In optimization, the objective
is to determine values for the design variables in such a way that these values are optimal. This raises the question how to define optimality of the design variables.

For this reason, the so-called objective function \( f(x) \) is introduced, which is the second basic element. The objective function, expressed in terms of the design variables, provides a measure of optimality. Intuitively, the values of the design variables are optimal in case they minimize\(^1\) the objective function. However, there may be several restrictions with respect to the values of the design variables.

These restrictions represent the third basic element and are referred to as constraints. It may be possible that the design variables are not allowed to take arbitrary values. In other words, they must satisfy certain constraints. Generally, a distinction is made between equality constraints and inequality constraints.

Taking into account these basic elements, it is possible to give a formal definition of the generalized optimization problem.

**Definition 13 (Generalized optimization problem).**

1. Select a set of design variables \( x \).
2. Select an objective function \( f(x) \), expressed in terms of the design variables \( x \).
3. Determine a set of constraints \( g_i(x) \), expressed in terms of the design variables \( x \). Distinguish between equality constraints \( g_i(x) = 0 \) and inequality constraints \( g_i(x) \geq 0 \), where \( i \) is an element of an index set, see Section 7.2.2.
4. Determine a set of values for the design variables, which minimizes the objective function, while satisfying all the constraints. This is the generalized optimization problem, which can formally be stated by:

   \[
   \min_x f(x),
   \]

   subject to:

   \[
   g_i(x) = 0, \quad i \in \mathcal{E} \tag{7.2}
   \]

   \[
   g_i(x) \geq 0, \quad i \in \mathcal{T}, \tag{7.3}
   \]

   where \( \mathcal{E} \) is an index set for the equality constraints and \( \mathcal{T} \) is an index set for the inequality constraints.

More details about formal mathematical models of optimization problems can be found in [30, Section 1.2].

**7.2.2 Feasibility**

An important topic in optimization is the notion of feasibility. The definition of a feasible point and a feasible region will be highlighted in this section.

\(^1\)A maximization problem is equivalent to a minimization problem.
Consider a set of constraints of the following form:
\begin{align}
g_i(x) & = 0, \ i \in \mathcal{E} \\
g_i(x) & \geq 0, \ i \in \mathcal{T}.
\end{align}
Here, \( g_i(x) \) are given functions that define the constraints in the optimization model. Furthermore, \( \mathcal{E} \) is an index set for the equality constraints and \( \mathcal{T} \) is an index set for the inequality constraints. It is obvious that any set of equality constraints and inequality constraints can be rearranged in this particular form.

A point that satisfies all these constraints is said to be a feasible point. The set of all feasible points is termed the feasible region or the feasible set and is denoted by \( S \).

At a feasible point \( \bar{x} \), an inequality constraint \( g_i(x) \geq 0 \) is said to be active if \( g_i(\bar{x}) = 0 \) and inactive if \( g_i(\bar{x}) > 0 \). Moreover, the point \( \bar{x} \) is said to be on the boundary of the constraint in the former case and in the interior of the constraint in the latter. Hence, all equality constraints are regarded as active at any feasible point. The active set of constraints at a feasible point is defined as the set of all constraints that are active at that point. The boundary of the feasible region is defined as the set of feasible points for which at least one inequality constraint is active. All other feasible points are called interior points.

### 7.2.3 Optimality

Another important topic in optimization is the notion of optimality. An optimal point is associated with a variety of conditions. Each of these conditions gives rise to a slightly different notion of an optimal solution, as will be shown in this section.

Consider the following \( n \)-dimensional optimization problem:
\[
\min_{x \in S} f(x).
\]

The set \( S \) of all feasible points is usually defined by a set of constraints, see Section 7.2.2. In case no constraints are present, the set \( S \) equals \( \mathbb{R}^n \), i.e., the set of vectors of length \( n \) whose components are real numbers.

The most basic definition of a solution is that \( x_\ast \) minimizes \( f \) if:
\[
f(x_\ast) \leq f(x),
\]
for all \( x \in S \). The point \( x_\ast \) is referred to as a global minimizer of \( f \) in \( S \). If in addition \( x_\ast \) satisfies:
\[
f(x_\ast) < f(x),
\]
for all \( x \in S \) such that \( x \neq x_\ast \), then \( x_\ast \) is referred to as a strict global minimizer. It is important to realize that not all functions have a finite global minimizer. Furthermore, even if a function has a global minimizer there is no guarantee that it will have a strict global minimizer.

In general, it is difficult to find global minimizers. This is due to the fact that many optimization algorithms are based on the Taylor series. The Taylor series provides an approximation of the function, based on information about the function at a single point. Consequently, this approximation is normally valid only within a small neighbourhood of that particular point, see [28, Section 2.6]. An important special case in which it is possible to guarantee
that a global solution is found is the case where the function $f$ and the set $S$ are both convex, see Section 7.2.4.

Although it is difficult to find a global minimizer, it is at least required to find a point that is better than its surrounding points. In other words, it is desired to find a local minimizer of $f$ in $S$, which is a point that satisfies:

$$f(x_*) \leq f(x),$$

for all $x \in S$ such that $\|x - x_*\| < \varepsilon$. Here, $\varepsilon$ is a small positive number whose value may depend on $x_*$. Furthermore, the point $x_*$ is a strict local minimizer if:

$$f(x_*) < f(x),$$

for all $x \in S$ such that $x \neq x_*$ and $\|x - x_*\| < \varepsilon$.

Generally, strict local minimizers can be found by using first-order and second-order derivative values at the point $x = x_*$. This implies that algorithms that compute first and second derivatives of the problem functions can be used to identify these strict local minimizers. Many algorithms, however, in particular those that only compute first derivative values, are only guaranteed to find a stationary point for the problem. Although a local minimizer of $f$ is also a stationary point of $f$, the reverse needs not to be true.

### 7.2.4 Convexity

As already mentioned in Section 7.2.3, there is one important situation where global solutions can be found. This is the case in which the objective function is a convex function and the feasible region is a convex set.

A set $S$ is called convex if, for any elements $x$ and $y$ of $S$, the following holds:

$$\alpha x + (1 - \alpha)y \in S,$$

for all $0 \leq \alpha \leq 1$. In other words, if $x$ and $y$ are in the set $S$, then the line segment connecting $x$ and $y$ is also in $S$. An example of a convex set and a nonconvex set is depicted in Figure 7.1. More generally, it can be stated that every set defined by a system of linear constraints is a convex set.

![Figure 7.1: Example of a convex set and a nonconvex set.](image)

A function $f$ is convex on a convex set $S$ if it satisfies the following inequality:

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y),$$

(7.12)

$^2$A local minimizer that is not a strict local minimizer is often considered to be a special situation.
for all $0 \leq \alpha \leq 1$ and for all $x, y \in S$. This definition states that the line segment connecting the points $(x, f(x))$ and $(y, f(y))$ lies on or above the graph of the function, see Figure 7.2.

![Figure 7.2: Definition of a convex function.](image)

Analogously, a function $f$ is concave on $S$ if it satisfies the following inequality:

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y),$$

for all $0 \leq \alpha \leq 1$ and for all $x, y \in S$. Hence, it can be concluded that linear functions are both convex and concave.

In addition, a function is called strictly convex if it holds that:

$$f(\alpha x + (1 - \alpha)y) < \alpha f(x) + (1 - \alpha)f(y),$$

for all $x \neq y$ and $0 < \alpha < 1$, where $x, y \in S$.

Taking these definitions into account, it is possible to discuss local and global solutions. For this reason, the following convex programming problem is introduced:

$$\min_{x \in S} f(x),$$

subject to:

$$g_i(x) \geq 0,$$

for $i = 1, \ldots, m$. Here, $S$ is a convex set, $f$ is a convex function on $S$ and $g_i(x)$ are concave functions. It can be shown that any local solution of such a convex programming problem is also a global solution. This result is stated by the following theorem and is especially useful in linear programming, since every linear program is a convex programming problem. For general problems, however, it may be difficult to determine if the function $f$ and the set $S$ are both convex, so this result is less useful.

**Theorem 3 (Global solutions of convex programs).** Let $x_*$ be a local minimizer of a convex programming problem. Then, $x_*$ is also a global minimizer. Furthermore, if the objective function is strictly convex, then $x_*$ is a strict global minimizer. For a proof, the reader is referred to [28, Section 2.3].
Fortunately, it is possible to give an alternative definition of convexity in case the function \( f \) has continuous first-order and second-order derivatives. Such a function is convex if and only if the Hessian matrix of second derivatives is positive semi-definite. That is, at every point \( x \in S \) it must hold that:
\[
y'' \nabla^2 f(x) y \geq 0,
\]
for all \( y \). Notice that the vector \( y \) is not restricted to lie in the set \( S \). Alternatively, it is also possible to verify if the eigenvalues of the Hessian matrix are all greater than or equal to zero. In case the Hessian matrix \( \nabla^2 f(x) \) is positive definite for all \( x \in S \), then the function is strictly convex on the set \( S \).

### 7.2.5 Generalized Optimization Algorithm

Numerous algorithms are available that can be used to solve different types of optimization problems. Despite this diversity in algorithms, it can be concluded that all algorithms more or less have the same general form.

**Algorithm 2 (Generalized optimization algorithm).**

- Specify some initial guess of the solution \( x_0 \).
- For \( k = 0, 1, \ldots \)
  - If \( x_k \) is optimal, stop.
  - Determine a search direction \( p_k \).
  - Determine a step length \( \alpha_k \).
  - Determine an improved estimate of the solution according to:
    \[
    x_{k+1} = x_k + \alpha_k p_k.
    \]

With respect to this generalized optimization algorithm, several remarks can be made.

**Remark.** According to the algorithm, the test for optimality and the determination of the new point \( x_{k+1} \) are separate ideas. Often, however, the information obtained from the optimality test is used in the computation of the new point. In case of a one-dimensional problem without constraints, for instance, the optimality test is often based on the first-order derivative. This value also provides information about the behaviour of the function, which is useful in the computation of \( x_{k+1} \).

**Remark.** The computation of \( x_{k+1} \) is divided into two calculations. First, a search direction \( p_k \) is determined that improves the solution in some sense. For an unconstrained problem, it is typically required that the search direction \( p_k \) be a descent direction for the function \( f \) at the point \( x_k \). Once the search direction \( p_k \) has been computed, it is possible to compute the step length \( \alpha_k \) that determines the point \( x_{k+1} \). Ideally, the step length is determined in such a way that the function is minimized in the direction \( p_k \). This is a problem that involves only one variable, the parameter \( \alpha_k \).
Remark. From the algorithm, it can be concluded that the optimal solution is found in an iterative way. This is due to the fact that generally no formulas for the solution exist in case of more complex optimization problems. Moreover, the optimal solution often is an approximate solution, an element of the sequence that is sufficiently accurate. Determination of the exact solution generally requires an infinite number of iterations.

Remark. In the previous remark, it is mentioned that many algorithms do not find a solution in a finite number of steps. Instead, these algorithms compute a sequence of approximate solutions that hopefully become more accurate. Obviously, it is desired to be able to quantify the efficiency of such an algorithm. One measure of efficiency is the rate of convergence. It describes how quickly the estimates of the solution approach the exact solution. More details about rates of convergence can be found in [28, Section 2.5].

7.3 Unconstrained Optimization

In this section, the following optimization problem is solved:

$$\min_x f(x),$$

(7.19)

where no constraints are present on the variables $x = [x_1, \ldots, x_n]^T$. This is a so-called unconstrained optimization problem, which is of particular interest with respect to the control concepts discussed in subsequent chapters. Intuitively, this follows from the fact that the controller parameters generally are unconstrained.

Essentially, the strategy used to solve (7.19) is as follows. First, several optimality conditions for the unconstrained optimization problem are derived. One of these conditions, the first-order necessary condition, consists of a system of nonlinear equations. Application of Newton's method to this system of equations makes it possible to solve the unconstrained optimization problem.

7.3.1 Newton's Method for Nonlinear Equations

One-Dimensional Case

In this section, methods are considered that can be used to solve $^3$:

$$f(x) = 0,$$

(7.20)

where $x$ is a scalar and $f$ is a real-valued function. This is the so-called one-dimensional case. Throughout the section it is assumed that the function $f$ has two continuous derivatives.

If $f(x)$ is a linear function, it is possible to find a solution provided that the system is nonsingular. In the nonlinear case, it is generally not possible to guarantee that a solution can be found. However, there are effective algorithms, based on solving a sequence of linear equations, that can be applied to a wide variety of problems.

These algorithms are based on Newton's method. Given an estimate of the solution $x_k$, the function $f$ is approximated by the linear function consisting of the first two terms of the Taylor series for the function $f$ at the point $x_k$. The resulting linear system is then solved to obtain a new estimate of the solution $x_{k+1}$.

$^3$Notice that this problem and the optimization problem (7.19) are two separate problems.
In order to derive the formulas corresponding to Newton’s method, first the Taylor series for the function \( f \) at the point \( x_k \) is given:

\[
f(x_k + p) \approx f(x_k) + p \dot{f}(x_k).
\]  

(7.21)

In case \( \dot{f}(x_k) \neq 0 \), it is possible to solve:

\[
f(x) \approx f(x_k) + p \dot{f}(x_k) = 0
\]

(7.22)

for \( p \), leading to:

\[
p = -\frac{f(x_k)}{\dot{f}(x_k)}.\]

(7.23)

The new estimate of the solution is then \( x_{k+1} = x_k + p \), which is equivalent to:

\[
x_{k+1} = x_k - \frac{f(x_k)}{\dot{f}(x_k)}.
\]

(7.24)

This is the formula for Newton’s method. Summarizing, it can be stated that Newton’s method corresponds to approximating the function \( f \) by its tangent line at the point \( x_k \). Subsequently, the point where the tangent line crosses the \( x \)-axis is taken as the new estimate of the solution. Newton’s method converges rapidly and it can be shown that the convergence rate is typically quadratic, see [28, Section 2.7]. Unfortunately, there is also a possibility for Newton’s method to fail. This happens if \( \dot{f}(x_k) = 0 \) for some \( k \), leading to a division by zero in the formula. Furthermore, the proof of convergence for Newton’s method requires that the initial point \( x_0 \) be sufficiently close to a zero. Otherwise, the method may possibly fail to converge, even when there is no division by zero in the formula for the method, see [28, Section 2.7].

**Multi-Dimensional Case**

The derivation of the formula for Newton’s method in the \( n \)-dimensional case is similar to the one-dimensional case. Suppose that the following problem is to be solved:

\[
f(x) = 0,
\]

(7.25)

where this represents:

\[
\begin{align*}
f_1(x_1, \ldots, x_n) & = 0 \\
f_2(x_1, \ldots, x_n) & = 0 \\
\vdots & \\
f_n(x_1, \ldots, x_n) & = 0
\end{align*}
\]

(7.26)

Define the matrix \( \nabla f(x) \) with columns \( \nabla f_1(x), \ldots, \nabla f_n(x) \). This matrix is the transpose of the Jacobian of \( f \) at the point \( x \). As before, the Taylor series approximation for the function \( f \) at the point \( x_k \) is given:

\[
f(x_k + p) \approx f(x_k) + \nabla f(x_k)p,
\]

(7.27)
7.3 Unconstrained Optimization

with \( p \) a vector. Next, it is possible to solve:

\[
f(x_*) \approx f(x_k) + \nabla f(x_k)^T p = 0 \tag{7.28}
\]

for \( p \), leading to:

\[
p = -\nabla f(x_k)^T f(x_k). \tag{7.29}
\]

The new estimate of the solution is then given by:

\[
x_{k+1} = x_k + p = x_k - \nabla f(x_k)^T f(x_k), \tag{7.30}
\]

which is the formula for Newton’s method in the \( n \)-dimensional case. With respect to this formula, several remarks can be made, see [28, Section 2.7.1]. Most remarks appear to be analogous to the one-dimensional case and are, therefore, not mentioned here.

7.3.2 Optimality Conditions

In this section, conditions are derived that are satisfied by solutions to the optimization problem:

\[
\min_{x} f(x). \tag{7.31}
\]

Let \( x_* \) denote a candidate solution to this minimization problem. From Section 7.2.3, it follows that the definition of a global minimizer is not very useful in practice, since it requires information about the function at every point. Most algorithms, however, only have information about the function at a finite set of points. For this reason, the objective is to find local minimizers, see Section 7.2.3. The conditions that must be satisfied by a local minimizer are no more practical than those for a global minimizer. This is due to the fact that they require information about the function at an infinite number of points as well. However, practical optimality conditions can be obtained by imposing additional assumptions on the function \( f \).

To this end, assume that the function \( f \) is differentiable and that its first and second derivatives are continuous in a neighbourhood of the point \( x_* \). All conditions will be derived using Taylor series expansions of \( f \) about the point \( x_* \).

Next, suppose that \( x_* \) is a local minimizer of \( f \). Consider the Taylor series with remainder term:

\[
f(x_* + p) = f(x_*) + \nabla f(x_*)^T p + \frac{1}{2} p^T \nabla^2 f(\xi) p, \tag{7.32}
\]

where \( p \) is a nonzero vector and \( \xi \) is a point between \( x \) and \( x_* \). If \( x_* \) is a local minimizer, there can be no feasible descent directions at \( x_* \). Consequently, it must hold that:

\[
\nabla f(x_*)^T p \geq 0, \tag{7.33}
\]

for all feasible directions \( p \). In case of an unconstrained problem, all directions \( p \) are feasible, so the gradient at \( x_* \) must be zero. Hence, if \( x_* \) is a local minimizer of \( f \), it holds that:

\[
\nabla f(x_*) = 0. \tag{7.34}
\]

A point that satisfies this condition is called a stationary point of the function \( f \). The condition (7.34) is referred to as the first-order necessary condition for a minimizer. It is a
necessary condition, since if \( x_* \) is a local minimizer then it necessarily satisfies this condition. The condition is not sufficient, because a point that satisfies \( \nabla f(x_*) = 0 \) can be a local minimizer, a local maximizer or a saddle point.

In order to distinguish various stationary points, it is necessary to examine second derivatives. Consider the Taylor series about \( x = x_* + p \), using condition (7.34):

\[
f(x) = f(x_* + p) = f(x_*) + \frac{1}{2} p^T \nabla^2 f(x_*) p. \tag{7.35}\]

By making use of (7.35), it can be shown that \( \nabla^2 f(x_*) \) must be positive semi-definite. Otherwise, it holds that \( v^T \nabla^2 f(x_*) v < 0 \) for some \( v \). As a result, it is also true that \( v^T \nabla^2 f(\xi)v < 0 \) if \( ||\xi - x_*|| \) is small. This is because \( \nabla^2 f \) is assumed to be continuous at \( x_* \). If \( p \) is chosen as some sufficiently small multiple of \( v \), then the point \( \xi \) will be close enough to \( x_* \) to guarantee that \( f(x) < f(x_*) \), which is a contradiction. Hence, if \( x_* \) is a local minimizer then \( \nabla^2 f(x_*) \) is positive semi-definite. This is referred to as the second-order necessary condition for a minimizer.

There also exists a second-order sufficient condition, that is sufficient to guarantee that \( x_* \) is a local minimizer. If (7.34) holds and \( \nabla^2 f(x_*) \) is positive definite, then \( x_* \) is a strict local minimizer of \( f \). For a proof, the reader is referred to [28, Section 10.2].

### 7.3.3 Newton’s Method for Minimization

The basic form of Newton’s method for minimization is discussed in this section. In subsequent sections, it is shown how the method can be adjusted to guarantee that the search directions are descent directions, to guarantee convergence and to lower the costs of the method.

From Section 7.3.1, it follows that Newton’s method can be used to find a zero of a nonlinear function. In order to use Newton’s method for optimization, it is applied to the first-order necessary condition for a local minimizer:

\[
\nabla f(x) = 0. \tag{7.36}\]

The Jacobian of \( \nabla f(x) \) is given by \( \nabla^2 f(x) \) and using this information in (7.30) leads to:

\[
x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k) = x_k + p_k. \tag{7.37}\]

Here, \( p_k \) is the solution to the Newton equations:

\[
[\nabla^2 f(x_k)]p_k = -\nabla f(x_k). \tag{7.39}\]

Hence, the step \( p_k \) is usually obtained by solving a linear system of equations rather than by computing the inverse of the Hessian.

In Section 7.3.1, Newton’s method is derived by finding a linear approximation to a nonlinear function, using the Taylor series. The formula for Newton’s method represents the step to the zero of this linear approximation. In case of the nonlinear function (7.36), the linear approximation is as follows:

\[
\nabla f(x_k + p_k) \approx \nabla f(x_k) + \nabla^2 f(x_k) p_k. \tag{7.40}\]
This linear approximation is the gradient of the quadratic function:

\[ Q(p_k) = f(x_k) + \nabla f(x_k)^T p_k + \frac{1}{2} p_k^T \nabla^2 f(x_k) p_k. \quad (7.41) \]

Obviously, the quadratic function \( Q(p_k) \) corresponds to the first three terms of a Taylor series about \( x_k \). An interesting point of view is obtained when it is observed that Newton's method approximates \( f(x_k) \) by \( Q(p_k) \) at every iteration, subsequently minimizes \( Q(p_k) \) as a function of \( p_k \) and then updates \( x_{k+1} \) according to (7.38). Hence, the nonlinear function is approximated by a quadratic model at every iteration.

Typically, Newton's method has a quadratic rate of convergence, although cases exist in which it diverges or fails, see Section 7.3.1. If Newton's method converges, it will converge to a stationary point. Moreover, in the basic form discussed in this section, there are no elements in the algorithm that bias the method towards finding a minimum, see Section 7.3.4.

However, Newton's method is rarely used in its basic form. The method is modified in order to make it more reliable and to make it less expensive. Reliability can be improved by using Newton's method in some sort of auxiliary strategy. This makes it possible to guarantee convergence to a stationary point and possibly to a local minimizer, in case one exists. One approach is to use the Newton direction \( p_k \) in the generalized optimization algorithm, see Section 7.2.5. This leads to the following iterative method:

\[ x_{k+1} = x_k + \alpha_k p_k, \quad (7.42) \]

where \( \alpha_k \) is a scalar chosen so that \( f(x_{k+1}) < f(x_k) \). Comparison with (7.38) reveals that in the basic form \( \alpha_k = 1 \) at every iteration, which implies there is no guarantee that the function value is decreased in case of a nonconvex problem. The computational costs in case of Newton's method are given by derivatives, calculations and storage. In order to reduce these costs, many compromises on Newton's method have been proposed. Some of these compromises are highlighted in Sections 7.3.6 and 7.3.7. A disadvantage of the resulting algorithms is that they have slower rates of convergence and typically use more but cheaper iterations to solve problems, see [28, Section 10.3].

### 7.3.4 Search Direction

In the previous section, the choice for the generalized optimization algorithm is motivated. This algorithm determines the new estimate of the solution according to:

\[ x + \alpha p, \quad (7.43) \]

where \( \alpha > 0 \) and \( f(x + \alpha p) < f(x) \). Obviously, one of the questions is how to determine the search direction \( p \). In order to obtain an improved solution, it is required that the search direction \( p \) is a descent direction. In other words, it must hold that:

\[ p^T \nabla f(x) < 0. \quad (7.44) \]

The subject of this section is to find conditions that guarantee that (7.44) holds for Newton's method. The question how to choose the step length \( \alpha \) is answered in Section 7.3.5.

Basically, the search direction in Newton's method is defined by:

\[ p = -[\nabla^2 f(x)]^{-1} \nabla f(x), \quad (7.45) \]
see (7.39). The search direction \( p \) is required to be a descent direction at the point \( x \), leading to the following condition:

\[
p^T \nabla f(x) = -\nabla f(x)^T [\nabla^2 f(x)]^{-1} \nabla f(x) < 0,
\]

which is equivalent to:

\[
\nabla f(x)^T [\nabla^2 f(x)]^{-1} \nabla f(x) > 0.
\]

This condition is satisfied if \( \nabla^2 f(x) \) is positive definite. Notice that requiring that \( \nabla^2 f(x) \) be positive definite is a stronger condition than \( p^T \nabla f(x) < 0 \).

An important issue is what to do in case the Hessian matrix \( \nabla^2 f(x) \) is indefinite. One possible strategy is to replace the Hessian matrix by some related positive definite matrix in the formula for the search direction. This strategy guarantees that the search direction is a descent direction. Furthermore, it implies that the search direction corresponds to the minimization of a quadratic approximation to the objective function \( f \) that is obtained from the Taylor series by replacing \( \nabla^2 f(x) \) with the related positive definite matrix.

This strategy might seem arbitrary, but it can be shown that the resulting algorithm converges when used inside a line search method, see Section 7.3.5. Furthermore, the Hessian will normally only be replaced at points far away from the solution to the optimization problem. Also, the related positive definite matrix can be found with little additional computation, as discussed in [28, Section 10.4].

7.3.5 Line Search

In order to guarantee convergence, it is necessary to use auxiliary techniques. These techniques are referred to as globalization strategies, in order to distinguish the method used for selecting the new estimate of the solution from the method used for computing the search direction. Essentially, the globalization strategy for choosing the new estimate of the solution is designed to guarantee global convergence. Here, global convergence means convergence from any starting point to a stationary point.

Roughly, two important types of globalization strategy can be distinguished. Line search methods are the most widely used and will be discussed in this section. Trust-region methods are considered in [28, Section 10.6].

Let \( x_k \) be the current estimate of a minimizer of \( f \) and let \( p_k \) be the search direction at the point \( x_k \). The new estimate of the solution is then given by:

\[
x_{k+1} = x_k + \alpha_k p_k,
\]

where the step length \( \alpha_k \) is chosen so that \( f(x_{k+1}) < f(x_k) \). In the previous section, it is stated that the search direction \( p_k \) is chosen to be a descent direction at \( x_k \). If \( p_k \) is a descent direction, then \( f(x_k + \alpha p_k) < f(x_k) \), at least for small positive values of \( \alpha \). For this reason, it is assumed that the step length satisfies \( \alpha_k > 0 \).

The technique is called a line search, because a search for a new point \( x_{k+1} \) is performed along the line \( x_k + \alpha p_k \). Intuitively, it is desired to choose \( \alpha_k \) as the solution to the following problem:

\[
\min_{\alpha > 0} F(\alpha) = f(x_k + \alpha p_k).
\]
That is, $\alpha_k$ is the result of a one-dimensional minimization problem. In practice, an approximate minimizer is found, due to the fact that it is usually too expensive to solve the one-dimensional minimization problem exactly. This approximate minimizer reduces the value of the function $f$, but this appears to be insufficient to guarantee convergence.

Four additional assumptions are necessary to guarantee convergence. The assumptions on the search direction $p_k$ are that (a) it produces sufficient descent and (b) it is gradient related. The assumptions on the step length $\alpha_k$ are that (a) it produces a sufficient decrease in the function $f$ and (b) it is not too small. These assumptions will now be highlighted.

- In Section 7.3.4, it is stated that the search direction must be a descent direction, that is, it must hold that $p_k^T \nabla f(x_k) < 0$. This condition still holds in case $p_k$ becomes arbitrarily close to being orthogonal to $\nabla f(x_k)$, although this implies that the algorithm will make little progress toward a solution. To avoid this, the following condition is introduced:

$$\frac{-p_k^T \nabla f(x_k)}{||p_k|| \||\nabla f(x_k)||} \geq \varepsilon > 0,$$

(7.50)

where $\varepsilon > 0$ is some specified tolerance. If this condition is satisfied for all $k$, the search direction $p_k$ produces sufficient descent.

- The search directions are said to be gradient related if the following condition holds for all $k$:

$$||p_k|| \geq m ||\nabla f(x_k)||,$$

(7.51)

where $m > 0$ is some specified constant. This condition states that the norm of the search direction cannot become too much smaller than that of the gradient.

The sufficient-descent and gradient-relatedness conditions can normally be guaranteed by making slight modifications to the method used to compute the search direction.

- The sufficient decrease condition on $\alpha_k$ ensures that a reduction in the function value is obtained at each iteration. This reduction is in some way arbitrary, as will be shown next. A linear approximation to $f(x_k + \alpha p_k)$ is obtained from the Taylor series:

$$f(x_k + \alpha p_k) \approx f(x_k) + \alpha p_k^T \nabla f(x_k).$$

(7.52)

In the line search, it is required that the step length $\alpha_k$ produces a decrease in the function value that is at least some fraction of the decrease predicted by (7.52). Formally, this leads to the following condition:

$$f(x_k + \alpha_k p_k) \leq f(x_k) + \mu \alpha_k p_k^T \nabla f(x_k),$$

(7.53)

where $\mu$ is some scalar that satisfies $0 < \mu < 1$. The fact that $\mu$ is allowed to take values in a range reflects that the reduction is arbitrary. Essentially, the condition prevents $\alpha$ from being too large, which follows from the illustration in Figure 7.3.

- The last condition is that the step length $\alpha_k$ is required not to be too small. One way to satisfy this condition is to apply a simple line search algorithm that uses backtracking.
This algorithm will be considered here, but is not recommended for practical computations. Another way is to use the so-called Wolfe condition, leading to better but more complicated algorithms. An elaborate discussion on the Wolfe condition can be found in [28, Section 10.5.1].

In the simple line search algorithm, it is assumed that $p_k$ is a search direction that satisfies the sufficient descent condition. According to the backtracking principle, $a_k$ is defined to be the first element of the sequence $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots, 2^{-i}, \ldots$. It is always possible to choose $a_k$ such that the sufficient decrease condition is satisfied. Obviously, because a large step $a = 1$ is tried first and then reduced, the step lengths $a_k$ that are generated by this algorithm will not be too small. Using this algorithm and imposing several additional assumptions, a convergence result can be derived that states that $B_f(x_k) + 0$. Detailed information about this can be found in [28, Section 10.5.1].

7.3.6 Steepest Descent Method

The steepest descent method is based on Newton’s method, but uses a different formula to compute the search direction. In contrast to Newton’s method, it does not require the computation of second derivatives and the storage of matrices. Furthermore, it is not required to solve a system of linear equations in order to compute the search direction. So, in every way it reduces the costs of Newton’s method, at least the costs per iteration. A disadvantage of the steepest descent method is that it has a slower rate of convergence than Newton’s method. It can be shown that it only converges at a linear rate with a constant that is usually close to one, see [28, Section 11.2]. As a result, even though the costs per iteration are low, the overall costs of solving an optimization problem by making use of the steepest descent method are high.

The steepest descent method computes the search direction according to:

$$p_k = -\nabla f(x_k).$$  \hspace{1cm} (7.54)

Subsequently, a line search is performed to determine $x_{k+1} = x_k + a_k p_k$. Hence, the cost of computing the search direction equals the cost of computing the gradient. Because the gradient must be computed to determine if the solution is found, it can be stated that the
search direction is available without extra costs. The search direction is a descent direction if \( \nabla f(x_k) \neq 0 \). In other words, \( p_k \) is a descent direction unless \( x_k \) is a stationary point of the function \( f \).

The formula for the search direction (7.54) can be derived in two ways. The first derivation is based on an approximation to the Hessian matrix. The formula for the steepest descent method is obtained by approximating the Hessian matrix in (7.39) by the identity matrix. This approach, where an approximation to the Hessian is used in the Newton formula, is the basis of the quasi-Newton methods discussed in Section 7.3.7. The second derivation is based on the Taylor series and can be found in [28, Section 11.2].

### 7.3.7 Quasi-Newton Methods

Quasi-Newton methods are the most widely used methods for nonlinear optimization. Although there are many different quasi-Newton methods, they are all based on approximating the Hessian \( \nabla^2 f(x_k) \) by another matrix \( B_k \) that can be obtained at lower cost. Using the matrix \( B_k \), the search direction is obtained by solving:

\[
B_k p = -\nabla f(x_k). \tag{7.55}
\]

Obviously, the search direction is obtained from the Newton equations, with the Hessian replaced by \( B_k \). In case the matrix \( B_k \) is positive definite, this is equivalent to minimizing the following quadratic model:

\[
\min_p \quad f(x_k) + \nabla f(x_k)^T p \quad + \quad \frac{1}{2} p^T B_k p. \tag{7.56}
\]

Different quasi-Newton methods are obtained by different choices of \( B_k \).

There are several advantages to this approach. The approximation \( B_k \), for instance, can be found using only first-derivative information. Furthermore, calculation of the search direction requires less computational effort compared to Newton’s method. Unfortunately, there are also disadvantages. The methods do not converge quadratically, but they can converge superlinearly. Also, quasi-Newton methods still require matrix storage, so other methods are preferred in case large problems have to be solved.

Quasi-Newton methods are generalizations of the so-called secant method, which is used for one-dimensional problems. The secant method is based on the following approximation:

\[
\hat{f}(x_k) \approx \frac{\hat{f}(x_k) - \hat{f}(x_{k-1})}{x_k - x_{k-1}}. \tag{7.57}
\]

Application of (7.57) in Newton’s method leads to:

\[
x_{k+1} = x_k - \frac{(x_k - x_{k-1})}{\hat{f}(x_k) - \hat{f}(x_{k-1})} \hat{f}(x_k). \tag{7.58}
\]

In the multi-dimensional case, however, (7.57) cannot be used, because it would involve division by a vector, which is an undefined operation. For this reason, the condition is rewritten as follows:

\[
\nabla^2 f(x_k)(x_k - x_{k-1}) \approx \nabla f(x_k) - \nabla f(x_{k-1}). \tag{7.59}
\]
Using (7.59), the quasi-Newton approximation $B_k$ is defined by the following condition:

$$B_k(x_k - x_{k-1}) = \nabla f(x_k) - \nabla f(x_{k-1}),$$

(7.60)

which is referred to as the secant condition. For an $n$-dimensional problem, this condition represents a set of $n$ equations that must be satisfied by $B_k$. However, the matrix $B_k$ has $n^2$ entries, which implies that this condition is insufficient to define $B_k$ uniquely. Therefore, additional conditions must be imposed to specify a particular quasi-Newton method.

In order to facilitate the notation, it is useful to introduce the following definitions:

$$s_k = x_{k+1} - x_k$$

(7.61)

$$y_k = \nabla f(x_{k+1}) - \nabla f(x_k).$$

(7.62)

Using these definitions, the secant condition can be rewritten as follows:

$$B_{k+1}s_k = y_k.$$  

(7.63)

When a line search is used, it holds that $x_{k+1} = x_k + \alpha_k p_k$, where $\alpha_k$ is the step length and $p_k$ is the search direction. In this case, it follows that $s_k = \alpha_k p_k$.

A simple example of a quasi-Newton approximation is given by the following update formula:

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}. $$

(7.64)

From this update formula for $B_{k+1}$, a number of general properties of quasi-Newton methods can be derived.

- The secant condition will be satisfied regardless of the choice for the matrix $B_k$. This can easily be verified by substituting (7.64) into (7.63).

- The new approximation $B_{k+1}$ is obtained by updating the old approximation $B_k$. To start a quasi-Newton method, it is necessary to specify an initial approximation $B_0$. Often, $B_0 = I$ is used, but it is preferred to supply a better initial approximation if one can be obtained with little effort.

- The new approximation $B_{k+1}$ can be obtained from $B_k$ using $O(n^2)$ arithmetic operations. This is due to the fact that the difference $B_{k+1} - B_k$ only involves products of vectors of length $n$.

A variety of quasi-Newton methods is obtained by imposing conditions on the approximation $B_k$. Usually, these conditions are properties of the Hessian matrix of which it is desired that they are incorporated in the approximation. An example of such a property is symmetry of the Hessian matrix. From (7.64), it follows that this quasi-Newton formula preserves symmetry. It is called the symmetric rank-one update formula. This name follows from the fact that the update term is a matrix of rank one. It can be shown that this is the only rank-one update formula that preserves symmetry, see [28, Section 11.3].

Using a quasi-Newton method in combination with a line search, see Section 7.3.5, leads to an algorithm of the following form.
Algorithm 3 (Quasi-Newton algorithm).

- Specify some initial guess of the solution $x_0$.
- Specify some initial Hessian approximation $B_0$ (for example $B_0 = I$).
- For $k = 0, 1, \ldots$
  - If $x_k$ is optimal, stop.
  - Solve $B_k p_k = -\nabla f(x_k)$ for the search direction $p_k$.
  - Use a line search to determine the new estimate $x_{k+1} = x_k + \alpha_k p_k$.
  - Compute $s_k = x_{k+1} - x_k$.
  - Compute $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$.
  - Compute $B_{k+1} = B_k + \cdots$ using an update formula.

Another property that can be imposed is that the matrices $B_k$ are required to be positive definite. This condition is reasonable, since the Hessian matrix at the solution $x_*$ will normally be positive definite, or at least positive semi-definite. Furthermore, the positive definiteness condition also guarantees that the quasi-Newton method corresponds to minimizing a quadratic model of the nonlinear function $f$ and that the search direction is a descent direction, see Section 7.3.4.

However, there is no rank-one update formula that maintains both symmetry and positive definiteness. Therefore, a rank-two update formula is used. The rank-two update formula that is considered to be most effective is the Broyden Fletcher Goldfarb Shanno (BFGS) update formula:

$$B_{k+1} = B_k - \frac{(B_k s_k)(B_k s_k)^T}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}.$$  \hfill (7.65)

It can easily be verified that (7.65) satisfies the secant condition (7.63). In order to guarantee positive definiteness of the new matrix $B_{k+1}$, it must hold that $y_k^T s_k > 0$. See [28, Section 11.3] for additional information.
Chapter 8

Self-Tuning Regulator

8.1 Introduction

Subject of this chapter is the self-tuning regulator (STR), which is a so-called adaptive controller. In this case, an adaptive controller is defined to be a controller with adjustable parameters and a mechanism for adjusting the parameters. Consequently, adaptive control is related to Machine-In-the-Loop control optimization. The main reason for using an adaptive controller is that the process and/or its environment is changing continuously. Consequently, the analysis of such systems is not straightforward. In order to simplify the problem, it is assumed that the process has constant but unknown parameters. The term self-tuning is used to express the property that the controller parameters converge to the controller that is designed in case the process is known a priori. An interesting result is that this also happens if the model structure is incorrect.

The purpose of this chapter is to present the basic ideas and to illustrate some properties of self-tuning regulators. It is assumed that the process model and the controller are linear systems. The discussion will also be restricted to discrete time single input single output (SISO) systems. A general description of the self-tuning regulator is given in Section 8.2. In subsequent sections, different parts of the self-tuning regulator will be discussed in more detail.

8.2 General Description

Development of a control system involves many tasks, such as process modeling, design of a control law, implementation and validation. The self-tuning regulator attempts to automate several of these tasks. This is illustrated in Figure 8.1, which shows the block diagram of a process with a self-tuning regulator.

The adaptive controller in Figure 8.1 can be thought of as being composed of two loops. The inner loop consists of the process and an ordinary feedback controller. The parameters of the controller are adjusted by the outer loop. This loop is composed of a parameter estimator and a control design problem. The parameter estimator updates the parameters of the process model. Solution of the control design problem, using the estimated process parameters, leads to an update of the controller parameters. Hence, the system may be viewed as an automation of process modeling and control design, in which the process model and the control design are updated at each sampling period.
In the sequel, it is assumed that the structure of the process model is specified. Examples of model structures can be found in Chapter 5. From the above, it follows that the parameters of the process model are estimated on-line by the block “Estimation” in Figure 8.1. In the sequel, it is assumed that this block is a recursive least squares parameter estimator, which is discussed in [26, Section 11.2]. An overview of different recursive parameter estimation methods can be found in [4, Section 2].

The block “Controller Design” performs computations that are required for the design of a controller according to a specified method, in which a few design parameters can be chosen externally. The design problem is called the underlying design problem in case the process parameters are known exactly, see [7, Section 1.4]. An implementation of the controller, with controller parameters obtained from the control design, is given by the block “Controller” in Figure 8.1.

The tasks shown in the block diagram can be performed in many different ways. Estimation, for example, can be performed continuously or in batches. It is also possible to use hybrid schemes, in which control is performed continuously and the process parameters are updated discretely. Parameter estimation itself can be done in many ways, see [7, Chapter 2]. As already mentioned, recursive least squares will be used throughout the chapter. Notice that in all parameter estimation methods the selection of the input signal is crucial for successful system identification. In this context, the notion of persistent excitation is of importance, see Section 6.5. Furthermore, in adaptive systems, there is an additional complication, because the input signal to the process is generated by feedback. As a result, in certain cases it is not possible to determine the parameters uniquely, which is a highly undesirable situation that can be seen as a disadvantage of the self-tuning regulator. Also with respect to control design, a choice has to be made from a large variety of techniques. An overview of different control design methods can be found in [4, Section 2]. In Section 8.3, deterministic pole placement design is discussed, which is a relatively simple design method. Combination of this design method and least squares estimation gives a self-tuning regulator which clearly shows the basic principles. A different type of self-tuning regulator is obtained when minimum variance (MV) control design is combined with recursive least squares estimation. This type of self-tuning regulator is highlighted in Section 8.4.
The self-tuning regulator scheme in Figure 8.1 is, thus, very flexible with respect to the choice of the design and estimation methods. Many different combinations are possible. A straightforward approach is to estimate the parameters of the process transfer function model. This leads to an indirect algorithm, because the controller parameters are updated indirectly via the estimation of the process parameters. It is sometimes possible to reparameterize the process in such a way that the model can be expressed in terms of the controller parameters. This gives a direct algorithm, because the control design calculations are eliminated and the controller parameters are updated directly.

It should be noted that the self-tuning regulator makes use of the certainty equivalence principle. This follows from the fact that the controller parameters or the process parameters are used as if they are equal to the true parameters. In other words, the uncertainties of the estimates are not considered.

8.3 Pole Placement Design

Pole placement design, a simple control design method, will now be discussed. The basic idea of pole placement design is to determine a controller that gives desired closed-loop poles. In addition, it is required that the closed-loop system responds to command signals in a specified manner. This method clearly shows the principles involved in the design of self-tuning regulators.

It is assumed that the process is described by the following discrete time SISO system:

\[ A(q)y(t) = B(q)(u(t) + v(t)), \quad (8.1) \]

where the disturbances \( v(t) \) enter at the process input. Notice that in case of linear systems, it is always possible to find an equivalent input disturbance, as a result of the superposition principle. Furthermore, \( u(t) \) is the controller output, \( y(t) \) is the process output and \( A(q) \) and \( B(q) \) are polynomials in the forward shift operator \( q \). The polynomials have the degrees \( \deg A = n \) and \( \deg B = \deg A - d_0 \). Here, parameter \( d_0 \) represents the integer part of the ratio of time delay and sampling period. Moreover, it is assumed that \( A \) and \( B \) are relatively prime, which means that they do not have any common factors. Also, \( A \) is assumed to be monic.

Next, the following general linear controller is introduced:

\[ Ru(t) = Tu_c(t) - Sy(t), \quad (8.2) \]

where \( R, S \) and \( T \) are polynomials. This control law is composed of a feedback part, with transfer operator \( \frac{S}{R} \) and a feedforward part, with transfer operator \( \frac{T}{R} \). Hence, it is a two-degree-of-freedom controller. Using (8.1) and (8.2), it is possible to derive the following equations for the closed-loop system:

\[
\begin{align*}
y(t) &= \frac{BT}{AR + BS}u_c(t) + \frac{BR}{AR + BS}v(t) \\
u(t) &= \frac{AT}{AR + BS}u_c(t) - \frac{BS}{AR + BS}v(t).
\end{align*}
\]

Obviously, the closed-loop characteristic polynomial is given by \( A_c = AR + BS \). The key idea of the design method now is to specify the desired closed-loop characteristic polynomial \( A_c \). The polynomials \( R \) and \( S \) can then be solved from \( A_c = AR + BS \), which is called the
Diophantine equation. The equation always has solutions if the polynomials $A$ and $B$ are relatively prime, see [7, Chapter 11].

Recall that (8.2) consists of polynomials $R$, $S$ and $T$, while the Diophantine equation only determines $R$ and $S$. For this reason, the following additional condition is introduced in order to determine the polynomial $T$:

$$ A_m y_m(t) = B_m u_c(t). \quad (8.5) $$

Basically, the polynomials $A_m$ and $B_m$ specify the response $y_m(t)$ to the command signal $u_c(t)$. Using (8.3), it follows that:

$$ \frac{BT}{AR + BS} = \frac{BT}{A_c} = \frac{B_m}{A_m}. \quad (8.6) $$

This model-following condition states that the response of the closed-loop system to command signals is as specified by the model (8.5). The consequences of this condition will be investigated next.

From (8.6), it follows that there are cancelations of factors of $BT$ and $A_c$. Factor the $B$ polynomial as follows:

$$ B = B^+ B^-, \quad (8.7) $$

where $B^+$ is a monic polynomial with stable, well-damped zeros, that can be canceled by the controller and $B^-$ corresponds to unstable and/or weakly damped factors, that cannot be canceled. From (8.6), it follows that $B^-$ must be a factor of $B_m$:

$$ B_m = B^- B'_m. \quad (8.8) $$

Since $B^+$ is canceled, it must be a factor of $A_c$. Moreover, $A_m$ must be a factor of $A_c$ as well, according to (8.6). Therefore, it holds that:

$$ A_c = A_o A_m B^+. \quad (8.9) $$

As a result of $B^+$ being a factor of $B$ and $A_c$, it follows from $A_c = AR + BS$ that $R$ can be factored as follows:

$$ R = R' B^+. \quad (8.10) $$

Consequently, the Diophantine equation can be reduced to:

$$ AR' + B^- S = A_o A_m = A'_c. \quad (8.11) $$

Substitution of (8.7), (8.8) and (8.9) into (8.6) leads to the $T$ polynomial:

$$ T = A_o B'_m. \quad (8.12) $$

In order to obtain a controller that is causal, two additional conditions are required:

$$ \deg S \leq \deg R \quad (8.13) $$

$$ \deg T \leq \deg R. \quad (8.14) $$
In [7, Section 3.2], it is shown that the causality conditions can be written as:

\[
\begin{align*}
\text{deg } A_c & \geq 2\text{deg } A - 1 \\
\text{deg } A_m - \text{deg } B_m & \geq \text{deg } A - \text{deg } B = d_0,
\end{align*}
\]

in case a solution is chosen in which the controller has the lowest possible degree.

Next, it is required that the controller does not introduce extra delay. This implies that polynomials \( R, S \) and \( T \) should have the same degrees. Then, the following control design procedure can be obtained.

**Algorithm 4 (Minimum degree pole placement).**

Data: Polynomials \( A \) and \( B \).
Specifications: Polynomials \( A_m, B_m \) and \( A_0 \).
Compatibility conditions: \( \text{deg } A_m = \text{deg } A \)
\( \text{deg } B_m = \text{deg } B \)
\( \text{deg } A_0 = \text{deg } A - \text{deg } B^+ - 1 \)
\( B_m = B^{-1}B_m \).

**Step 1:** Factor \( B \) as \( B = B^+B^- \), where \( B^+ \) is monic.

**Step 2:** Find the solution \( R' \) and \( S \) with \( \text{deg } S < \text{deg } A \) from \( AR' + B^-S = A_0A_m \). An efficient algorithm that can be used to solve this Diophantine equation can be found in [7, Section 11.4].

**Step 3:** Form \( R = R'B^+ \) and \( T = A_0B'_m \) and compute the control signal from the control law \( Ru = Tu_c - Sy \).

Two special cases of the design procedure are of interest. These are the situation in which all process zeros are canceled and the situation in which no process zeros are canceled, see [7, Section 3.2].

As already mentioned in Section 8.2, there are several different methods available for estimating the parameters of the model given by (8.1). One of these methods, recursive least squares estimation, will now be combined with the minimum degree pole placement method to obtain a simple self-tuning regulator.

**Algorithm 5 (Indirect self-tuning regulator).**

Data: Specifications in the form of a desired closed-loop transfer function \( \frac{B_m(q)}{A_m(q)} \) and a desired observer polynomial \( A_0 \). Additional information concerning the observer polynomial can be found in [7, Section 3.2].

**Step 1:** Estimate the coefficients of the polynomials \( A(q) \) and \( B(q) \) in (8.1), using the recursive least squares method discussed in [7, Section 3.3].

**Step 2:** Apply the minimum degree pole placement method given by Algorithm 4, where the polynomials \( A(q) \) and \( B(q) \) are the estimates obtained in Step 1. The polynomials \( R, S \) and \( T \) of the control law are then obtained.
Step 3: Compute the control signal from (8.2).

Repeat Steps 1, 2 and 3 at each sampling period.

Algorithm 5, based on model-following, is a straightforward application of the concept of self-tuning. The algorithm can be generalized in many different ways, by choosing other recursive parameter estimation methods and other control design techniques.

8.4 Minimum Variance Control

In the previous section, self-tuning controllers are derived that give desired responses to command signals. This section is devoted to a discussion about self-tuners for the regulation problem. The key issue now is to design a controller that reduces disturbances as well as possible. In this context, a steady-state regulation problem is considered, in which the performance criterion is based on the mean square deviations of output and control signal.

It is assumed that the process can be described by a standard ARMAX model:

\[ A(q)y(t) = B(q)u(t) + C(q)e(t), \]

see Section 5.3.3 for more details. The A and C polynomials are normalized such that they both are monic. Moreover, the C polynomial is normalized such that \( \deg C = \deg A = n \). The A and B polynomials may have zeros inside or outside the unit disc. The zeros of the C polynomial are assumed to be inside the unit disc. By spectral factorization, it is always possible to change the polynomial \( C(q) \) in such a way that all its zeros are inside the unit disc or on the unit circle, see [7, Section 4.2].

As already mentioned, the criterion is expressed in terms of the steady-state variances of output and control signal. This leads to the following performance criterion:

\[ J = E\{y^2(t) + \rho u^2(t)\}, \]

where \( E \) denotes mathematical expectation with respect to the noise acting on the system. The control law that minimizes (8.18) is known as the linear quadratic Gaussian (LQG) controller, see [7, Section 4.5]. In case \( \rho = 0 \), the resulting controller is referred to as the minimum variance (MV) controller. It should be noted that (8.18) is defined in discrete time. This implies that only the behaviour at the sampling instances is considered.

In the sequel, it is assumed that \( \rho = 0 \) and, hence, attention is focused on the minimum variance controller. Furthermore, the process is assumed to be minimum phase, which means that the B polynomial has all zeros inside the unit disc. The pole excess of the system is again defined by \( d_0 = \deg A - \deg B \) and equals the time delay in the system. Hence, the input at time \( t \) will influence the output first at time \( t + d_0 \). Consider the following system:

\[ y(t + d_0) = \frac{B}{A} u(t + d_0) + \frac{C}{A} e(t + d_0). \]

Subsequently, \( q^{d_0 - 1}C \) is divided by \( A \):

\[ \frac{q^{d_0 - 1}C(q)}{A(q)} = F(q) + \frac{G(q)}{A(q)}, \]
where the $F$ polynomial of degree $d_0 - 1$ is the quotient and the $G$ polynomial of degree $n - 1$ is the remainder of the division. Relation (8.20) can be interpreted as a Diophantine equation:

$$q^{d_0-1}C(q) = A(q)F(q) + G(q).$$

(8.21)

Using (8.21), it is possible to rewrite (8.19) as follows:

$$y(t + d_0) = \frac{B}{A}u(t + d_0) + Fe(t + 1) + \frac{qG}{A}e(t),$$

(8.22)

where:

$$F(q) = q^{d_0-1} + f_1q^{d_0-2} + \cdots + f_{d_0-1}$$

(8.23)

$$G(q) = g_0q^{n-1} + g_1q^{n-2} + \cdots + g_{n-1}.$$  

(8.24)

By making use of (8.17) and the measurement of $y(t)$ and $u(t)$ it is possible to compute $e(t)$, the innovations. Substitution of the expression for $e(t)$ into (8.22) and performing some simplifications leads to:

$$y(t + d_0) = Fe(t + 1) + \frac{qBF}{C}u(t) + \frac{qG}{C}y(t)$$

(8.25)

$$= Fe(t + 1) + \hat{y}(t + d_0|t).$$

(8.26)

From (8.26), it follows that $y(t + d_0)$ is divided into two parts. The first part, $F(q)e(t + 1)$, depends on the noise acting on the system from $t + 1, \ldots, t + d_0$. The second part, $\hat{y}(t + d_0|t)$, depends on measured outputs ($y(t)$) and applied inputs ($u(t)$). Taking a close look at (8.25), it follows that $\hat{y}(t + d_0|t)$ is the mean square prediction of $y(t + d_0)$ given data up to and including time $t$. The prediction error and its variance are given by:

$$\hat{y}(t + d_0|t) = y(t + d_0) - \hat{y}(t + d_0|t) = F(q)e(t + 1)$$

(8.27)

$$\text{var} \hat{y}(t + d_0|t) = \lambda(1 + f_1^2 + f_2^2 + \cdots + f_{d_0-1}^2).$$

(8.28)

Minimum variance of the output is now obtained by the following control law:

$$u(t) = -\frac{G(q)}{B(q)F(q)}y(t).$$

(8.29)

Using (8.25), $y(t + d_0)$ equals:

$$y(t + d_0) = F(q)e(t + 1)$$

(8.30)

$$= e(t + d_0) + f_1e(t + d_0 - 1) + \cdots + f_{d_0-1}e(t + 1).$$

(8.31)

Consequently, the minimum output variance equals (8.28). From the resulting closed-loop characteristic equation and the assumptions with respect to the polynomials, it can be concluded that all poles are inside the unit disc. Furthermore, it can be observed that minimum variance control is the same as predicting the output $d_0$ steps ahead and then choosing the control signal such that the predicted value is equal to the desired reference value. Extensions to nonminimum phase systems and moving-average control can be found in [7, Section 4.2].

Next, it is possible to obtain a self-tuning regulator by combining a parameter estimation method and the minimum variance controller. Several methods to estimate the parameters in the $A$, $B$ and $C$ polynomials in (8.17) are available, see [7, Chapter 2]. The estimated
parameters are then used in (8.21) to obtain the minimum variance controller. This kind of algorithm has some important properties. In [5, Section 5], two theorems are given which characterize the closed-loop system that is obtained under the assumption that the parameter estimates converge. Additional results concerning stability and convergence of the algorithm can be found in [4, Section 3]. It appears that stability is guaranteed in case of least squares estimation, provided that the process can be described by an ARX model structure and three relatively weak conditions are fulfilled. Convergence of the regulator, however, is less straightforward to prove. Moreover, the available proofs impose several restrictions, for instance with respect to the noise properties. For some algorithms, it even appears to be impossible to prove convergence.
Chapter 9

Model Reference Adaptive Control

9.1 Introduction

This chapter is devoted to model reference adaptive control (MRAC), which is another adaptive control method. In this method, a reference model is used to specify the ideal response of the adaptive control system to an external command signal. The objective is to minimize the tracking error, which is defined as the difference between the output of the adaptive control system and the output of the reference model. In order to achieve this, an adaptation mechanism is used to adjust the parameters in the control law. Consequently, it is obvious that this control method is related to Machine-In-the-Loop control optimization.

A general description of the model reference adaptive control method is given in Section 9.2. It will appear that the major difficulty is to determine the adaptation mechanism in such a way that a stable system, which brings the error to zero, is obtained. Two ways to obtain a parameter adjustment mechanism are considered. In Section 9.3 a gradient method is applied and in Section 9.4 use is made of stability theory. The discussion will be restricted to continuous time systems, because this is the most convenient way to explain both methods. A corresponding analysis for discrete time systems can be found in [25].

9.2 General Description

Generally, a model reference adaptive control system can be schematically represented by the block diagram in Figure 9.1. It is composed of four parts. A process containing unknown parameters, a reference model for specifying the performance of the adaptive control system, a feedback control law containing adjustable parameters and an adaptation mechanism for updating the adjustable parameters.

The controller can be thought of as consisting of two loops. The process and the controller compose the inner loop, which is an ordinary feedback loop. The outer loop adjusts the controller parameters in such a way that the error tends to zero. Here, the error is defined as the difference between the output of the system $y$ and the output of the reference model $y_m$.

The process is assumed to have a known structure, although the parameters are unknown. A reference model is used to specify the ideal response of the adaptive control system to a command signal $u_c$. The choice of the reference model is part of the adaptive control system design and has to satisfy two requirements, see [12]. On the one hand, it should reflect the performance specification in the control tasks, such as rise time, settling time and overshoot.
On the other hand, this ideal behaviour should be achievable for the adaptive control system, which may impose some restrictions. The controller is usually parameterized by a number of adjustable parameters. These controller parameters are updated by an adaptation mechanism. In model reference adaptive control systems, the adaptation mechanism adjusts the parameters in such a way that the response of the adaptive control system becomes the same as that of the reference model. Two ways to obtain an adaptation mechanism are discussed in Section 9.3 and Section 9.4, respectively.

9.3 Gradient Based Approach

The first parameter adjustment mechanism to be discussed is the so-called MIT\(^1\) rule. It is the original approach to model reference adaptive control. For simplicity, consider a closed-loop system in which the controller has only one adjustable parameter \(\theta\). The desired closed-loop response is specified by the output of the reference model \(y_m\). As before, the error \(e\) is the difference between the responses \(y\) and \(y_m\). One possibility is to adjust the controller parameter in such a way that the following loss function is minimized:

\[
J(\theta) = \frac{1}{2}e^2. \tag{9.1}
\]

In order to minimize \(J(\theta)\), it is reasonable to change the parameter in the direction of the negative gradient of \(J(\theta)\):

\[
\frac{d\theta}{dt} = -\gamma \frac{\partial J}{\partial \theta} = -\gamma e \frac{\partial e}{\partial \theta}. \tag{9.2}
\]

This update mechanism is known as the MIT rule. The quantity \(\frac{\partial e}{\partial \theta}\) is the sensitivity derivative of the error with respect to the parameter \(\theta\). If it is assumed that the changes in \(\theta\) are slower than the other variables in the system, the partial derivative \(\frac{\partial e}{\partial \theta}\) may be evaluated under the assumption that \(\theta\) is constant. In order to compute this derivative, it is assumed that the actual system can be represented by the reference model. The parameter \(\gamma\) determines the adaptation rate. In case more than one parameter is adjustable, (9.2) is still valid. Then, \(\theta\) should be interpreted as a vector and \(\frac{\partial e}{\partial \theta}\) as the gradient of the error with respect to the parameters.

\(^1\)The name is derived from the fact that it was developed at MIT.
parameter vector. Actually, the MIT rule can be regarded as a gradient scheme to minimize the squared error \( e^2 \), where the Hessian matrix is chosen equal to the unit matrix.

Recall that the objective in a model reference adaptive control system is to make sure that the error \( e = y - y_m \) tends to zero. This does not necessarily imply that the controller parameters approach their true values, see [7, Section 5.2]. In order to make sure that the parameters converge to their correct values, the input signal must satisfy certain conditions, such as the notion of persistent excitation. More information about this subject can be found in Section 6.5.

An important issue in the design procedure is the choice of the adaptation gain \( \gamma \), which can be crucial, see [7, Section 5.2]. In [7, Section 5.3], several methods are discussed that can be used to determine the adaptation gain. It appears that the stability of the system strongly depends on the signal levels. In this context, the signals \( y_m \) and \( u \) are of interest. For this reason, the algorithm is modified so that it does not depend on the signal levels. This leads to the following modified adjustment rule:

\[
\frac{d\theta}{dt} = \frac{\gamma \varphi e}{\alpha + \varphi^T \varphi},
\]

(9.3)

where \( \varphi = -\frac{\partial \varphi}{\partial \theta} \) and \( \alpha > 0 \). The parameter \( \alpha \) is introduced to avoid difficulties when \( \varphi \) is small. The adaptation rule given by (9.3) is called the normalized MIT rule and results in a substantially improved performance.

By construction, the normalized MIT rule is less sensitive to signal levels. However, the choice of the adaptation gain \( \gamma \) is still critical and very complex behaviour (instability) may be obtained for large adaptation gains. Therefore, a different approach, based on stability theory, is discussed in Section 9.4.

### 9.4 Stability Based Approach

It follows from Section 9.3 that there is no guarantee that an adaptive controller based on the MIT rule gives a stable closed-loop system. In order to design adaptive controllers that can guarantee the stability of the system, it is necessary to have some knowledge of the Lyapunov stability theory. A summary of the Lyapunov stability theory is given in [7, Section 5.4]. An extensive treatment can be found in [31, Chapter 5]. In the sequel, it is assumed that the reader is familiar with Lyapunov's stability theory.

Next, Lyapunov's stability theory is used to derive a stable model reference adaptive controller for a general linear system. During the derivation, the following approach is applied.

1. Find an appropriate controller structure.
2. Derive the error equation and its corresponding differential equation.
3. Find a Lyapunov function and use it to derive a parameter adaptation mechanism such that the error will go to zero.

Consider a general linear system in state space notation:

\[
\frac{dx}{dt} = Ax + Bu.
\]

(9.4)
Assume that the reference model, which gives the desired response to a command signal $u_c$, is given by:

$$\frac{dx_m}{dt} = A_m x_m + B_m u_c. \quad (9.5)$$

A general linear control law for the system (9.4) can then be stated by:

$$u = M u_c - L x. \quad (9.6)$$

Using (9.4) and (9.6), the closed-loop system is obtained:

$$\frac{dx}{dt} = (A - BL)x + BM u_c \quad (9.7)$$

$$= A_c(\theta)x + B_c(\theta)u_c. \quad (9.8)$$

Parameterization of the control law (9.6) can be done in many different ways. All parameters in the matrices $M$ and $L$ are free to choose. In order to capture the general case, it is assumed that the closed-loop system is described by (9.8), where matrices $A_c$ and $B_c$ depend on a parameter $\theta$.

As already mentioned in Section 9.2, it is not always possible to find parameters $\theta$ such that (9.8) is equivalent to (9.5). A sufficient condition is that there exists a parameter value $\theta^0$ such that perfect model-following is achieved:

$$A_c(\theta^0) = A_m \quad (9.9)$$

$$B_c(\theta^0) = B_m. \quad (9.10)$$

In case all parameters in the control law are free to choose, this implies that the following conditions hold:

$$A - A_m = BL \quad (9.11)$$

$$B_m = BM. \quad (9.12)$$

This means that the columns of matrices $A - A_m$ and $B_m$ are linear combinations of the columns of matrix $B$. If these conditions are satisfied and the columns of $B$ and $B_m$ are linearly independent, then the matrices $L$ and $M$ are given by:

$$L = (B^T B)^{-1} B^T (A - A_m) = (B_m^T B_m)^{-1} B_m^T (A - A_m) \quad (9.13)$$

$$M = (B^T B)^{-1} B^T B_m = (B_m^T B_m)^{-1} B_m^T B_m. \quad (9.14)$$

Subsequently, the following error equation is introduced:

$$e = x - x_m. \quad (9.15)$$

Differentiation with respect to time and substitution of (9.4) and (9.5) results in the following equation:

$$\dot{e} = \dot{x} - \dot{x}_m \quad (9.16)$$

$$= Ax + Bu - A_m x_m - B_m u_c. \quad (9.17)$$

It can be seen that the second equality in both equations holds by multiplying the equations to the left by $B$. 
Adding and subtracting $A_m x$ from the right-hand side and assuming that the conditions for perfect model-following are satisfied, makes it possible to rewrite (9.17) as follows:

\[ \dot{e} = A_m e + (A_c(\theta) - A_m)x + (B_c(\theta) - B_m)u_c \]

where the assumption is required for $\theta^0$ to exist. Subsequently, a Lyapunov function is introduced in order to derive a parameter adaptation mechanism:

\[ V(e, \theta) = \frac{1}{2} \left( \gamma e^T P e + (\theta - \theta^0)^T (\theta - \theta^0) \right), \]

where $P$ is a positive definite matrix. From (9.20), it follows that $V(e, \theta)$ is positive definite as well. In order to investigate whether it is a Lyapunov function, its time derivative is computed:

\[ \dot{V}(e, \theta) = -\frac{\gamma}{2} e^T Q e + (\theta - \theta^0)^T (\dot{\theta} + \gamma \Psi^T P e), \]

which holds if $P$ is a positive definite matrix and $A_m$ is stable, see [31, Section 5.7.1]. Next, the parameter adjustment law is chosen as follows:

\[ \dot{\theta} = -\gamma \Psi^T P e. \]

This choice implies that (9.21) simplifies to:

\[ \dot{V}(e, \theta) = -\frac{\gamma}{2} e^T Q e. \]

From (9.24), it can be concluded that $\dot{V}(e, \theta)$, the time derivative of the Lyapunov function, is negative semi-definite. By making use of Barbalat’s lemma, it is possible to show that the error $e$ tends to zero, see [7, Section 5.4]. Here, Barbalat’s lemma is given as follows.

**Lemma 1 (Barbalat’s lemma).** If $g$ is a real function of a real variable $t$, defined and uniformly continuous for $t \geq 0$ and if the limit of the integral:

\[ \int_0^t g(s) \, ds \]

as $t$ tends to infinity exists and is a finite number, then:

\[ \lim_{t \to \infty} g(t) = 0. \]

See [7, Section 5.4].

However, the parameters will not necessarily converge to their correct values. To have parameter convergence, it is necessary to impose conditions on the excitation of the system, see Section 6.5. Several examples and extensions with respect to the construction of parameter adjustment rules based on Lyapunov’s stability theory can be found in [7, Section 5.5]. In [12], a detailed description is given with respect to the derivation of a model reference adaptive control system for a general second-order system, based on Lyapunov theory.
Chapter 10
Iterative Feedback Tuning

10.1 Introduction

Many control objectives can be expressed in terms of a criterion function. In this context, one can think of the model reference adaptive control concept, see Section 9.3. Generally, it can be concluded that full knowledge of process and disturbances and complete freedom in the complexity of the controller is required in order to explicitly solve such an optimization problem. However, the process and the disturbances are usually unknown in practice. Moreover, it is often desirable to achieve the best possible performance with a controller of limited complexity, such as a simple PID controller.

The optimization of such a criterion function typically requires iterative gradient based minimization procedures. The major difficulty in solving the optimization problem is given by the computation of the gradient of the criterion function with respect to the controller parameters. Generally, the gradient appears to be a rather complicated function of the process and the disturbance dynamics. Consequently, in case these dynamics are unknown, the gradient cannot be computed in a straightforward way.

Obviously, it is possible to compute the gradient of the criterion function with respect to the controller parameters if a process and a disturbance model are available. However, if the models are not accurate, the performance of the controller is usually poor. Due to the fact that it is often time-consuming and expensive to obtain accurate models in practice, it is desired to search for a method in which no models are required.

In [22], it is shown that an unbiased estimate of the gradient can be computed from signals obtained from closed loop experiments with the most recent controller operating on the actual system. Minimization of the control performance criterion can then be performed iteratively by a Gauss-Newton based scheme. For a one-degree-of-freedom controller, two batch experiments are to be performed at each step of the iterative design. The first experiment consists of collecting a batch of data under normal operating conditions. The second experiment is a special experiment, in which the error measured during normal operation is fed back at the process input. Therefore, this scheme is referred to as iterative feedback tuning (IFT). In case of a two-degree-of-freedom controller, one additional experiment under normal operating conditions is required.

This chapter is devoted to a basic description of the iterative feedback tuning concept. During the derivation, attention is restricted to discrete time single input single output linear time-invariant systems. Extensions to multi input multi output (MIMO) systems can be
found in [20, Section 4]. The IFT concept is discussed for a one-degree-of-freedom and a two-degree-of-freedom control structure in Sections 10.2 and 10.3, respectively. Implementation issues and design choices are considered in Section 10.4.

10.2 One-Degree-of-Freedom Control

10.2.1 Control Design Criterion

Consider the discrete time closed loop system depicted in Figure 10.1. It can easily be verified that this system is described by:

\[
\begin{align*}
    y(t) &= P_0(q)u(t) + v(t) \\
    u(t) &= C(q, \rho)(r(t) - y(t)).
\end{align*}
\]

Here, \( P_0(q) \) is a linear time-invariant operator and \( q \) denotes the forward shift operator, see Section 3.4. Furthermore, \( u(t) \in \mathbb{R} \) represents the process input and \( y(t) \in \mathbb{R} \) represents the corresponding process output. The unmeasurable disturbance, located at the output, is denoted by \( v(t) \in \mathbb{R} \) and is assumed to be zero mean and weakly stationary, see Section 2.3.3. An external deterministic reference signal is represented by \( r(t) \in \mathbb{R} \). The system is assumed to be controlled by the one-degree-of-freedom controller \( C(q, \rho) \). Obviously, this linear time-invariant transfer function is parameterized by the parameter vector \( \rho \in \mathbb{R}^{np} \). Recall that \( u(t, \rho) \) and \( y(t, \rho) \) denote signals that are obtained from the closed loop system.

![Figure 10.1: Closed loop system with one-degree-of-freedom control.](image)

Next, a desired output response to the deterministic reference signal \( r(t) \) will be given. One possibility is to define the response of the closed loop system as the output of a reference model \( T_d(q) \), according to:

\[
y^d(t) = T_d(q)r(t).
\]

Subsequently, the error between the achieved and the desired response of the system is defined by:

\[
\bar{y}(t, \rho) = y(t, \rho) - y^d(t).
\]

It is natural to formulate the control design objective as a minimization of some objective function of \( \bar{y}(t, \rho) \). For simplicity, the following quadratic objective function will be used:

\[
J(\rho) = \frac{1}{2N} E \left[ \sum_{t=1}^{N} \bar{y}^2(t, \rho) \right],
\]
where $E[\cdot]$ denotes expectation with respect to the disturbance $v(t)$. Of course, other objective functions can be used, for example by using the absolute error instead of the quadratic error. See [20, Section 7.1] for some additional suggestions.

The objective of the iterative feedback tuning algorithm is to find the optimal parameter vector $\rho$ for the controller that minimizes the objective function (10.5). Formally, this can be stated by:

$$\rho^* = \arg \min_{\rho} J(\rho).$$  (10.6)

In the case where use is made of a reference model $y_d(t) = T_d(q)r(t)$, the problem definition is closely connected to model reference adaptive control, see Chapter 9. Model reference adaptive control is based on the minimization of a criterion of the same type as (10.5) with respect to the controller parameters. As will be seen in the next section, the gradient that is required to solve the optimization problem depends on the transfer function of the unknown closed loop system. Essentially, the MRAC solution to this minimization problem is to replace the true closed loop system by the reference model in the gradient computation. The IFT approach, in contrast to the MRAC approach, computes the gradient by only making use of input-output data obtained from the actual closed loop system. Hence, no approximations are required to generate the gradient.

10.2.2 Criterion Minimization

A necessary condition for the optimal controller tuning $\rho^*$ is that the first derivative of the objective function with respect to the controller parameter vector $\rho$ is zero:

$$\frac{\partial J(\rho)}{\partial \rho} = \frac{1}{N} E \left[ \sum_{t=1}^{N} \tilde{y}(t, \rho) \frac{\partial \tilde{y}(t, \rho)}{\partial \rho} \right] = \frac{1}{N} E \left[ \sum_{t=1}^{N} \tilde{y}(t, \rho) \frac{\partial y(t, \rho)}{\partial \rho} \right] = 0,$$  (10.7)

where use is made of the fact that the desired output $y_d(t)$ is independent of the controller parameters. In order to achieve optimal performance, it is necessary to be able to detect this condition for arbitrary controllers. Hence, approximations of the following quantities are required.

- The error signal $\tilde{y}(t, \rho)$, see (10.4).
- The gradient $\frac{\partial y(t, \rho)}{\partial \rho}$.

If an unbiased estimate of the gradient $\frac{\partial J(\rho)}{\partial \rho}$ is available, the solution of (10.7) can be found by using the following iterative algorithm:

$$\rho_{i+1} = \rho_i - \gamma_i R_i^{-1} \text{est} \left[ \frac{\partial J(\rho_i)}{\partial \rho} \right].$$  (10.8)

Here, $\gamma_i$ is a sequence of positive real numbers that determines the step size and $R_i$ is a sequence of appropriate positive definite matrices. The choice of both parameters will be discussed in Section 10.4.3. Notice that the "est" operator denotes an approximation of the gradient.
In (10.8), use is made of an iterative approach, because an explicit solution to the optimization problem requires full knowledge of the process and the disturbances, as well as complete freedom in the complexity of the controller. This is where the iterative feedback tuning algorithm differs from, for instance, $H_2$ and $H_\infty$ control.

As stated above, the signal $\dot{y}(t, \rho)$ and the gradient $\partial y(t, \rho) / \partial \rho$ are required to compute an estimate of the partial derivative of the objective function $J(\rho)$ with respect to the parameter vector $\rho$. The signal $\dot{y}(t, \rho)$ can be obtained by conducting an experiment using the closed loop system in Figure 10.1. Substitution of the measured output into (10.4) results in the desired signal. The gradient of the process output with respect to the controller parameters is more difficult to obtain. This can be seen by differentiating (10.1) and (10.2) with respect to an arbitrary element of $\rho$:

$$\frac{\partial y(t, \rho)}{\partial \rho_j} = P_0(q) \frac{\partial u(t, \rho)}{\partial \rho_j}$$

(10.9)

$$\frac{\partial u(t, \rho)}{\partial \rho_j} = \frac{\partial C(q, \rho)}{\partial \rho_j} (r(t) - y(t, \rho)) - C(q, \rho) \frac{\partial y(t, \rho)}{\partial \rho_j}.$$  

(10.10)

It is possible to express (10.9) and (10.10) in a block diagram, see Figure 10.2.

![Figure 10.2: Closed loop system corresponding to gradient (10.9) and (10.10).](image)

Using this figure, the derivative of $y(t, \rho)$ with respect to the full parameter vector $\rho$ can be expressed by:

$$\frac{\partial y(t, \rho)}{\partial \rho} = P_0(q) S_0(q, \rho) \frac{\partial C(q, \rho)}{\partial \rho} (r(t) - y(t, \rho)),$$

(10.11)

where the sensitivity function $S_0(q, \rho)$, achieved with the controller $C(q, \rho)$, is given by:

$$S_0(q, \rho) = \frac{1}{1 + P_0(q)C(q, \rho)}.$$  

(10.12)

Furthermore, the gradient of the controller $C(q, \rho)$ with respect to the parameter vector $\rho$ is given by:

$$\frac{\partial C(q, \rho)}{\partial \rho} = \left[ \frac{\partial C(q, \rho)}{\partial \rho_1} \frac{\partial C(q, \rho)}{\partial \rho_2} \cdots \frac{\partial C(q, \rho)}{\partial \rho_p} \right]^T.$$  

(10.13)

Taking a close look at (10.11) reveals that it depends on the unknown process $P_0(q)$. However, using the configuration in Figure 10.2, it is possible to apply the signal $r(t) - y(t, \rho)$ of a previous experiment, filtered by $\partial C(q, \rho) / \partial \rho_j$, at the process input of the actual process.
this way, it is possible to obtain an estimate for \( \partial y(t, \rho)/\partial \rho_j \) by execution of one additional experiment. However, in order to obtain the derivative with respect to all parameters, this additional experiment needs to be performed \( n_p \) times. Fortunately, under the assumption that \( P_0(q) \) is a single input single output linear time-invariant system, Figure 10.2 can be transformed into Figure 10.3.

![Figure 10.3: Closed loop system corresponding to gradient in case of a single input single output linear time-invariant system.](image)

There are several alternatives with respect to the setup shown in Figure 10.3. An overview of the different setups can be found in [20, Section 3.5]. The main advantage of Figure 10.3 compared to Figure 10.2 is that an estimate of \( \partial y(t, \rho)/\partial \rho \) can be obtained by performing only one special experiment instead of \( n_p \) special experiments. Due to the fact that the real process introduces an unmeasurable disturbance, it is only possible to obtain an estimate of the gradient.

The above leads to the following basic iterative feedback tuning algorithm for a one-degree-of-freedom controller. Notice that an initial stabilizing controller is required for the algorithm to function properly.

Algorithm 6 (IFT algorithm for a one-degree-of-freedom controller).

1. Perform an experiment using the closed loop system from Figure 10.1 with controller \( C(q, \rho) \) and reference signal \( r(t) \). Collect \( N \) measurements of the output \( y(t, \rho) \) given by:

   \[
   y(t, \rho) = T_0(q, \rho) r(t) + S_0(q, \rho) v(t), \tag{10.14}
   \]

   where:

   \[
   T_0(q, \rho) = \frac{P_0(q) C(q, \rho)}{1 + P_0(q) C(q, \rho)}. \tag{10.15}
   \]

   This experiment is referred to as the normal experiment, because the experiment is performed under normal operating conditions. Furthermore, the superscript 1 will be used to denote the first experiment, while the subscript \( i \) will be used to indicate the iteration number, see (10.8). Using (10.14) it is possible to compute \( \hat{y}_1(t, \rho_i) \) in an exact way according to (10.4).

2. Perform a second experiment using the setup from Figure 10.3. This experiment is referred to as the gradient experiment and will be indicated by superscript 2. The error signal \( r(t) - y_1^1(t, \rho_i) \) is applied to the process input, leading to the following output:

   \[
   y_2^2(t, \rho_i) = \frac{1}{C(q, \rho_i)} T_0(q, \rho_i) (r(t) - y_1^1(t, \rho_i)) + S_0(q, \rho_i) v_2^2(t). \tag{10.16}
   \]
3. Take as gradient approximation:

\[
\text{est} \left[ \frac{\partial \tilde{y}(t, \rho_i)}{\partial \rho} \right] = \text{est} \left[ \frac{\partial J(q, \rho_i)}{\partial \rho} \tilde{y}(t, \rho_i) \right],
\]

see Figure 10.3.

4. Next, it is possible to compute an estimate of the derivative of the objective function \( J(\rho_i) \) with respect to the parameter vector \( \rho_i \):

\[
\text{est} \left[ \frac{\partial J(\rho_i)}{\partial \rho} \right] = \frac{1}{N} \sum_{t=1}^{N} \tilde{y}(t, \rho_i) \text{est} \left[ \frac{\partial y(t, \rho_i)}{\partial \rho} \right].
\]

Obviously, in this case the estimate is computed by using experimental data only.

5. Update the controller parameter vector \( \rho_i \) according to (10.8).

6. Repeat the sequence, starting from Step 1, until (10.7) is fulfilled to a sufficient extent.

10.3 Two-Degree-of-Freedom Control

In Section 10.2, the iterative feedback tuning algorithm for a one-degree-of-freedom controller is discussed. It is possible to extend the iterative feedback tuning algorithm in such a way that it can handle a two-degree-of-freedom controller. Because this extension may be useful with respect to feedforward control design, it will be discussed in this section. Moreover, use will be made of a more general control design criterion in comparison with the one used in Section 10.2.

10.3.1 Control Design Criterion

The unknown system under consideration is the same as in the previous section. For the sake of completeness, it is given once again:

\[
y(t) = p_0(q)u(t) + v(t).
\]

In this case, however, the system is controlled by the following two-degree-of-freedom controller:

\[
u(t) = C_r(q, \rho_r) r(t) - C_y(q, \rho_y) y(t),
\]

where \( C_r(q, \rho_r) \) and \( C_y(q, \rho_y) \) are linear time-invariant transfer functions. The controller parameters are denoted by \( \rho_r \in \mathbb{R}^{n_r} \) and \( \rho_y \in \mathbb{R}^{n_y} \). In the sequel, the controller parameters will be collected in one vector of controller parameters \( \rho \in \mathbb{R}^{n_r+n_y} \), with \( \rho = [\rho_r^T, \rho_y^T]^T \). Recall that in case \( C_r(q, \rho) = C_y(q, \rho) \), a one-degree-of-freedom controller is obtained. A block diagram of the closed loop system is depicted in Figure 10.4.

Now, the control design objective will be introduced. In order to improve the flexibility of the design, various design parameters are included in the objective function:

\[
J(\rho) = \frac{1}{2N} \sum_{t=1}^{N} (L_y(q)\tilde{y}(t, \rho))^2 + \lambda \sum_{t=1}^{N} (L_u(q)u(t, \rho))^2.
\]
10.3 Two-Degree-of-Freedom Control

From (10.21), it follows that the control design criterion is again quadratic. Furthermore, it is possible to penalize both the error $\hat{y}(t, \rho)$ and the control effort $u(t, \rho)$. The parameter $\lambda$ can be used to change the penalty on the control effort. Moreover, the error and the control effort can be frequency weighted by the filters $L_y(q)$ and $L_u(q)$, respectively. These parameters give added flexibility to the design.

The objective of the iterative feedback tuning algorithm is the same as in Section 10.2. The algorithm attempts to minimize the objective function (10.21) in order to find the optimal parameter vector $\rho$ for the controller:

$$\rho^* = \arg \min_{\rho} J(\rho).$$

**10.3.2 Criterion Minimization**

In this section, the minimization of the objective function $J(\rho)$ with respect to the controller parameter vector $\rho$ is addressed. In order to obtain the minimum of $J(\rho)$ and, consequently, the optimal parameter vector, it is necessary that the first derivative of the objective function with respect to $\rho$ equals zero:

$$\frac{\partial J(\rho)}{\partial \rho} = \frac{1}{N} E \left[ \sum_{t=1}^{N} (L_y(q)\hat{y}(t, \rho)) \left( L_y(q) \frac{\partial \hat{y}(t, \rho)}{\partial \rho} \right) + \lambda \sum_{t=1}^{N} (L_u(q)u(t, \rho)) \left( L_u(q) \frac{\partial u(t, \rho)}{\partial \rho} \right) \right] = 0.$$

In case the gradient $\frac{\partial J(\rho)}{\partial \rho}$ can be computed, the solution of (10.23) can be obtained by application of the following iterative algorithm:

$$\rho_{i+1} = \rho_i - \gamma_i R_i^{-1} \text{est} \left[ \frac{\partial J(\rho)}{\partial \rho} \right].$$

This algorithm is identical to the one used in Section 10.2.

It also follows from this section that the error $\hat{y}(t, \rho)$ can be obtained in a straightforward way, using (10.4). This holds for the signal $u(t, \rho)$ as well, because it can be measured. The gradients of $y(t, \rho)$ and $u(t, \rho)$, however, are more difficult to compute. Therefore, it is shown in this section in which way it is possible to obtain estimates of these gradients.
To this end, two transfer functions are introduced. These are the functions $T_0(q, p)$ and $S_0(q, p)$, denoting the achieved closed loop response and sensitivity function with the controller \{$C_r(q, p), C_y(q, p)$\}, respectively. They are defined according to:

\[
T_0(q, p) = \frac{C_r(q, p)P_0(q)}{1 + C_y(q, p)P_0(q)} \tag{10.25}
\]

\[
S_0(q, p) = \frac{1}{1 + C_y(q, p)P_0(q)} \tag{10.26}
\]

In Appendix A.1, the following expression for $\frac{\partial y(t, p)}{\partial p}$ is derived:

\[
\frac{\partial y(t, p)}{\partial p} = \frac{1}{C_r(q, p)} \left[ \left( \frac{\partial C_r(q, p)}{\partial p} - \frac{\partial C_y(q, p)}{\partial p} \right) T_0(q, p) r(t) + \frac{\partial C_y(q, p)}{\partial p} T_0(q, p) (r(t) - y(t, p)) \right]. \tag{10.27}
\]

Similarly, in Appendix A.2, an expression for $\frac{\partial u(t, p)}{\partial p}$ is derived:

\[
\frac{\partial u(t, p)}{\partial p} = S_0(q, p) \left[ \left( \frac{\partial C_r(q, p)}{\partial p} - \frac{\partial C_y(q, p)}{\partial p} \right) r(t) + \frac{\partial C_y(q, p)}{\partial p} (r(t) - y(t, p)) \right]. \tag{10.28}
\]

In contrast to the one-degree-of-freedom controller in Section 10.2, it is not straightforward to express (10.27) and (10.28) in a block diagram. For this reason, three experiments will be posed that are performed during each iteration $i$ of the controller tuning algorithm. By making use of the results of these experiments, it is possible to derive estimates for the gradients.

Each experiment is of length $N$, with the fixed controller $C(q, \rho) = \{C_r(q, \rho), C_y(q, \rho)\}$ operating on the actual process. The first and third experiment are identical and consist of collecting measurement data under normal operating conditions. The second experiment is a special experiment. In the following algorithm, the reference signal, the process input and the process output of all three experiments are indicated.

Algorithm 7 (IFT algorithm for a two-degree-of-freedom controller).

1. Experiment 1.

Reference signal: $r_1^i(t) = r(t)$
Process input: $u_1^i(t, \rho_i) = S_0(q, \rho_i)(C_r(q, \rho_i)r(t) - C_y(q, \rho_i)v_1^i(t))$
Process output: $y_1^i(t, \rho_i) = T_0(q, \rho_i)r(t) + S_0(q, \rho_i)v_1^i(t)$

2. Experiment 2.
10.3 Two-Degree-of-Freedom Control

Reference signal: $\tilde{r}^2(t) = r(t) - y_1^4(t, \rho_i)$

Process input: $u_1^2(t, \rho) = S_0(q, \rho_i) C_r(q, \rho_i)(r(t) - y_1^4(t, \rho_i)) - C_y(q, \rho_i) v_1^2(t)$

Process output: $y_1^2(t, \rho) = T_0(q, \rho_i) (r(t) - y_1^4(t, \rho_i)) + S_0(q, \rho_i) v_1^2(t)$

3. Experiment 3.

Reference signal: $\tilde{r}^2(t) = r(t)$

Process input: $u_1^3(t, \rho) = S_0(q, \rho_i) C_r(q, \rho_i) r(t) - C_y(q, \rho_i) v_1^3(t)$

Process output: $y_1^3(t, \rho) = T_0(q, \rho_i) r(t) + S_0(q, \rho_i) v_1^3(t)$

The first experiment is used for the computation of $\tilde{y}(t, \rho_i)$ and to retrieve $u(t, \rho_i)$. The second and third experiment are utilized to estimate the gradient of $y(t, \rho_i)$ according to:

$$
\text{est} \left[ \frac{\partial y(t, \rho_i)}{\partial \rho} \right] = \frac{1}{C_r(q, \rho_i)} \left[ \left( \frac{\partial C_r(q, \rho_i)}{\partial \rho} - \frac{\partial C_y(q, \rho_i)}{\partial \rho} \right) y_1^2(t, \rho_i) + \right.
\left. \frac{\partial C_y(q, \rho_i)}{\partial \rho} y_1^2(t, \rho_i) \right]. \quad (10.29)
$$

This estimate is a perturbed version of (10.27) due to the disturbances $v_1^2(t)$ and $v_1^3(t)$. Indeed, by comparing (10.29) with (10.27), using the signals from Algorithm 7, it is seen that:

$$
\text{est} \left[ \frac{\partial y(t, \rho_i)}{\partial \rho} \right] = \frac{\partial y(t, \rho_i)}{\partial \rho} + \frac{S_0(q, \rho_i)}{C_r(q, \rho_i)} \left[ \left( \frac{\partial C_r(q, \rho_i)}{\partial \rho} - \frac{\partial C_y(q, \rho_i)}{\partial \rho} \right) v_1^3(t) + \right.
\left. \frac{\partial C_y(q, \rho_i)}{\partial \rho} v_1^2(t) \right]. \quad (10.30)
$$

Similarly, it is possible to use the second and third experiment to estimate the gradient of $u(t, \rho_i)$ as follows:

$$
\text{est} \left[ \frac{\partial u(t, \rho_i)}{\partial \rho} \right] = \frac{1}{C_r(q, \rho_i)} \left[ \left( \frac{\partial C_r(q, \rho_i)}{\partial \rho} - \frac{\partial C_y(q, \rho_i)}{\partial \rho} \right) u_1^2(t, \rho_i) + \right.
\left. \frac{\partial C_y(q, \rho_i)}{\partial \rho} u_1^2(t, \rho_i) \right], \quad (10.31)
$$

which is a perturbed version of (10.28). This can easily be seen by comparing (10.31) with (10.28):

$$
\text{est} \left[ \frac{\partial u(t, \rho_i)}{\partial \rho} \right] = \frac{\partial u(t, \rho_i)}{\partial \rho} - \frac{C_y(q, \rho_i) S_0(q, \rho_i)}{C_r(q, \rho_i)} \left[ \frac{\partial C_y(q, \rho_i)}{\partial \rho} v_1^2(t) + \right.
\left. \left( \frac{\partial C_r(q, \rho_i)}{\partial \rho} - \frac{\partial C_y(q, \rho_i)}{\partial \rho} \right) v_1^3(t) \right]. \quad (10.32)
$$
Hence, both gradient estimates are perturbed by the disturbances \( v_2^1(t) \) and \( v_3^1(t) \). Next, it is possible to construct an experimentally based estimate of the gradient \( J(p_i) \) with respect to the controller parameter vector \( \rho_i \):

\[
\text{est} \left[ \frac{\partial J(p_i)}{\partial \rho} \right] = \frac{1}{N} \sum_{t=1}^{N} \left[ (L_y(q)\hat{y}(t, \rho_i)) \left( L_y(q) \text{est} \left[ \frac{\partial y(t, \rho_i)}{\partial \rho} \right] \right) + \lambda L_u(q)u(t, \rho_i) \right] \cdot (10.33)
\]

For the algorithm to work properly, it is required that this estimate is unbiased. This requirement can be expressed as follows:

\[
E \left\{ \text{est} \left[ \frac{\partial J(p_i)}{\partial \rho} \right] \right\} = \frac{\partial J(p_i)}{\partial \rho} \cdot (10.34)
\]

It can be concluded that this unbiasedness property holds due to the fact that the third experiment is performed. In case the data from the first experiment instead of the third experiment would be used in (10.29) and (10.31), (10.34) would not hold. This is due to the fact that the errors in est\([\partial y(t, \rho_i)/\partial \rho]\) and est\([\partial u(t, \rho_i)/\partial \rho]\) would be correlated with est\([\hat{y}(t, \rho_i)]\) and est\([u(t, \rho_i)]\), respectively. A proof of the unbiasedness property can be found in [29, Section 2.3].

By making use of the gradient estimate (10.33), it is possible to update the controller parameter vector by applying (10.24).

### 10.4 Implementation Issues and Design Choices

In the previous two sections, iterative feedback tuning algorithms are derived for a one-degree-of-freedom control structure and a two-degree-of-freedom control structure, respectively. Several important issues regarding both algorithms will be discussed in this section.

#### 10.4.1 Convergence

Exact conditions for which the controller parameters, updated according to the IFT algorithm, converge to the set of stationary points of the design criterion are given in this section.

Let \( \mathcal{D} \) be a compact subset of \( \mathbb{R}^{n_\rho} \). Subsequently, introduce the following conditions.

- There exists a neighbourhood \( \mathcal{O} \) to \( \mathcal{D} \) such that the controller \( C(q, \rho) \) is two times continuously differentiable with respect to \( \rho \) in \( \mathcal{O} \).

- All elements of the transfer functions \( C_y(q, \rho) \), \( \partial C_y(q, \rho)/\partial \rho \), \( \partial^2 C_y(q, \rho)/\partial \rho^2 \) and \( \partial^2 C_y(q, \rho)/\partial \rho^2 \) have their poles and zeros uniformly bounded away from the unit circle on \( \mathcal{D} \).

- The linear time-invariant closed loop systems are stable and have all their poles uniformly bounded away from the unit circle on \( \mathcal{D} \).
10.4 Implementation Issues and Design Choices

- In any experiment, the disturbance signal \( v(t) \) is a bounded discrete time stochastic process with zero mean. The second-order statistics of \( v(t) \) are the same for all experiments, but need not be stationary within one experiment. Disturbance sequences from different experiments are mutually independent.

- The elements of the sequence \( \gamma_i \) satisfy \( \gamma_i \geq 0, \sum_{i=1}^{\infty} \gamma_i = \infty \) and \( \sum_{i=1}^{\infty} \gamma_i^2 < \infty \).

**Theorem 4 (Convergence).** Consider the iterative algorithm (10.24). Assume that the conditions above are satisfied. Suppose that \( R_i \) satisfies \( \frac{1}{\delta} \leq R_i \geq \delta I \) for some \( \delta > 0 \). Then:

\[
\lim_{i \to \infty} \rho_i = D_c \triangleq \left\{ \rho : \frac{\partial J(\rho)}{\partial \rho} = 0 \right\} \text{ w. p. 1} \tag{10.35}
\]

on the set \( A = \{ \rho_i \in D \forall i \} \). See [21].

The most important requirement for convergence is that the signals remain bounded throughout the iterations. Furthermore, it can be remarked that the only assumption is that the system is linear time-invariant. The controller is allowed to be of arbitrary complexity. In practice, the conditions on the disturbance signal \( v(t) \) are often not satisfied. Consequently, the convergence proof is of limited use.

10.4.2 Nonminimum Phase Controllers

From (10.29) and (10.31), it follows that the computation of the concerning gradient estimates requires filtering with the inverse of the controller \( C_r(q, \rho) \). If \( C_r(q, \rho) \) is nonminimum phase, as may especially happen during the first iterations, its inverse is unstable and the gradient estimates are not feasible.

In general, at least for low-order controllers, it is noticed that the optimal controller is minimum phase. This raises the question why the controller may become nonminimum phase. One reason is that a zero of the controller might cancel the fixed integrator. It is standard to include a fixed integrator in the controller in order to suppress low frequency disturbances and to ensure correct static gain. Due to finite data and disturbances, it may happen that this zero is just outside the unit circle, leading to a nonminimum phase controller. After several additional iterations, however, the controller usually becomes minimum phase again, because the fixed integrator is important in the controller design.

There are several solutions available to avoid the controller from becoming nonminimum phase. Three different approaches are discussed in [29, Section 3.3]. Also in [21], a discussion on the problem of nonminimum phase controllers can be found, together with several solutions.

10.4.3 Design Choices Iterative Algorithm

From Section 10.3, it follows that the optimal controller tuning \( \rho^* \) is obtained by using an iterative Gauss-Newton based scheme:

\[
\rho_{i+1} = \rho_i - \gamma_i R_i^{-1} \text{est} \left[ \frac{\partial J(\rho_i)}{\partial \rho} \right]. \tag{10.36}
\]

Here, \( i \) denotes the iteration number, \( \gamma_i \) is a positive real scalar that determines the step size and \( R_i \) is a positive definite matrix. Hence, two design parameters are available, the choice of which will be discussed in this section.
With respect to the positive definite matrix $R_t$, several design choices are possible. One possibility is to choose $R_t$ equal to the identity matrix, leading to an update in the negative gradient direction. The resulting scheme is known as the steepest descent method, see Section 7.3.6. This method is exact for objective functions that are linear in the controller parameters, which is generally not the case here. As a result, the step size $\gamma_t$ is restricted to relatively small values. For this reason, an alternative choice for $R_t$ is desired. The alternative is given by an approximation of the Hessian of $J(\rho)$ with respect to the parameter vector $\rho$, leading to a so-called quasi-Newton method, see Section 7.3.7. This method is exact for objective functions that are quadratic in the controller parameters. Indeed, this implies that the minimum of a quadratic objective function can be found in one iteration, in case the step size is chosen equal to one. Unfortunately, the objective function used in the iterative feedback tuning algorithm is not quadratic in the controller parameters. Nevertheless, the quasi-Newton method is known to have better convergence properties compared to the steepest descent method. The main disadvantage of the quasi-Newton method is the effort that is required to compute the Hessian. One possibility to approximate the Hessian is given by:

$$R_t = \frac{1}{N} \sum_{i=1}^{N} \left( \text{est} \left[ \frac{\partial y(t, \rho_i)}{\partial \rho} \right] \text{est} \left[ \frac{\partial y(t, \rho_i)}{\partial \rho} \right]^T + \lambda \text{est} \left[ \frac{\partial u(t, \rho_i)}{\partial \rho} \right] \text{est} \left[ \frac{\partial u(t, \rho_i)}{\partial \rho} \right]^T \right), \quad (10.37)$$

which is proposed in [21] and used in [23, Section 7.3.1]. A major advantage of this approximation is that the required signals are already available from the regular experiments. However, due to the fact that the gradient estimates are perturbed by the disturbances during the second and third experiment, the estimate of the Gauss-Newton direction will be biased. Several other possibilities to approximate the Hessian are given in [29, Section 3.4] and [20, Section 3.6].

The step size $\gamma_t$ can be used to control how much a controller changes from one iteration to another. In order to verify whether the choice for the step size is acceptable, it is useful to compare the Bode plots of the new controller and the previous controller. If there is a significant difference between both Bode plots, it is recommended to reduce the step size in order to avoid instability. Notice that this check is difficult to automate in practice. It is also possible to increase the length of the experiment in order to reduce the effects of the disturbances in the gradient computation. In this way, a more reliable search direction is obtained. Generally, it can be concluded that for a small enough step size and a large enough data set one will always go in a descent direction of the criterion. The step size can also be optimized along the gradient direction by line search optimization. However, in this case a destabilizing controller may be obtained.

### 10.4.4 Design Parameters

It is important to realize that the properties of the resulting system depend entirely on the control design criterion. Consequently, the sensitivity function, the complementary sensitivity function and the stability margins, for instance, cannot be shaped in a direct way. However, it is possible to indirectly influence these quantities by appropriate choices for the user controlled parameters in the control design criterion. These user controlled quantities are the reference signal $r(t)$, the reference model $T_d(q)$, the controller structure and the weightings $\lambda$, $L_u(q)$ and $L_u(q)$. The exact influence of these quantities is rather complex, but some insight can be gained by studying expressions of the control design criterion in the frequency domain.
10.4 Implementation Issues and Design Choices

Using Parseval’s formula (2.12) and assuming that a reference model $T_d(q)$ is used, it is possible to approximate the control design criterion as follows:

$$J(\rho) \approx \frac{1}{4\pi} \int_{-\pi}^{\pi} \left\{ |[T_0(q, \rho) - T_d(q)]^2|L_y(q)|^2 + \lambda |L_u(q)|^2 |S_0(q, \rho)|^2 |C_r(q, \rho)|^2 \right\} d\omega,$$

$$(10.38)$$

see [21]. In (10.38), $\Phi_{rr}$ and $\Phi_{uv}$ are the spectra of $r(t)$ and $v(t)$, respectively. From (10.38), it can be concluded that at frequencies where the power spectrum $\Phi_{rr}$ dominates the power spectrum $\Phi_{uv}$ and $\lambda |L_u(q)|^2$ is small compared to $|L_y(q)|^2$, the controller parameters will converge to a value that makes the difference between $T_0(q, \rho)$ and $T_d(q)$ small. Hence, it is possible to influence the closed loop response by suitable choices of the design parameters $\sqrt{L_u(q)}$, $L_y(q)$, $\Phi_{rr}$ and $T_d(q)$.

In general, the frequency weighting filters $L_u(q)$ and $L_y(q)$ can be used to focus the attention of the controller on specific frequency ranges in the input and output response of the closed loop system, respectively. In this way, it is possible to suppress undesirable oscillations in these signals, for example.

The choice of the reference trajectory $r(t)$ is a very important one. As pointed out in [29, Section 3.1], it is often noticed that the objective function is nonconvex, leading to the existence of local minima. Consequently, numerical optimization may be difficult due to the fact that the surface of the criterion can be very rough, thus allowing only small steps in each direction. Generally, it can be concluded that the iterative algorithm ends up in a local minimum. As a result, the optimal controller may be only slightly better than the initial controller. Therefore, this can be seen as a serious disadvantage of the iterative feedback tuning concept. In order to find the global minimum, it is required that the initial controller is close to the globally optimal one. This implies that for a weakly tuned controller the process output $y(t, \rho)$ should not differ too much from the desired process output $y_d(t)$. Therefore, in this situation it is advised to choose the reference signal $r(t)$ in such a way that it can easily be tracked by the system.
Chapter 11

Lifted Iterative Learning Control

11.1 Introduction

Basically, iterative learning control (ILC) is a technique that can be used to improve the transient response and tracking performance of systems that execute the same trajectory over and over again. In this context, one can think of a reticle stage, a wafer stage and a pick and place robot, for instance. The approach is motivated by the observation that if the system controller is fixed and if the system’s operating conditions are the same each time it executes, then any errors in the output response will be repeated during each operation. These errors can be recorded during system operation and can subsequently be used to compute modifications to the input signal that will be applied to the system during the next operation, or trial, of the system. Summarizing, it can be stated that ILC updates the feedforward control signal of a system executing a repeating task iteratively in such a way that the tracking error decreases.

Roughly, two different approaches to ILC design can be distinguished. The classic approach is based on frequency domain design, where use is made of a learning filter $L$ and a robustness filter $Q$, see [1] and [27]. This approach has several drawbacks, such as the limited design freedom. Moreover, the observation and control window necessarily have the same dimension and the determination of $L$ and $Q$ is not straightforward. Another disadvantage is that the analysis of noise propagation is not straightforward either. In this respect, the lifted approach, based on time domain design, provides many interesting features. In this lifted representation of the learning control system, the system dynamics are described by a static map, whereas the dynamics of the learning mechanism are described by a difference equation in the trial domain. Advantages of this approach are the large design flexibility, the free choice of observation and control window and the improved analysis properties. Due to these advantages, the discussion will be restricted to lifted iterative learning control in this chapter.

In [14, Chapter 3], it is shown that ILC can be described as a multivariable feedback system in a generalized plant setting. This description allows the system under consideration to be time-varying or even nonlinear. In this chapter, however, attention is restricted to linear time-invariant systems. Moreover, only discrete time single input single output systems are taken into account. The reason for this is that these systems are most common in practice.

One of the basic elements in lifted iterative learning control is the so-called lifted system representation. This way of representing a system is discussed in Section 11.2. Next, the
control design problem is formulated in Section 11.3. Based on the internal model principle, it is possible to derive a general update law. Using this update law, the steady state convergence properties are investigated in Section 11.4. Finally, in Section 11.5 a multi-objective design method is presented for the design of the learning matrix.

11.2 Lifted System Representation

Consider the general closed loop system $P_d$ depicted in Figure 11.1. Here, $P$ is the process and $C$ is a feedback controller that is fixed. More specifically, $P$ is assumed to be a SISO linear time-invariant process. Furthermore, $r$ denotes the reference trajectory, $e$ the tracking error, $y$ the process output, $u_{ff}$ the feedforward input signal, $u_{fb}$ the feedback input signal, $u$ the process input, $w$ the disturbances and $v$ the measurement noise.

![Figure 11.1: General closed loop system with feedback and feedforward control.](image)

A state-space description of the SISO LTI process $P$ is given by

$$
\begin{align*}
x_{k+1}^P &= A^P x_k^P + B^P (u_{fb,k} + u_{ff,k}) + B_w w_k \\
y_k &= C^P x_k^P + v_k,
\end{align*}
$$

where $w_k$ and $v_k$ are assumed to be Gaussian white noise with covariance matrices $W$ and $V$, respectively. The fixed controller $C$ is also represented by a state-space description:

$$
\begin{align*}
x_{k+1}^C &= A^C x_k^C + B^C (r_k - y_k) \\
u_{fb,k} &= C^C x_k^C + D^C (r_k - y_k).
\end{align*}
$$

Combination of (11.1,11.2) and (11.3,11.4) makes it possible to write the closed loop system as follows:

$$
\begin{align*}
x_{k+1} &= A x_k + B \bar{u}_k \\
e_k &= C x_k + D \bar{u}_k,
\end{align*}
$$

where $\bar{u}_k = [u_{ff,k}, u_{ff,k}^T, v_k, r_k]^T$ represents the external input signals. Explicit expressions for the matrices $A$, $B$, $C$ and $D$ in (11.5,11.6) can be found in [35, Section 2]. From (11.5,11.6), it is obvious that in iterative learning control the system output is defined to be the tracking error $e_k$.

Next, the behaviour of the closed loop system (11.5,11.6) is considered in case a repeating task is performed. This task consists of the execution of a desired reference trajectory $r$.
starting from an initial condition \( x_0 \). The reference trajectory \( r \) is of finite length \( N \) and constant in the trial domain. The objective of ILC now is to compute a feedforward signal \( u_{ff,k} \) of finite length \( N \) for the current trial. This is done in such a way that the error in the current trial is minimized, using the errors from previous trials. Later on, it will be obvious that it is possible to choose the control window different from the observation window. This is an important property, because it is often desired to take the observation window larger than the control window, in order to observe the influence of the feedforward signal outside the control window.

For discrete time signals and systems, the concept of signal and system lifting can be applied in a straightforward way. Essentially, a discrete time signal can be represented by a vector, in which each element denotes the signal value at the corresponding time instant. Taking this observation into account, introduce the lifted feedforward signal \( u_{ff,l} \) that represents the feedforward input to the system in trial \( l \) according to:

\[
 u_{ff,l} = [u_{ff,0}^l \ u_{ff,1}^l \ \ldots \ u_{ff,N-1}^l]^T. \tag{11.7} 
\]

Similarly, the lifted representation of the tracking error in trial \( l \) is given by:

\[
 e^l = [e_0^l \ e_1^l \ \ldots \ e_{N-1}^l]^T. \tag{11.8} 
\]

Based on the lifted representation of these signals and the remaining signals in \( \ddot{u} \), the lifted system \( \mathcal{P} \) is given by:

\[
 e^l = \mathcal{P}_1 u_{ff,l}^l + \mathcal{P}_2 w^l + \mathcal{P}_3 v^l + \mathcal{P}_4 r + \mathcal{P}_x x_0^l, \tag{11.9} 
\]

where \( \mathcal{P}_1 : \mathbb{R}^N \rightarrow \mathbb{R}^N, \mathcal{P}_2 : \mathbb{R}^{Nw} \rightarrow \mathbb{R}^N, \mathcal{P}_3 : \mathbb{R}^N \rightarrow \mathbb{R}^N \) and \( \mathcal{P}_4 : \mathbb{R}^N \rightarrow \mathbb{R}^N \) are Toeplitz matrices. A lower triangular Toeplitz matrix is defined as follows:

\[
 \mathcal{P}_1 = \begin{bmatrix}
 G_0^1 & 0 & \ldots & 0 \\
 G_0^1 & G_0^2 & \ddots & \vdots \\
 \vdots & \ddots & \ddots & 0 \\
 G_{N-1}^1 & \ldots & G_1^1 & G_0^N 
\end{bmatrix}. \tag{11.10} 
\]

Hence, a lower triangular Toeplitz matrix is a matrix in which the elements on each diagonal are identical and all elements above the main diagonal are zero. This matrix can be used to describe the mapping from an input vector to an output vector for a discrete time causal linear time-invariant system, see [14, Section 3.2.3]. The elements \( G_k^i = D^i \) and \( G_k^i = CA^{k-1}B^i \), for \( k = 1, 2, \ldots, N - 1 \) and \( i = 1, \ldots, 4 \), are the Markov parameters of the closed loop system (11.5, 11.6). Notice that the lifted system representation (11.9) is a static multivariable system with \( N \) outputs. In addition, the matrix \( \mathcal{P}_x \) is given by:

\[
 \mathcal{P}_x = [CT \ (CA)^T \ \ldots \ (CA^{N-1})^T]^T. \tag{11.11} 
\]

A disadvantage of iterative learning control is that these Toeplitz matrices are determined only once by execution of specific experiments. As a result, changes in the motion system dynamics are not taken into account during operation of the learning control system.
11.3 Control Design Problem

In iterative learning control, it is generally assumed that the initial condition \( x_0 \) is zero, or at least constant, for each trial. This assumption implies that the initial condition in each trial is not dependent on the ILC action in the previous trial, which is not very realistic. Fortunately, this assumption is not necessary in the lifted approach. Assume that \( x_0 \) is a random variable, with zero mean and covariance matrix \( X \). Here, \( X \) is the stationary closed loop variance of the closed loop system (11.5,11.6):

\[
X = AXA^T + \begin{bmatrix} BwWb_w^T & 0 \\ 0 & B^CVB^C_T \end{bmatrix},
\]

(11.12)

see [35, Section 2.3]. As a result of the fact that the random variables \( u^l, v^l \) and \( x^l_0 \) are uncorrelated, it is possible to give a compact formulation of the lifted system representation (11.9) according to:

\[
e^l = \mathcal{P}_1 u^l_{ff} + \mathcal{P}_4 r + G\hat{v}_r,
\]

(11.13)

where \( \hat{v} \in \mathbb{R}^N \) is a generalized random variable with covariance matrix \( I \). Furthermore, \( G \) is defined as the Choleski factor of the composite covariance matrix, i.e., \( GG^T = \mathcal{P}_2 W\mathcal{P}_2^T + \mathcal{P}_3 V\mathcal{P}_3^T + \mathcal{P}_4 X\mathcal{P}_4^T \). This compact lifted system representation is depicted in Figure 11.2, where it is interconnected with the learning controller \( \text{ILC} \).

![Figure 11.2: Lifted system representation and interconnection with learning controller.](image)

Taking into account Figure 11.2, it is possible to define the following ILC design problem.

**Definition 14 (ILC design problem).** Find a feedback control law \( u^l_{ff} = \text{ILC}(e) \) in the trial domain, such that the feedback interconnection of \( \text{ILC} \) and \( \mathcal{P} \) is stable and such that the tracking error \( e \) converges to a sufficiently small value.

11.4 Internal Model Principle and Convergence

Typically, the learning mechanism is described by a difference equation in the trial domain:

\[
u^l_{ff}^{l+1} = u^l_{ff} + Le^l.
\]

(11.14)

A motivation for this choice of the update law is given by using the internal model principle, see [16]. Suppose that the closed loop system operates in absence of noise. Furthermore, the reference trajectory \( r \) is fixed in the trial domain. Obviously, it is desired that the closed loop system asymptotically tracks this reference trajectory. In order to achieve this, the internal model principle states that it is necessary to include a model of the reference dynamics in
the controller. In case of a constant reference trajectory, this is an integrator in the trial domain. This approach is illustrated in Figure 11.3, in which the update law (11.14) can be recognized. Here, \( Z \) denotes the one trial delay operator.

\[
\begin{align*}
\dot{u}^l_f & = r_f + LP_1 u^l_f + LP_2 r + \delta, \\
\text{Figure 11.3: Lifted system representation interconnected with first-order iterative learning control system.}
\end{align*}
\]

However, the internal model principle only holds in case the number of inputs is at least equal to the number of outputs. Due to the fact that \( P_1 \) is not square in general, see [35, Section 2.2], it can be concluded that the internal model principle does not apply. For this reason, it is necessary to investigate the steady state error properties of the nonsquare ILC scheme.

Combination of (11.13) and (11.14) makes it possible to write the system in Figure 11.3 as follows:

\[
\begin{align*}
\dot{u}^{l+1}_f & = u^l_f + \Delta u^l_f, \\
e^l & = P_1 u^l_f + P_2 r + \dot{\delta},
\end{align*}
\]

which describes the dynamics of the ILC scheme in the trial domain. In the absence of noise and for any stabilizing control law \( \dot{u}^l_f = Le^l \), it is possible to rewrite (11.15,11.16) as follows:

\[
\begin{align*}
\dot{u}^{l+1}_f & = u^l_f + LP_1 u^l_f + LP_2 r. \\
\end{align*}
\]

In steady state, it holds that \( \dot{u}^{l+1}_f = \dot{u}^l_f \), which implies that the learning iteration converges to a fixed point \( u^*_f \) according to:

\[
0 = LP_1 u^*_f + LP_2 r.
\]

After application of model reduction, a reduced system representation \( P_1 \in \mathbb{R}^{N_x \times N_u} \) is obtained that has full column rank, see [35, Section 2.2]. Hence, it is possible to choose the learning matrix \( L \) in such a way that \( LP_1 \) is nonsingular. Substitution of \( u^*_f \) from (11.18) into \( e^l \) from (11.16) gives the steady state tracking error:

\[
\begin{align*}
e^* & = P_2 r - P_1 (LP_1)^{-1}LP_2 r.
\end{align*}
\]

Obviously, it is possible to achieve zero steady state tracking error if and only if the reference trajectory \( r \) is chosen such that \( P_2 r = P_1 z \), for any \( z \in \mathbb{R}^{N_u} \). This can be achieved by minimization of the difference between \( r \) and an ideal desired reference \( r^* \) in a least squares sense. Formally, this can be stated by:

\[
\min_{r,z}(r - r^*)^T Q^*(r - r^*),
\]
subject to:
\[ P_A r + P_1 z = 0. \]  
(11.21)

Hence, it can be concluded that the integrating learning update law is a good choice. In the absence of noise, it provides zero steady state tracking error for any stabilizing gain.

### 11.5 Learning Matrix

After the discussion on convergence properties in the previous section, the question remains how to determine the learning matrix \( L \). Due to the fact that iterative learning control in the lifted approach is based on a time domain description of the system, it is possible to utilize time domain (optimal) control methods for the design of the learning matrix. In [14, Section 4.4] and [10], results for the linear quadratic regulator (LQR) optimal control design method are discussed. This method provides a learning matrix \( L \) that stabilizes the feedback interconnection of \( L \) and \( P \) in the trial domain, which can be expected from the viewpoint of optimal control theory. An extension to this is given by the multi-objective design method, proposed in [35, Section 4] and used in [14, Section 4.6]. In this method, the convergence speed is optimized subject to a bound on the closed loop variance due to stochastic initial conditions, process disturbances and measurement noise. Hence, the multi-objective design method clearly illustrates the advantageous property of lifted ILC with respect to the analysis of noise propagation. For this reason, the multi-objective design problem is highlighted in this section. The solution can be found in [35, Section 4] and in [14, Section 4.6].

#### 11.5.1 Multi-Objective Design Method

In the multi-objective design method, it is possible to determine the learning matrix \( L \) in such a way that the convergence speed is optimal, while taking into account a prespecified bound on the closed loop variance. It is assumed that the equality \( P_A r = P_1 z \) holds for any \( z \in \mathbb{R}^{N_u} \). Then, the lifted system representation can be transformed into:

\[ \tilde{u}^{l+1} = \tilde{u}^l + \Delta \tilde{u}^l \]
\[ e'^l = P_1 \tilde{u}^l + Go'^l, \]  
(11.22)
\( (11.23) \)

where \( \tilde{u} = u_{ff} + z \) and \( \tilde{u}_0 = -z \). Hence, the initial condition is not equal to zero. The problem now is to find a learning matrix \( L \) by taking into account two control objectives.

The first control objective is to limit the steady state variance of the feedforward signal and of the tracking error. To this end, an objective function is defined for each element \( \tilde{u}^{l,i} \), \( i = 1, \ldots, N_u \), of the lifted feedforward input \( \tilde{u}^l \):

\[ J_i^l = \lim_{M \to \infty} \frac{1}{M} E \left[ \sum_{i=1}^{M} \tilde{u}_{i}^{l,i} \right], \]  
(11.24)

where \( M \) denotes the number of trials. A motivation for this choice of the objective function \( J_i^l \) is given in [35, Section 4.1].

The second control objective is to guarantee a certain speed of convergence of the learning controller. In [14, Section 4.3.4], it is stated that the convergence rate of the learning controller is determined by the location of the closed loop trial poles. Optimal performance
can be achieved by constraining the closed loop trial poles to be located in a prescribed region within the unit circle. However, dependent on the reference trajectory it may not be necessary for all trial poles to be located within a prescribed region in order to obtain fast convergence. Furthermore, the enforcement of all closed loop trial poles may lead to a high gain ILC feedback, which in turn may lead to significant noise amplification. For this reason, a deterministic LQR objective is defined:

\[ J^{II} = \sum_{l=1}^{\infty} E(\tilde{u}^{l})^{T} Q E(\tilde{u}^{l}) + E(\Delta \tilde{u}^{l})^{T} R E(\Delta \tilde{u}^{l}). \] (11.25)

The combination of both objectives mentioned above leads to the multi-objective design problem.

**Definition 15 (Multi-objective design problem).** Find a static feedback $L$ such that the LQR objective $J^{II}$ is minimized, while satisfying a prespecified bound on the closed loop variance. Formally, this can be stated by:

\[ \min_{L} J^{II}, \] (11.26)

subject to:

\[ J_{i}^{I} \leq \gamma_{i}, \] (11.27)

for $i = 1, \ldots, N_u$.

When considering a diagonal weighting $Q = I$ and $R = \beta I$, the multi-objective problem becomes fully diagonal and the optimal solution can be found easily, see [35, Section 4.3] and [14, Section 4.6.1].
Chapter 12

Evaluation of Control Concepts

12.1 Introduction

In the previous chapters, four different control concepts have been discussed. These are the self-tuning regulator in Chapter 8, model reference adaptive control in Chapter 9, iterative feedback tuning in Chapter 10 and lifted iterative learning control in Chapter 11. Obviously, these control concepts have been selected because they are related to Machine-In-the-Loop control optimization. The subject of this chapter is to investigate this relation.

In order to evaluate the relation between the control concepts and Machine-In-the-Loop control optimization, it is necessary to define certain criteria. In this context, the desired properties of Machine-In-the-Loop control optimization mentioned in Section 1.2 are taken as a starting point. Additional criteria will be defined in this chapter. Taking into account these criteria, it is possible to evaluate the properties of each control concept with respect to these criteria. This evaluation is performed in Section 12.2.

12.2 Evaluation

Formulation of control design problem as optimization problem The self-tuning regulator is the only concept in which the control design problem is not formulated as an optimization problem. Instead, it uses model-based control design and updates the controller using recursively identified models of the system according to the certainty equivalence principle. In the remaining three concepts, objective functions are defined that typically contain terms concerning the tracking error and the control effort. In model reference adaptive control and iterative feedback tuning the controller parameters are optimized, whereas in lifted iterative learning control the feedforward signal is optimized.

Solution of minimization problem using a gradient based optimization method In model reference adaptive control, the minimization problem is solved by application of the steepest descent method, which is a gradient based optimization method. Similarly, use is made of a quasi-Newton method in the iterative feedback tuning concept, with different possibilities to approximate the Hessian matrix. In lifted iterative learning control, use is made of a quasi-Newton method as well. In this case, the update is based on both model knowledge and measurement data.
Gradient computations performed by using measurement data The purpose of model reference adaptive control is to tune the controller such that a certain desired closed loop response is achieved, generated by the reference model. Assuming this already holds, the reference model can be used to compute an approximation of the gradient. Hence, no use is made of measurement data. Iterative feedback tuning, however, approximates the gradient by using measurement data only. Here, the measurement data is obtained by performing a series of dedicated experiments on the actual system. In lifted iterative learning control, the update is given by the learning matrix multiplied by the measured tracking error. Because the learning matrix is model-based, both model knowledge and measurement data are used to compute the update.

Adaptation to changes in motion system dynamics In both the self-tuning regulator and model reference adaptive control, the controller is adapted continuously. As a result, these control concepts adapt to changes in the motion system dynamics. In contrast to these methods, iterative feedback tuning requires one dedicated experiment during each iteration. Consequently, this method is not able to adapt to changes in the motion system dynamics during normal operation. However, if an iteration is performed, measurement data is used and, thus, changes in the motion system dynamics are taken into account. In lifted iterative learning control, Toeplitz matrices are used to describe the system. Because these matrices are determined only once, changes in the motion system dynamics are not taken into account directly. However, during operation of the learning control system, the tracking error is measured and used to compute the update. In other words, the effect of changes in the motion system dynamics, expressed by the tracking error, is taken into account in the computation of the update.

Generic method with large application area It is known that the self-tuning regulator can be applied to feedback as well as feedforward control design problems. On the contrary, model reference adaptive control is typically used in feedback control. Iterative feedback tuning can be used to simultaneously update the feedback and the feedforward controller parameters. Finally, lifted iterative learning control has been developed specifically for feedforward control design.

Optimality of performance (global / local) In case of the self-tuning regulator it is generally difficult to prove that the resulting controller guarantees optimal performance. This is only possible for a small number of algorithms under very restrictive conditions. In model reference adaptive control, it can be guaranteed that the tracking error reduces to zero, by using Lyapunov theory. The main difficulty in iterative feedback tuning is that the objective function generally is nonconvex. Hence, the initial tuning of the controller parameters must be close to the optimal tuning in order to find the global minimum. Otherwise, the algorithm ends up in a local minimum, possibly leading to a controller that is only slightly better than the initial one. In case of lifted iterative learning control, it can be shown that the algorithm eliminates all systematic errors that occur during execution of a repeating task.

Automated procedure Basically, it can be concluded that all four control concepts are automated procedures. In case of iterative feedback tuning, however, it may be necessary to manually check the Bode plot of the new controller, in order to guarantee stability.
of the closed loop system. As a result, this method is not fully automated, unless a different approach is found to guarantee stability of the resulting closed loop system.

Applicaton of optimized controller to variety of tasks Obviously, in case of the self-tuning regulator and model reference adaptive control, the controller is optimized during normal operation. Hence, it is possible to apply arbitrary reference trajectories. In iterative feedback tuning, the controller is optimized for the reference trajectory that is applied during the dedicated experiments. Consequently, the performance will decrease in case different reference trajectories are applied. Lifted iterative learning control is highly inflexible with respect to changes in the reference trajectory. In case the reference trajectory is changed, the learning process needs to be performed again from scratch.

Demonstrable / guaranteed convergence Convergence is difficult to prove in case of the self-tuning regulator. It is only possible for a small class of algorithms under very restrictive conditions. This also holds for a model reference adaptive control system, where parameter convergence can be guaranteed only in case certain excitation conditions are fulfilled. For the iterative feedback tuning concept, a convergence proof is available. However, this proof imposes restrictions on the noise properties that are generally not satisfied in practice. The convergence proof in case of the lifted iterative learning control concept imposes restrictions as well, but these restrictions can be satisfied more easily.

Model knowledge limited to simple, cheap models Determination of a process model is a basic element in the self-tuning regulator. A straightforward approach is to estimate the parameters of the transfer function of the process. However, in order to obtain the transfer function, it is necessary to choose a model structure and a corresponding parameterization, which are expensive procedures in general. In case of model reference adaptive control and iterative feedback tuning, no model knowledge is required. In the lifted iterative learning control concept, use is made of Toeplitz matrices in order to describe the system. These Toeplitz matrices can be obtained relatively easy. Hence, the use of model knowledge is limited to simple, cheap models in this case.
Chapter 13

Conclusions and Recommendations

This survey concerns the investigation of the Machine-In-the-Loop control optimization concept. To this end, two main questions have been formulated in Section 1.3:

- What are the most relevant theoretical aspects to and building blocks for Machine-In-the-Loop control optimization that are available in related fields?
- Which related control techniques are already available and to what extent are they useful with respect to Machine-In-the-Loop control optimization?

Answers to these questions are given in Section 13.1. Finally, recommendations for future research are given in Section 13.2.

13.1 Conclusions

The conclusions are categorized according to the questions outlined above.

Theoretical Aspects and Building Blocks

An important building block for Machine-In-the-Loop control optimization is given by discrete time signals and discrete time signal analysis. Measurement and analysis of signals is of crucial importance in a number of situations. In order to evaluate the performance of a closed loop system, for instance, it is often required to measure and analyze the tracking error. Also, the tracking error may be used for control purposes. Another situation occurs in case a process model needs to be determined. This requires measurement and analysis of multiple input and output signals. Hence, it can be concluded that signals and signal analysis are important aspects with respect to Machine-In-the-Loop control optimization.

Signals and signal analysis are not the only important aspects in case a process model needs to be determined. In addition, it is necessary to have knowledge about systems and system modeling. This is the second important building block for Machine-In-the-Loop control optimization. In this context, only single input single output linear time-invariant systems are considered, because they form the most important class of dynamical systems considered in theory and practice. An important aspect in system modeling is the determination of the unknown parameters in the process model. This is done by using parameter estimation methods. Thus, it can be concluded that systems and system modeling are important building blocks for Machine-In-the-Loop control optimization.
The last important building block for Machine-In-the-Loop control optimization is represented by optimization techniques. The basic idea behind Machine-In-the-Loop control optimization and other related control concepts is to update the controller parameters in such a way that a certain objective function is minimized. In order to adequately solve these minimization problems, it is necessary to have a fundamental knowledge of optimization techniques. More specifically, unconstrained optimization techniques are of main interest, because the controller parameters generally are unconstrained. Summarizing, it can be concluded that optimization techniques form an important building block for Machine-In-the-Loop control optimization.

Related Control Techniques

Four different control concepts that are related to Machine-In-the-Loop control optimization have been investigated. These are the self-tuning regulator, model reference adaptive control, iterative feedback tuning and lifted iterative learning control. Generally, it can be concluded that the objective of all concepts is to minimize the tracking error by changing the controller parameters. However, each method tries to realize this objective in a different way. In this context, one can think of design / tuning of either the feedback controller or the feedforward controller, but also simultaneous design / tuning of both controllers. A consequence of these different approaches is that each control concept imposes different assumptions with respect to process and controller, for instance. Hence, it can be concluded that the scope of these approaches is rather narrow and, furthermore, none of the methods satisfies all desired properties of Machine-In-the-Loop control optimization.

Nevertheless, it can be concluded that the investigated control concepts use common tools / techniques in order to solve the corresponding specific problems. More specifically, in case of model reference adaptive control, iterative feedback tuning and lifted iterative learning control, use is made of optimization techniques. It is obvious that the definition and the solution of the actual optimization problem are different for each method.

Hence, the main conclusion is that the investigated control concepts all solve specific problems, thereby imposing specific assumptions. As a result, it is not possible to use a single approach as a basis for Machine-In-the-Loop control optimization. However, the observation that three methods are based on optimization techniques can be taken as a starting point in future research.

13.2 Recommendations

In the previous section, it is concluded that the four investigated control concepts all solve specific control design problems. Furthermore, none of these control concepts satisfies all desired properties of Machine-In-the-Loop control optimization. Nevertheless, it is observed that three control concepts make use of optimization techniques in order to solve the problem. These observations lead to the following recommendations for future research:

- Investigate whether a general approach, denoted Machine-In-the-Loop control optimization, can be developed, based on an optimization point of view, with the following properties:
  - Adaptation to changes in motion system dynamics
13.2 Recommendations

- Generic method with large application area
- Optimal performance
- Automated procedure
- Application of optimized controller to variety of tasks
- Demonstrable / guaranteed convergence
- Model knowledge limited to simple, cheap models

- In the first instance, focus on the use of Machine-In-the-Loop control optimization in feedforward control design and machine calibration. These applications are considered to be most relevant for Philips Applied Technologies, because they are directly related to machine performance. Also, they put less focus on stability issues, which are crucial in feedback control design.
Appendix A

Iterative Feedback Tuning: Gradient Computations

A.1 Process Output Gradient

From the process (10.19) and the two-degree-of-freedom controller (10.20), it follows that the closed loop system can be described by:

\[(1 + P_0(q)C_y(q, \rho))y(t, \rho) = P_0(q)C_r(q, \rho)r(t) + v(t).\]  \hspace{1cm} (A.1)

Differentiation of the output \(y(t, \rho)\) with respect to the controller parameter vector \(\rho\) leads to:

\[
\frac{\partial y(t, \rho)}{\partial \rho} = \frac{P_0(q)}{1 + P_0(q)C_y(q, \rho)} \frac{\partial C_r(q, \rho)}{\partial \rho} r(t) - \frac{C_r(q, \rho)P_0^2(q)}{(1 + P_0(q)C_y(q, \rho))^2} \frac{\partial C_y(q, \rho)}{\partial \rho} r(t) - \frac{P_0(q)}{(1 + P_0(q)C_y(q, \rho))^2} \frac{\partial C_y(q, \rho)}{\partial \rho} v(t) = \quad (A.2)
\]

\[
= \frac{1}{C_r(q, \rho)} \frac{\partial C_r(q, \rho)}{\partial \rho} T_0(q, \rho)r(t) - \frac{1}{C_r(q, \rho)} \frac{\partial C_y(q, \rho)}{\partial \rho} (T_0^2(q, \rho)r(t) + T_0(q, \rho)S_0(q, \rho)v(t)). \quad (A.3)
\]

In (A.3), the quantities \(C_r(q, \rho), \frac{\partial C_r(q, \rho)}{\partial \rho}\) and \(\frac{\partial C_y(q, \rho)}{\partial \rho}\) are known functions of \(\rho\) that depend on the parameterization of the restricted complexity controller. The quantities \(T_0(q, \rho)\) and \(S_0(q, \rho)\), however, depend on the unknown system \(P_0(q)\) and are thus not computable.

Observe that the following relation holds:

\[
T_0(q, \rho)y(t, \rho) = T_0^2(q, \rho)r(t) + T_0(q, \rho)S_0(q, \rho)v(t). \quad (A.4)
\]

Using (A.4), it is possible to rewrite (A.3) according to:

\[
\frac{\partial y(t, \rho)}{\partial \rho} = \frac{1}{C_r(q, \rho)} \left[ \frac{\partial C_r(q, \rho)}{\partial \rho} T_0(q, \rho)r(t) - \frac{\partial C_y(q, \rho)}{\partial \rho} T_0(q, \rho)y(t, \rho) \right]. \quad (A.5)
\]
Iterative Feedback Tuning: Gradient Computations

Substitution of (10.19) into (10.20) leads to the following expression for the process input:

\[
u(t, \rho) = \frac{C_r(q, \rho)}{1 + C_y(q, \rho)P_0(q)} r(t) - \frac{C_y(q, \rho)}{1 + C_y(q, \rho)P_0(q)} v(t) = S_0(q, \rho)(C_r(q, \rho)r(t) - C_y(q, \rho)v(t)). \tag{A.7}
\]

Furthermore, differentiation of (10.26) with respect to the parameter vector \( \rho \) leads to:

\[
\frac{\partial S_0(q, \rho)}{\partial \rho} = -\frac{1}{C_r(q, \rho)} T_0(q, \rho) S_0(q, \rho) \frac{\partial C_y(q, \rho)}{\partial \rho}. \tag{A.8}
\]

Then, it follows that:

\[
\frac{\partial u(t, \rho)}{\partial \rho} = S_0(q, \rho) \left[ \frac{\partial C_r(q, \rho)}{\partial \rho} r(t) - \frac{\partial C_y(q, \rho)}{\partial \rho} v(t) \right] + \\
\frac{\partial S_0(q, \rho)}{\partial \rho} \left[ C_r(q, \rho)r(t) - C_y(q, \rho)v(t) \right] = \tag{A.9}
\]

\[
S_0(q, \rho) \left[ \frac{\partial C_r(q, \rho)}{\partial \rho} r(t) - \frac{\partial C_y(q, \rho)}{\partial \rho} \right] T_0(q, \rho) r(t) + \\
+ v(t) - \frac{C_y(q, \rho)}{C_r(q, \rho)} T_0(q, \rho) v(t) \right] = \tag{A.10}
\]

\[
S_0(q, \rho) \left[ \frac{\partial C_r(q, \rho)}{\partial \rho} r(t) - \frac{\partial C_y(q, \rho)}{\partial \rho} \right] (T_0(q, \rho)r(t) + S_0(q, \rho)v(t)) = \tag{A.11}
\]

\[
S_0(q, \rho) \left[ \left( \frac{\partial C_r(q, \rho)}{\partial \rho} - \frac{\partial C_y(q, \rho)}{\partial \rho} \right) r(t) + \frac{\partial C_y(q, \rho)}{\partial \rho} (r(t) - y(t, \rho)) \right], \tag{A.12}
\]

which is the required expression.
Bibliography


Distribution List

ASML
Edwin van Donkelaar
Rob de Jongh
Ramidin Kamidi
Tim Tso

Delft University of Technology
Camile Hol
Eduard van der Meché
Carsten Scherer

Eindhoven University of Technology
Matthijs Boerlage
Okko Bosgra
Jeroen van Helvoort
Tom Oomen
Maarten Steinbuch
Jeroen van de Wijdeven

Philips Applied Technologies
Georgo Angelis
Gregor van Baars
Jan van Eijk
Erik Gaal
Marcel Heertjes
Norbert Jansen
Marco Koevoets
Jan Nouwens
Felix Peeters
Patrick Philips
Adrian Rankers
Rob Rijs
Gert van Schothorst
Rob Tousain
Arno de Vet
Huub Vroomen
Marc van de Wal