

**ISSUES OF ALGEBRA AND  
OPTIMALITY IN ITERATIVE  
LEARNING CONTROL**

**JARI  
HÄTÖNEN**

Department of Process and  
Environmental Engineering,  
University of Oulu

OULU 2004





*JARI HÄTÖNEN*

**ISSUES OF ALGEBRA AND  
OPTIMALITY IN ITERATIVE  
LEARNING CONTROL**

Academic Dissertation to be presented with the assent of the Faculty of Technology, University of Oulu, for public discussion in Raahensali (Auditorium L10), Linnanmaa, on June 11th, 2004, at 12 noon.

OULUN YLIOPISTO, OULU 2004

Copyright © 2004  
University of Oulu, 2004

Reviewed by  
Professor Heikki Koivo  
Professor Pertti Mäkilä

ISBN 951-42-7350-8 (nid.)  
ISBN 951-42-7351-6 (PDF) <http://herkules.oulu.fi/isbn9514273516/>  
ISSN 0355-3213 <http://herkules.oulu.fi/issn03553213/>

OULU UNIVERSITY PRESS  
OULU 2004

# **Hätönen, Jari, Issues of algebra and optimality in Iterative Learning Control**

Department of Process and Environmental Engineering, University of Oulu, P.O.Box 4300, FIN-90014 University of Oulu, Finland

2004

Oulu, Finland

## ***Abstract***

In this thesis a set of new algorithms is introduced for Iterative Learning Control (ILC) and Repetitive Control (RC). Both areas of study are relatively new in control theory, and the common denominator for them is that they concentrate on controlling systems that include either reference signals or disturbances which are periodic. This provides opportunities for using past information or experience so that the control system *learns* the control action that results in good performance in terms of reference tracking or disturbance rejection.

The first major contribution of the thesis is the algebraic analysis of ILC systems. This analysis shows that in the discrete-time case ILC algorithm design can be considered as designing a multivariable controller for a multivariable static plant and the reference signal that has to be tracked is a multivariable step function. Furthermore, the algebraic analysis reveals that time-varying algorithms should be used instead of time-invariant ones in order to guarantee monotonic convergence of the error in norm.

However, from the algebraic analysis it is not clear how to select the free parameters of a given ILC algorithm. Hence in this thesis optimisation methods are used to automate this design phase. Special emphasis is placed on the so called Norm-Optimal Iterative Learning Control (NOILC) that was originally developed in (Amann:1996) as a new result it is shown that a convex modification of the existing predictive algorithm will result in a considerable improvement in convergence speed. Because the NOILC algorithm is computationally quite complex, a new set of Parameter-Optimal ILC algorithms are derived that converge under certain assumptions on the original plant. Three of these new algorithms will result in monotonic convergence to zero tracking error for an arbitrary discrete-time, linear, time-invariant plant. This a very strong property that has been earlier reported for only a small number of ILC algorithms.

In the RC case it is shown that an existing RC algorithm that has been widely analysed and used in the research literature is in fact highly unrobust if the algorithm is implemented using sampled-data processing. Consequently, in this thesis a new optimality based discrete-time RC algorithm is derived, which converges to zero tracking error asymptotically for an arbitrary linear, time-invariant discrete-time plant under mild controllability and observability conditions.

*Keywords:* algebraic systems theory, Iterative Learning Control, Repetitive Control, robust control



*"Minä on aina toinen, maailma illuusiota, aika  
huiputusta, kieli sanoinkuvaamattoman ylle heitetty  
heiveröinen verho, kohteliaisuus julmuuden lykkäystä,  
mielihyvä moraalialue ja hellyys ainoa näköalamme"*

*"I'm always somebody else, reality is equal to illusion,  
time is equal to cheating, language is a thin cover thrown  
over the inexpressible, politeness is equal to delaying  
brutality, to have pleasure is to have morals, and being  
gentle is our only hope"*

*Roger-Pol Droit*



## Preface

This thesis presents research work that I have done within the field of Iterative Learning Control and Repetitive Control between October 2000 and October 2003. The thesis is written in monograph format because I feel that it is the format that allows a researcher to show his ability to carry out independent research work and to report it in a logically sound manner.

Originally my interest in systems theory was initiated by my supervisor Professor Raimo Ylinen from the University of Oulu, Finland. His main research interest is polynomial systems theory, and he encouraged me to write a thesis on applying this theory on multi-dimensional systems. I would like to thank him for introducing to me this exciting research topic and teaching me (with patience) the fundamental ideas behind polynomial systems theory.

During the first year of my post-graduate studies it turned out that from the vast area of multi-dimensional systems theory Iterative Learning Control attracted me the most. In order to gain a better understanding of Iterative Learning Control, I moved to Sheffield, England, in November 2000, to work with Professor David H. Owens. A lot of the work presented in this thesis are results obtained from the collaboration between Professor Owens and myself and I'm very grateful for his interest in my research work.

Another person that has greatly influenced my Iterative Learning Control research is Professor Kevin L. Moore from the Utah State University. During the second year of my PhD he invited me to visit his institution for two weeks. As a result the publication P2 (see section 1.4) was written on algebraic issues in Iterative Learning Control. I would like to thank him for the two weeks in Utah and sharing his ideas on Iterative Learning Control with me.

This work has been funded by the Academy of Finland, who granted me a position in the Graduate School of Electronics, Telecommunication and Automation (better known as GETA). In addition, I have received a grant from Jenny and Antti Wihuri's foundation, and I would like to thank both organisations for their financial support.

Sheffield, United Kingdom,  
November 30, 2003

Jari Hätönen



# Nomenclature

## Uppercase Roman Letters

$A, B, C, D$	system matrices for a continuous-time state-space equation
$G$	plant model
$G_e$	plant model in super-vector notation
$H$	composite map $GG^*$
$I$	identity operator
$J$	performance criterion
$K(t)$	Riccati feedback gain
$L$	learning operator
$L_n(H, \lambda)$	learning operator for predictive NOILC
$L_2[0, T]$	space of square-integrable functions

## Lowercase Roman Letters

$e(t)$	tracking error
$k$	trial index
$l_2$	space of square-summable sequences
$n$	prediction horizon
$p(t)$	co-state
$p_n(H, \lambda)$	polynomials defining $L_n(H, \lambda)$
$r(t)$	reference signal
$t$	time (either continuous or discrete)
$u(t)$	input function
$x(t)$	state
$x_0$	initial condition for a state-space description
$y(t)$	output
$z_0(t)$	initial condition response (unforced response)

### Greek letters

$\gamma$	(learning gain in "Arimoto-type" algorithms)
$\lambda$	weighting factor in the predictive cost function
$\psi(t)$	adjoint variable
$\Delta$	difference operator
$\Phi, \Gamma$	system matrices for a discrete-time state-space equation
$\xi_k(t)$	predictive term in NOILC

### Miscellaneous

$\langle \cdot, \cdot \rangle$	inner product
$ \cdot $	modulus
$\ \cdot\ $	norm
$\square$	indicates the end of a proof
$\vec{v}$	a super-vector
$\mathbb{N}$	the set of natural numbers
$\mathbb{Z}$	the set of intergers
$\mathbb{R}$	the field of real numbers
$\mathbb{C}$	the field of complex numbers
$M^T$	the transpose of a matrix $M$
$T^*$	the adjoint of an operator $T$
cl	subscript indicating "closed loop"

### Abbreviations

cf.	(Latin <i>confer</i> ) compare
e.g.	(Latin <i>exempli gratia</i> ) for example
etc.	(Latin <i>et cetera</i> ) and so on
i.e.	(Latin <i>id est</i> ) that is
2D	two-dimensional
ARE	Algebraic Riccati Equation
HOILC	High-order Iterative Learning Control
ILC	Iterative Learning Control
MIMO	multiple input, multiple output
nD	n-dimensional
NOILC	Norm-Optimal Iterative Learning Control
PID	proportional, integral, derivative
POILC	Parameter-Optimal Iterative Learning Control
RC	Repetitive Control
SISO	single input, single output

# Contents

Abstract	
Preface	
Nomenclature	
Contents	
1 Introduction . . . . .	15
1.1 What is new in ILC and RC? . . . . .	15
1.1.1 Fundamentals of Iterative Learning Control . . . . .	16
1.1.2 Fundamentals of Repetitive Control . . . . .	20
1.2 A formal definition of Iterative Learning Control . . . . .	22
1.3 A formal definition of Repetitive Control . . . . .	24
1.4 Contributions . . . . .	25
1.5 Organisation of the thesis . . . . .	27
2 Convergence theory for iterative processes . . . . .	30
2.1 Contraction mapping condition . . . . .	31
2.2 Spectral radius condition . . . . .	33
2.3 Spectral radius condition and dynamical systems . . . . .	34
2.4 Convergence analysis of high-order algorithms . . . . .	37
2.5 Summary . . . . .	38
3 Algebraic analysis of ILC systems . . . . .	39
3.1 Matrix representations of discrete-time dynamical systems . . . . .	40
3.2 The process model and invertibility . . . . .	42
3.3 Polynomial description of ILC systems . . . . .	43
3.4 Controller design . . . . .	47
3.5 Simulation examples . . . . .	49
3.6 Summary . . . . .	50
4 Norm Optimal ILC . . . . .	53
4.1 A review of earlier work on optimality and ILC . . . . .	53
4.1.1 Gradient methods for discrete-time plants . . . . .	54
4.1.2 Soft constraints in the control magnitude . . . . .	55
4.1.3 Steepest-descent method for continuous-time systems . . . . .	55
4.1.4 Newton-Raphson method . . . . .	56
4.1.5 Constrained optimisation and ILC . . . . .	57

4.2	Derivation and Convergence Analysis for Norm-Optimal ILC . . . . .	58
4.3	Causal implementation for continuous-time systems . . . . .	62
4.4	Causal implementation for discrete-time systems . . . . .	64
4.5	Predictive Norm Optimal Iterative Learning Control . . . . .	64
4.6	Derivation of the error evolution equations . . . . .	66
4.7	Partial orderings for the learning operators . . . . .	68
4.8	Almost geometric convergence . . . . .	72
4.9	A causal implementation of the predictive algorithm . . . . .	74
4.10	A numerical example . . . . .	75
	4.10.1 The effect of $\alpha_i$ . . . . .	75
	4.10.2 The effect of weighting parameter $\lambda$ . . . . .	77
	4.10.3 Effect of prediction horizon $n$ . . . . .	78
4.11	Summary . . . . .	78
5	Parameter Optimal ILC . . . . .	83
	5.1 The feedforward algorithm . . . . .	84
	5.2 Predictive algorithm . . . . .	86
	5.3 Conditioning and positivity . . . . .	88
	5.4 Simulation examples . . . . .	91
	5.5 A time-varying adaptive algorithm . . . . .	93
	5.6 Feedback implementation of the algorithm . . . . .	99
	5.7 Simulation examples . . . . .	100
	5.8 Summary . . . . .	101
6	Basis functions and HOILC . . . . .	104
	6.1 Motivation and derivation of the optimal high-order algorithm . . . . .	105
	6.2 Convergence analysis . . . . .	107
	6.3 Simulations . . . . .	110
	6.3.1 A positive system . . . . .	110
	6.3.2 A non-positive system . . . . .	112
	6.4 Summary . . . . .	116
7	A new robust steepest-descent algorithm . . . . .	119
	7.1 A modified steepest-descent algorithm . . . . .	121
	7.2 Robustness analysis . . . . .	122
	7.3 A connection between NOILC and adjoint algorithm . . . . .	124
	7.4 Simulation examples . . . . .	126
	7.4.1 The effect of $w$ on convergence speed . . . . .	126
	7.4.2 An example of increased robustness . . . . .	126
	7.5 Summary . . . . .	128
8	Repetitive Control . . . . .	131
	8.1 Problem definition and earlier work . . . . .	131
	8.2 A fundamental problem caused by implementation . . . . .	134
	8.3 A new algorithm for discrete-time repetitive control . . . . .	135
	8.4 Simulation examples . . . . .	139
	8.5 Summary . . . . .	141
9	Conclusions and future work . . . . .	147
	9.1 Overview . . . . .	147

9.2	Analysis and design of discrete-time ILC systems . . . . .	148
9.3	NOILC and predictive extensions . . . . .	148
9.4	Parameter Optimal algorithms . . . . .	149
9.5	High-Order algorithms . . . . .	150
9.6	A robust steepest-descent algorithm . . . . .	150
9.7	Repetitive Control . . . . .	151
9.8	Directions for future research . . . . .	151
	Bibliography . . . . .	153



# 1 Introduction

## 1.1 What is new in ILC and RC?

In this section classical feedback control is quickly reviewed, and the major differences between classical feedback control and Iterative Learning Control are discussed. As a starting point in feedback control a model

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t), & x(0) &= x_0 \\ y(t) &= Cx(t) + Du(t)\end{aligned}\tag{1.1}$$

is given, which describes the dynamical behaviour of a given system. In (1.1)  $A, B, C$  and  $D$  are matrices of appropriate dimensions, and  $u(\cdot)$  in the input variable or manipulated variable,  $x(\cdot)$  is the state variable,  $x_0$  the initial condition for  $x(\cdot)$ , and  $y(\cdot)$  is the output variable. In this thesis it is assumed that  $D = 0$  because it is extremely rare in physical systems that the input  $u(t)$  would affect the output  $y(t)$  without a transient. Note that this state-space representation can be written also in the input-output form

$$y(t) = [Gu](t) + z_0(t)\tag{1.2}$$

where the forced response is given by

$$[Gu](t) = C \int_0^t e^{A(t-\tau)} Bu(\tau) d\tau\tag{1.3}$$

and the unforced response by

$$z_0(t) = Ce^{At}x_0\tag{1.4}$$

The design of a controller for the dynamical system (1.1) is typically divided into two different design problems: the first design problem is a regulation problem

which consists of finding a control law that manipulates the input variable  $u(t)$  so that the system automatically holds the output  $y(t)$  at a constant value, even when unknown disturbances try to move  $y(t)$  away from this constant set point. The second design problem is a tracking problem where the objective is to make  $y(t)$  to follow a given reference signal  $r(t)$  by manipulating the input variable  $u(t)$ . The design for these two type of control systems has been solved efficiently in both classical and modern control theory by using *feedback control*: the idea is to measure the output  $y(t)$  of the system, and based on the difference between the reference  $r(t)$  and  $y(t)$ , the control input  $u(t)$  is changed according to some given rule so that the difference between  $r(t)$  and  $y(t)$  is reduced. The crucial point in design is then to find this 'magical rule', or more commonly, the control algorithm, that will keep the tracking error  $e(t) := r(t) - y(t)$  as small as possible. A lot of research effort has been put into this problem, and nowadays there exist a great many different algorithms, ranging from classical PID-control to more advanced methods such as adaptive control and robust control, that will solve this design problem. Furthermore, these design methods have been used with great success in practical applications, examples being cellular telephones, oil refineries, jumbo jets and washing machines, which clearly demonstrates the importance of feedback control in a modern society.

### 1.1.1 Fundamentals of Iterative Learning Control

Iterative Learning Control due to its special problem definition, allows more flexibility in the control system design than feedback control systems. In ILC the dynamical model is exactly the same as in (1.1), but the system (1.1) is defined only over a finite time-interval  $t \in [0, T]$ . Furthermore, a reference signal  $r(t)$  is given, and the system (1.1) has to track this reference signal as accurately as possible, resulting in a classical tracking problem over a finite time-interval. The assumption that differentiates ILC from standard feedback control is the following: when the system (1.1) has reached the final time point  $t = T$ , the state  $x$  of the system is (1.1) reset to  $x_0$ , and after resetting the system is supposed to track again the *same* reference signal  $r(\cdot)$ . At first sight this assumption might look artificial, but many important industrial applications do indeed fit into the ILC framework. Reported applications of ILC include robotics Zilouchian (1994), chemical batch processing Lee *et al.* (1996a), and servo systems Lee & Lee (1993), to name a few.

An illustrative example of a system that satisfies the ILC assumptions is a welding robot in car manufacturing. The task for the robot manipulator consists of following a given geometric trajectory and welding at specific points along the trajectory, which is the tracking part in the ILC problem definition. After the robot has finished welding the car, the robot is reset to the starting point of the trajectory and a new car of exactly of the same dimensions as the previous car is delivered for welding, which corresponds to resetting in the ILC problem definition. After the new car has arrived, the robot carries out the same trajectory tracking

and welding task on this new car as it did with the previous car, resulting in a repetition of the tracking and welding task.

In the past (and presumably still in most of the cases nowadays, at least in the industry), the control of this kind of robot (or any other system satisfying the ILC assumptions) is set up once. In other words the control action  $u(t)$  is determined from some form of fixed feedback control, which results in a control action  $u(t) = u_f(t)$ . The problem, however, is that the feedback controller will produce the same input function  $u_f(\cdot)$  during every repetition, and hence if the corresponding output function  $y_f(t) = [Gu_f](t)$  is not equal to  $r(t)$  for each  $t \in [0, T]$ , the resulting *nonzero* tracking error  $e_f(t)$  is repeated during each repetition. Consequently in Arimoto *et al.* (1984) it was suggested that one could use information from the previous repetitions to come up with a new input function  $u_k$ , where  $k$  is the repetition number, so that the tracking error will go to zero as the number of repetitions increased. In summary experience from previous repetitions or iterations is used such that the ILC system will gradually learn the control action that will result in perfect tracking, which intuitively sounds very appealing. Therefore in the robot example the robot manipulator would learn by itself the control action that gives perfect tracking, resulting in an autonomous system capable of manufacturing high quality products.

**Example 1.1** *To make this statement more concrete, consider a dynamical system*

$$(p^2 + 5p + 6)y(t) = (p + 1)u(t) \quad (1.5)$$

where  $p := \frac{d}{dt}$  and  $t \in [0, 6]$ . This system is sampled at intervals of  $T_s = 0.1$  seconds using zero-order hold resulting in a discrete-time plant model  $(\Phi, \Gamma, C)$ . The system is supposed to track a reference signal  $r(t) = \sin(2\pi t/6)$  and the system is controlled with a PID-controller

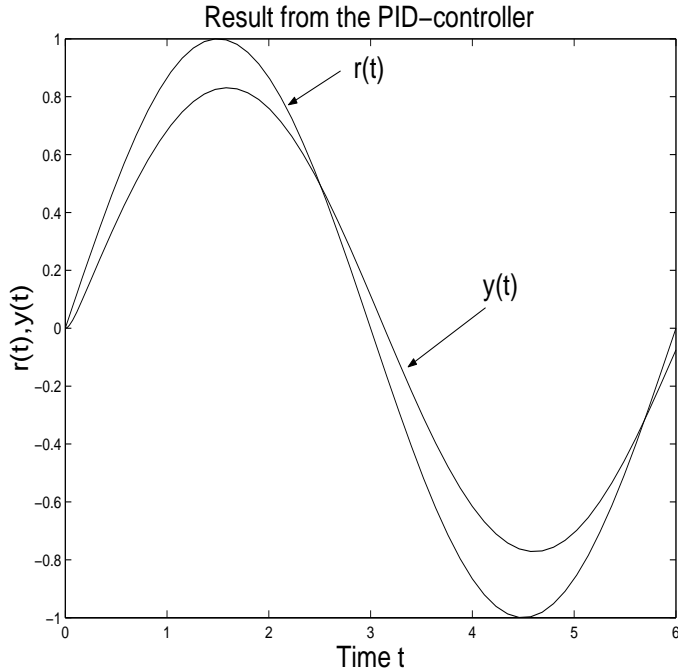
$$u(t) = K_p e(t) + K_I \int_0^t e(\tau) d\tau + K_D \dot{e}(t) \quad (1.6)$$

where  $K_p = 15$ ,  $K_I = 8$  and  $K_D = 0$ , which is also sampled using zero-order hold with the same sampling time. Fig. 1.1 shows the output  $y(t)$ , implying that the system is only capable of tracking the reference signal to a moderate degree of accuracy. Note that this same tracking result is obtained during each repetition, because the PID-controller has fixed parameters. Fig. 1.2, on the other hand, shows the  $l_2$ -norm of tracking error  $e_k(t) := r(t) - y_k(t)$  as a function of the iteration round  $k$  with the ILC algorithm

$$u_{k+1}(t) = u_k(t) + \gamma e_k(t + 0.1) \quad (1.7)$$

where  $\gamma = 9$ . This algorithm is analysed in detail in Chapter 3. Note that at first sight this algorithm seems to be non-causal, because  $u_{k+1}(t)$  is a function of  $e_k(t + 0.1)$ . However, because  $e_k(t)$  is available for  $t \in [0, T]$  for iteration  $k + 1$ , it is possible (and typically necessary) to make  $u_{k+1}(t)$  to be a function of  $e_k(s)$  for  $s \geq t$ . Based on the Fig. 1.2 it seems that the tracking error tends to zero as  $k \rightarrow \infty$ , but the convergence is not monotonic. In Chapter 2 and 3 it will be

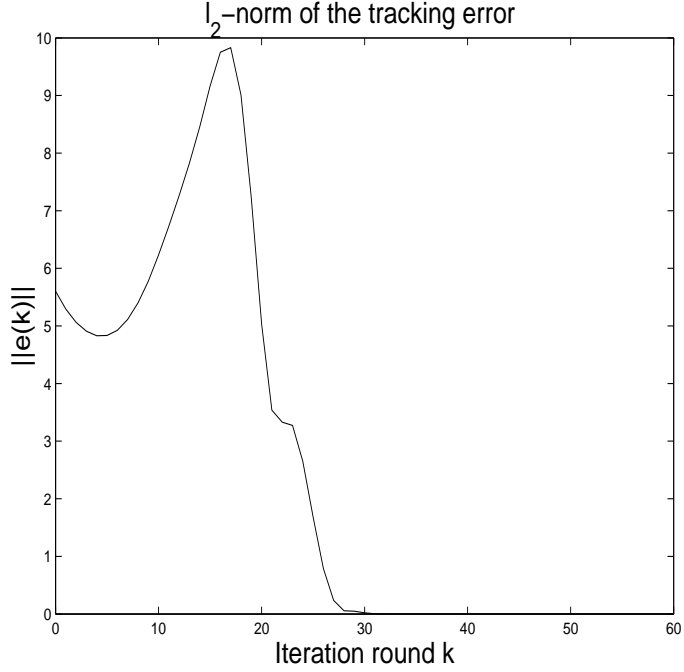
shown that this algorithm converges in limit to zero tracking error for an arbitrary time-invariant linear plant (assuming that  $CT \neq 0$ ) if the learning gain  $\gamma$  satisfies the inequality  $|1 - \gamma CT| < 1$ .



**Figure 1.1.** The response for the PID-controller (1.6).

Even though the possibility for using information from previous iterations sounds very appealing, it makes the convergence analysis of ILC systems more demanding when compared to standard feedback systems. This is because the resulting closed-loop system is in fact two-dimensional where the output of the system depends on both the infinite iteration axis  $k$  and the finite time axis  $t$ . More will be said on this in Chapter 2 and 3.

**Remark 1.1 (Who did it first?)** Note that in the ILC community it is now widely accepted that Uchiyama (1978) is the first publication to introduce the ILC concept. However, because this publication is written in Japanese, non-Japanese researchers were not aware of this publication when the ILC research initially started in USA and Western Europe. Therefore, the publication Arimoto et al. (1984) was referenced as being the starting point for ILC research (this happens quite often even nowadays). However, it seems that there are earlier publications



**Figure 1.2.** The response for the NOILC algorithm.

on ILC than Uchiyama's. In Cryer et al. (1976) the authors (who worked for General Motors Truck and Bus) proposed the following discrete-time 'iterative control' (as they call it) in the frequency domain

$$u_{k+1}(e^{j\omega T_s}) = u_k(e^{j\omega T_s}) + \beta G^{-1}(e^{j\omega T_s})e_k(e^{j\omega T_s}) \quad (1.8)$$

where  $\beta \in \mathbb{R}, 0 < \beta < 1$  is a design parameter, and  $G(e^{j\omega T_s})$  is the discrete-time Fourier transform of the impulse response  $g(t)$  of the plant in question. This is clearly an ILC algorithm and the authors apply this algorithm in the context of laboratory road simulation. As a first step in the simulation the car is driven along a certain road profile, and the accelerations caused by the road profile on the car are measured and recorded. After that the car is inserted on a test rig, which comprises hydraulic manipulators. These hydraulic manipulators are used to replicate these accelerations on the car as accurately as possible. As a result the same road profile can be repeated on the car as many times as necessary without a human driver, speeding up considerably durability tests on the different mechanical parts of the car. In order to find the control signals for the hydraulic manipulators that replicate the accelerations as accurately as possible, the 'learning process' (1.8) is used where  $G(e^{j\omega T_s})$  is now a dynamical model of the car. The algorithm (1.8) has in fact become the most common control algorithm in laboratory road simulations, see for

example Cuyper et al. (1999), and is used in several commercial test rigs for road simulation. For an excellent review on road simulations see Dodds & Plummer (2001).

Another even earlier reference to ILC seems to be the USA patent 'US3555252 - Learning Control of Actuators in Control Systems' from 1971, see also Chen & Moore (2000). The patent proposes

*A learning control technique which learns the characteristics of actuators, such as electric units, and provides actuator signals based on them. The command signal, which is related to the actuator characteristics, and from which the actuator signal is derived, is stored in the computer memory and modified by an amount related to a function of the error between the actual movement and the desired movement of the actuator. A specific application of learning control to both electric drive units and pneumatic actuators is described.*

*This description clearly fits into the framework of ILC, and is currently the oldest publication on ILC that the author is aware of.*

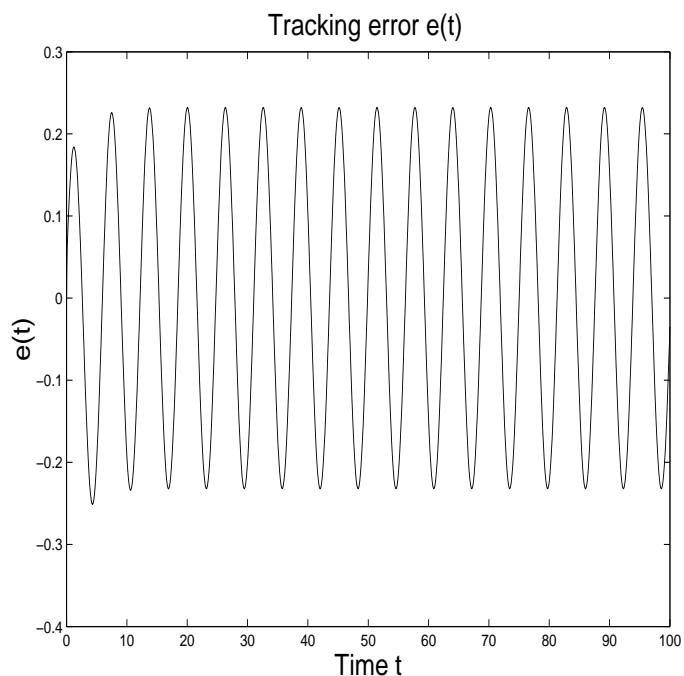
### 1.1.2 Fundamentals of Repetitive Control

In Repetitive Control, the starting point is the plant model (1.1), which is defined, as in standard feedback control, over the infinite-time interval  $t \in [0, \infty)$ . Furthermore, the system is supposed to track a  $T$ -periodic reference signal  $r(t)$ , i.e.  $r(t) = r(t + T)$ , but no other *a priori* information is assumed to be available for control algorithm design. There exists a lot of important applications that fit into this framework and reported applications of RC can be found in robotics (Kaneko & Horowitz (1997)), motors (Kobayashi *et al.* (1999)), rolling processes (Garimella & Srinivasan (1996)), hard-disc control (Smith *et al.* (1999)), peristaltic pumps (Hillerström & Sternby (1994)), general motion control (Tomizuka (1992)), PWM converters (Zhou & Wang (2001)) and rotating mechanisms (Fung *et al.* (2000)). Repetitive control has also been applied to active vibration and noise cancellation problems in Hillerström (1996), which is currently a very active research topic in the control community.

Note that RC problem setting is very similar to the ILC case: the only difference is that whereas in ILC the state of the system is reset to  $x_0$  at the end of the period, in RC the initial condition for each period is the final state from previous period. The fact that the reference signal  $r(t)$  is periodic, opens up again possibilities for learning strategies. As the reference signal is the same for each period, one can use information from previous periods to modify the input  $u(t)$  so that eventually the system will learn the input signal that gives the desired periodic solution.

**Example 1.2** *As an example of RC, consider again the continuous-time model (1.5), which is now defined over the infinite-time interval  $t \in [0, \infty)$ . Furthermore, the system is supposed to track a reference signal  $r(t) = \sin(2\pi t/6)$ . Fig. 1.3 shows the tracking result in terms of  $e(t)$  when system is controlled with PID-algorithm*

(1.6) with same parameters as in Example 1.1, resulting in a rather poor tracking performance.



**Figure 1.3.** The tracking response for the PID algorithm (1.6).

Consider now the RC algorithm

$$u(t) = u(t - 6) + e(t) \quad (1.9)$$

Fig. 1.4 shows the tracking performance, which considerably improved when compared to the PID-controller (1.6). Based on this figure it seems that tracking error  $e(t)$  tends to zero as  $t \rightarrow \infty$ . Whether or not this is true will be discussed in Chapter 8, where this algorithm is analysed in more detail.

Note, that even though the RC algorithm seems to perform better than the PID-algorithm, the stability analysis of this algorithm is more complicated. This is due the fact the algorithm (1.9) contains a delay-line, and hence it is infinite-dimensional, resulting in an infinite-dimensional closed-loop system.

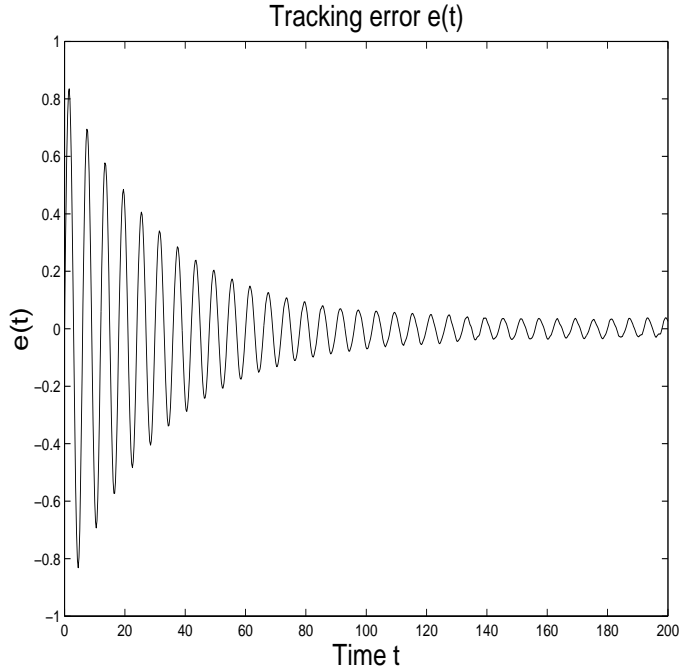


Figure 1.4. The tracking response for the RC algorithm (1.9).

## 1.2 A formal definition of Iterative Learning Control

To make a precise mathematical definition of the ILC problem consider the following standard continuous-time time-varying linear state-space model defined over a *finite* time-interval  $t \in [0, T]$ :

$$\begin{aligned} \dot{x}(t) &= A(t)x(t) + B(t)u(t), & x(0) &= x_0 \\ y(t) &= C(t)x(t) \end{aligned} \quad (1.10)$$

or the corresponding linear time-varying discrete-time system with sampling time  $T_s = h$

$$\begin{aligned} x(t+h) &= \Phi(t)x(t) + \Gamma(t)u(t), & x(0) &= x_0 \\ y(t) &= C(t)x(t) \end{aligned} \quad (1.11)$$

where the state  $x(\cdot) \in \mathbb{R}^n$ , output  $y(\cdot) \in \mathbb{R}^m$ , input  $u(\cdot) \in \mathbb{R}^m$ . The operators  $A(\cdot), \Phi(\cdot), B(\cdot), \Gamma(\cdot)$ , and  $C(\cdot)$  in (1.10) and (1.11) are matrices of appropriate dimensions. Note that in (1.10) time  $t$  is a continuous parameter whereas in (1.11) it only takes discrete values  $t = 0, h, 2h, \dots, T$ . In order to avoid technical difficulties in analysis, it is typically assumed that matrices are continuous with respect to time  $t$ . In addition, a reference signal  $r(t)$  is given and the task is to construct

$u(t)$  so that  $y(t)$  would track  $r(t)$  as accurately as possible. The system (1.10) (or the corresponding discrete-time model (1.11)) is supposed to follow the reference signal in a repetitive manner, i.e. after the system has reached the final time point  $t = T$ , the state of the system is reset to the initial condition  $x_0$  and the system is supposed to track the same reference signal  $r(t)$  again. If  $u_k(t)$  is the input applied at trial  $k \in \mathbb{N}$  and  $e_k(t) := r(t) - y_k(t)$  is the resulting tracking error, construct a control law

$$u_{k+1}(t) = f(e_{k+1}(\cdot), e_k(\cdot), \dots, e_{k-s}(\cdot), u_k(\cdot), u_{k-1}(\cdot), \dots, u_{k-r}(\cdot)) \quad (1.12)$$

so that  $\lim_{k \rightarrow \infty} u_k = \tilde{u}$  and  $\lim_{k \rightarrow \infty} e_k = 0$  in a suitable topology. Furthermore, for causality it is required that  $u_{k+1}(t)$  is a function of  $e_{k+1}(s)$  for  $s \leq t$ . Note that in the problem definition it is assumed that there exists an input  $u^*$  which gives perfect tracking. If this is not the case, the problem can be modified in the following manner: the algorithm should converge to a fixed point  $u^*$  where  $u^*$  is the solution of the optimisation problem

$$u^* = \arg \min_{u \in \mathcal{U}} \|r - Gu\|^2 \quad (1.13)$$

where  $\mathcal{U}$  is set of possible inputs and  $\|\cdot\|$  is a suitable norm. The original and modified problem definition generate at least the following two fundamental design questions that have to be asked before any further algorithm design can take place:

- Is the algorithm design done with the continuous-time system (1.10) or with the discrete-time model (1.11)?
- What topology (norm)  $\|\cdot\|$  is used in the algorithm design?

The first question is in fact a very subtle one: because in ILC the control algorithm uses information from previous repetitions, this information has to be recorded on a suitable media. However, this is only possible with digital devices<sup>1</sup>, and hence it seems to be natural to use the discrete-time model (1.11). However, as will be seen in future chapters, convergence results obtained for discrete-time models can be sometimes over-optimistic and hide problems with robustness when the discrete-time algorithm is applied on a plant that is originally a continuous-time system. Thus it might be argued that the continuous-time model (1.11) should be used in control design. A counterexample against this strategy will be given in Chapter 8: analysis in this chapter shows how the discretisation of a continuous-time RC algorithm (which is required if the control algorithm is implemented with a digital computer) will destroy convergence, and consequently design in continuous-time can also give over-optimistic results. This discussion indicates that, whatever model type is chosen, the theoretical convergence results have to be always checked with a simulation model which takes into account the hybrid nature of the problem, i.e. the controller is a discrete-time system and the plant in is a continuous-time system.

The answer to the second question could be the following one: select a normed space that is easy to work with. For example in this thesis  $L_2[0, T]$  and  $l_2[0, T]$  are

---

<sup>1</sup>Strictly speaking, the recording could be done with analogue components, but the resulting set-up is impractical due to high cost

used exclusively. This is due the fact that both spaces are complete inner-product spaces, and optimisation in these spaces is far more simpler than in a general Banach space setting, resulting in the rather complete theory of Norm-Optimal Iterative Learning Control.

Convergence is naturally the most important requirement for an ILC algorithm. However, additional requirements have been suggested in the research literature (see for example Bien & Xu (1998)), one of the most common ones being

- i) Convergence should be achieved with a minimal amount of information about the plant.
- ii) Convergence should be achieved even if there is uncertainty in the plant model.
- iii) Convergence should be achieved even if the resetting is not perfect, i.e. the initial condition for each iteration lies inside a ball of radius  $\delta$  centred at  $x_0$ , or in more mathematical notation,  $x_k(0) \in B_\delta(x_0) \in \mathbb{R}^n$  for  $k \in \mathbb{N}$ .

Note that the first additional requirement is not always sensible. This is due to the fact that for example in robotics either accurate models are available from the robot manufacturer, or they can be obtained rather easily using modern identification techniques, and it would be unwise to discard this information about the plant model in the ILC algorithm design.

As a final remark on models note that both models (1.10) and (1.11) are linear. This seems to be rather restrictive, because nonlinear models are quite common in robotics and chemical engineering. However, because in ILC the plant is defined only over a finite time-interval, according to Amann *et al.* (1998) it can be argued that a time-varying linear model can be used to approximate the original nonlinear plant around the operating point  $(u^*(t), r(t))$  with adequate accuracy.

### 1.3 A formal definition of Repetitive Control

As in Iterative Learning Control the starting point in Repetitive Control is a linear time-invariant continuous-time model

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), & x(0) &= x_0 \\ y(t) &= Cx(t) \end{aligned} \tag{1.14}$$

or a discrete-time model

$$\begin{aligned} x(t+h) &= \Phi x(t) + \Gamma u(t), & x(0) &= x_0 \\ y(t) &= Cx(t) \end{aligned} \tag{1.15}$$

that models the behaviour of the original plant. In this equation the state  $x(\cdot) \in \mathbb{R}^n$ , output  $y(\cdot) \in \mathbb{R}^m$ , input  $u(\cdot) \in \mathbb{R}^m$ . Both systems are defined over an infinite-time interval  $t \in [0, \infty)$  where in the discrete-time case  $t = kh$ ,  $k = 0, 1, 2, \dots$  and  $h$  is the sampling-time.  $A, \Phi, B, \Gamma$  and  $C$  are matrices of appropriate dimensions. Nonlinear models could be also considered, but in order to keep the analysis fairly simple, in this thesis they are omitted. The control design problem is to design a

feedback controller so that the system (1.14) or (1.15) would track a  $T$ -periodic reference signal  $r(t)$ , i.e.  $r(t) = r(t + T)$ , so that

$$\lim_{t \rightarrow \infty} e(t) = 0 \quad (1.16)$$

where  $e(t) := r(t) - y(t)$ . Furthermore, in the control law it is possible to use information from previous periods, i.e. the RC-algorithm is of the form

$$u(t) = f(u(t - T), u(t - 2T), \dots, u(t - mT), e(t), e(t - T), \dots, e(t - nT)) \quad (1.17)$$

for some  $m, n \in \mathbb{N}$ . Additional requirements could be that

- i) Convergence should be achieved even if there is uncertainty in  $T$ .
- ii) Convergence should be achieved in the presence of model uncertainty in (1.14) and (1.15).

Note that especially in servo systems it is very common that the reference signal is not periodic with respect to time  $t$ , but, rather, with respect to the angular position of servo system. As is shown in Mahawan & Luo (2000), under suitable assumptions this problem can be also tackled with modified RC, where the independent variable is no longer time  $t$  but the angular position.

## 1.4 Contributions

This monograph is based on the following publications:

- [P1] Owens, D.H. Hättönen, J. (2002) Convex modifications to an iterative learning control law. In Proc. of the 15th IFAC World Congress on Automatic Control. Barcelona, Spain. Accepted for publication in *Automatica*.
- [P2] Hättönen J., Owens, D.H and Moore K.L. (2004) An Algebraic Approach to Iterative Learning Control. *International Journal of Control* 77(1):45-54.
- [P3] Hättönen J., Owens D.H. and Feng K. (2003) New connections between positivity and Parameter-Optimal Iterative Learning Control. In Proc. of the 18th IEEE International Symposium on Intelligent Control. Houston, USA.
- [P4] Hättönen J., Feng K. and Owens D.H. Basis functions and Parameter Optimisation in High-Order Iterative Learning Control. Submitted to *Automatica*.
- [P5] Hättönen J., Owens D.H. and Ylinen R. (2003) A new optimality based Repetitive Control algorithm for discrete-time systems. In Proc. of the European Control Conference (ECC). Cambridge, UK.
- [P6] Hättönen J., Harte T., Owens D.H., Ratcliffe J., Lewin P. and Rogers, E. (2003) A new robust Iterative Learning Control algorithm for application on a gantry robot. In Proc. of the IEEE conference on Emerging Technologies in Factory Automation. Lisbon, Portugal.

Publication P1 shows how an existing Norm-Optimal Predictive ILC algorithm can be modified so that the resulting convergence speed is considerably improved when compared to the existing algorithm. The original idea for modifying the

algorithm came from David Owens. However, the theoretical analysis in the paper was done by the author of this thesis. Chapter 4 is based on this publication and the earlier work of Amann *et al.* (1996).

Publication P2 introduces a general algebraic framework for discrete-time ILC systems. The results presented in the paper rely on earlier results derived by Kevin Moore and David Owens. The main contribution in this paper from the author of this thesis was to combine and formalise the results of Moore and Owens, resulting in a rather general convergence theory of discrete-time ILC systems. In addition, in this publication he makes the important observation that typically time-varying ILC algorithms are needed in order to achieve acceptable convergence behaviour. Therefore a new optimality based adaptive ILC algorithm is also derived in this paper. The idea for this algorithm came from two different sources: The algebraic approach suggested that time-varying algorithms are needed in order to guarantee good transient behaviour. However, it was not clear how to select the design parameters in a time-varying ILC algorithm. Hence he extended the so-called Parameter-Optimal ILC approach by Kevin Feng and David Owens to cover the time-varying ILC algorithm. As a result monotonic convergence is achieved for an arbitrary time-invariant linear plant, demonstrating in a very concrete manner the feasibility of this algorithm. Chapter 3 presents the algebraic analysis carried out in this publication. Sections 5.5-5.7 in chapter 5 present this algorithm.

Publication P3 establishes new connections between positivity and Parameter-Optimal ILC. In this publication the work by Feng and Owens in Owens & Feng (2002) and Owens & Feng (2003) on feedforward ILC is extended, and it is shown that a predictive Parameter-Optimal ILC algorithm will result in geometric convergence, if the original plant satisfies positivity condition. Furthermore, as a new result it is shown that if the original plant is not positive, under suitable assumptions there exists a predictive feedback controller that ‘conditions’ the original plant so that it becomes positive. The author of this thesis was responsible for the theoretical developments in this paper. Sections 5.1 - 5.4 in Chapter 5 are based on this publication.

Publication P4 introduces a class of Parameter-Optimal high-order (POILC) algorithms. In POILC the flexibility of the algorithm is increased by adding more inputs and tracking errors from the previous iterations. However, convergence analysis shows that for convergence to zero tracking error it is required that the original plant satisfies the same positivity condition as the corresponding first-order time-invariant POILC algorithms. In order to relax the positivity assumption a set of basis functions is added to the algorithm. Subsequent analysis shows that if the basis functions are chosen appropriately, the positivity condition is no longer required for zero convergence. The author of this thesis came up with the idea of using basis functions. Furthermore, he was responsible for doing the simulation studies presented in the paper. Chapter 6 is based on this publication and the earlier work of Owens & Feng (2002).

Publication P5 discusses the importance of positive-real systems in RC. In this publication the author is able to prove that a well-known continuous-time RC algorithm that converges for continuous-time positive-real systems becomes unstable, if the algorithm is discretised. Discretisation, however, is always needed

due to implementation issues discussed in Section 1.2. Consequently positive-real systems in RC seem to be less important than was previously thought. Therefore in publication P5 a new optimality based RC algorithm is derived for discrete-time systems. The algorithm was developed by the author of this thesis together with Raimo Ylinen and David Owens. This algorithm will result in asymptotic convergence for an arbitrary linear time-invariant plant under mild controllability and observability conditions. Chapter 8 is an extended version of this publication.

Publication P6 can be seen as a starting point for robustness analysis for discrete-time Iterative Learning Control. In this paper, a modification of the steepest-descent algorithm is proposed. Theoretical analysis of the algorithm shows that if a tuning parameter in the algorithm is selected to be sufficiently large, the algorithm will result in monotonic convergence if the plant uncertainty satisfies a positivity condition. This is a major improvement when compared to the standard steepest-descent algorithm, which lacks a mechanism for finding a balance between convergence speed and robustness. In the paper the algorithm is applied on an industrial-scale multi-axis gantry robot. Experimental runs on this system demonstrate that the algorithm results in near perfect tracking after 300 iterations with a poor initial guess for the input function. The author of this thesis was responsible for the original idea of modifying the standard algorithm and deriving the theoretical results in this paper. In addition, he developed an optimisation software to fit the transfer function models to the experimental data. Chapter 7 is an extended version of this publication.

In addition to the work reported in this section, during the last three years the author of this thesis has investigated together with V. Hatzikos and D. Owens how to use genetic algorithms to implement the Norm-Optimal algorithm for non-linear plants. This work has been published in Hatzikos *et al.* (2003c), Hatzikos *et al.* (2003a), Hatzikos *et al.* (2003b) and Hatzikos *et al.* (2004). This material, however, is not included in this thesis in order to restrict the length of the monograph. The research work on Iterative Learning Control and Repetitive Control has also resulted in a library of MATLAB<sup>2</sup> functions. This library is not reported in this study, again due to space limitations. However, in the near future it will be made available to other researchers working on Iterative Learning Control and Repetitive Control.

## 1.5 Organisation of the thesis

The remainder of the thesis is divided into seven chapters. The beginning of each chapter contains a short introduction to the main results presented in the chapter, and each chapter ends with a summary that also links the results of the chapter to other chapters. Consequently the reader should be able to obtain a reasonable understanding of the contents of this thesis by glancing through the introduction and summary sections in each chapter. Chapters 3 to 8 also contain simulation studies, which were done using the Matlab software package. The purpose of these

---

<sup>2</sup>Matlab is a registered trademark of The Mathworks, Inc, 1994-2003

simulations is to illustrate how the theoretical results in the chapter can be seen in the convergence behaviour of numerical examples.

Note that the thesis does not contain a review chapter on previous work on ILC. This is due the fact that for example the review paper Moore (1998a) contains roughly 300 publications on ILC, and thus it is not clear how to give a balanced review of all these algorithms in a single chapter. Therefore instead having a separate chapter on previous work, each chapter includes material that relates the results of the chapter to the earlier work on ILC and RC (if there exists a relation).

The individual chapters are organised as follows: Chapter 2 is an introductory chapter - the main objective is to introduce to the reader the mathematical tools that are used in the rest of the chapters. The main theme running through this chapter is the rather abstract iteration process  $x_{k+1} = Tx_k$  and its mathematical properties. A particular emphasis is put on the convergence behaviour of this process, i.e. when does the sequence of vectors  $\{x_k\}$  converge to some well-defined vector  $x_\infty$  as the iteration index  $k$  goes to infinity? Both a contraction mapping and a spectral radius condition is established in order to answer this question. Furthermore, the spectral radius condition is analysed in detail when  $T$  is a dynamical system. Finally, these two tests are extended to cover high-order iterative processes of the form  $x_{k+1} = T_1x_k + T_2x_{k-1} + \dots + T_nx_{k-n}$ .

Chapter 3 analyses the algebraic structure of ILC systems in the discrete-time case. In this chapter it is shown how in the discrete-time case the original 2-D ILC problem can be converted to an equivalent one-dimensional multivariable control problem, where the objective is to make the output of the multivariable static plant track a 'multi-channel' step function. Consequently all the standard analysis and synthesis methods from linear multivariable control can be applied to ILC systems in the discrete-time case. The one-dimensional description of the ILC problem shows that a sufficient condition for asymptotic convergence is that the controller must include an integrator along the iteration axis but at the same time the resulting closed-loop system should be stable. This description also results in the important observation that time-varying algorithms should be used rather than time-invariant ones to guarantee that the norm of tracking error does not grow excessively during earlier iteration rounds.

After analysing the algebraic properties of ILC systems the next step is to apply optimisation within ILC. The topic of Chapter 4 is Norm-Optimal ILC (NOILC). As a starting point NOILC is related to other work in the research literature on optimality and ILC. Then, the convergence analysis for NOILC and its predictive extension are both provided. Finally, a new convex modification is proposed for the predictive algorithm, and it is shown that the convex modification results in considerable improvement in terms of convergence speed when compared to the standard predictive algorithm.

Chapter 5 concentrates on Parameter-Optimal ILC (POILC) algorithms. In this chapter it shown how relatively simple adaptive time-invariant POILC algorithms can result in monotonic convergence to zero tracking error, if the original plant satisfies a suitable positivity condition. Furthermore, if the original system is not positive, it is shown that under certain assumptions for the original system, a predictive feedback controller loop can be added into the original system so that

the overall system becomes positive. This broadens further the class of dynamical systems that can be approached with POILC. As a next step a new POILC algorithm is introduced where the learning gain is both time- and iteration-dependent. Subsequent analysis shows that this algorithm will result in monotonic convergence to zero tracking error for an arbitrary discrete-time LTI plant, which is a very strong property for any ILC algorithm.

In Chapter 6 a high-order POILC algorithm is proposed, where information from both previous inputs and tracking errors is used to generate the current control action. It turns out that despite the added flexibility, the original plant model has to satisfy the same positivity condition as the time-invariant POILC algorithms in order to guarantee zero tracking error in the limit. A possible explanation for this is collinearity, i.e. during terminal convergence the previous tracking errors and inputs point into the same direction, and therefore near convergence the high-order algorithm behaves in a similar manner as the first-order, time-invariant POILC algorithms. In order to rectify this problem a set of fixed basis functions is added into this algorithm, and it is shown that if the basis functions are chosen in a correct way, the resulting tracking error will converge monotonically to zero. This is theoretically a major improvement with respect to the original time-invariant POILC algorithms because the addition of basis functions removes the rather restrictive positivity condition on the plant model.

Chapter 7 shows how an adaptive modification of a standard steepest-descent algorithm in the ILC case enhances the robustness properties of the standard algorithm. If the plant model contains multiplicative uncertainty, and the uncertainty satisfies a positivity condition, the tracking error will converge monotonically to zero tracking error. The convergence also requires that a tuning parameter in the algorithm is selected to be sufficiently large. As a final step in this chapter, an interesting connection between the adaptive steepest-descent algorithm and the NOILC algorithm is established.

Chapter 8 is the only chapter in this thesis concerning Repetitive Control (RC). It is shown that a well-known continuous-time algorithm that has been studied extensively in the research literature can diverge if it is implemented using sampled data processing. Furthermore, it is demonstrated that the reason for divergence is that sampling destroys the positive-realness property of the continuous-time plant (which is the necessary condition for convergence in the continuous-time case). In order to remove this divergence problem, a new optimality based RC algorithm is derived purely in a discrete-time setting. Convergence analysis carried out in this chapter shows that this algorithm results in asymptotic convergence for an arbitrary LTI discrete-time plant, which is at least theoretically a very strong and desirable property.

Chapter 9 summarises the work presented in this thesis and gives directions for future research.

## 2 Convergence theory for iterative processes

As was explained in the previous chapter, the basic idea behind ILC is to use the repetitive nature of the problem definition to allow the system to learn the input function that results in perfect tracking. The learning mechanism, however, introduces a new axis: namely the iteration axis  $k$ . This results in a two-dimensional system, where the independent variables are the finite time axis  $t \in [0, T]$  and the infinite iteration axis  $k \in \mathbb{N}$ . As a first observation towards convergence/stability analysis, note that due to the finite nature of the time axis, the output of a finite-dimensional linear time-varying system can never become unbounded in finite time. Hence, by contrast to classical feedback control, it can be expected that the properties of the ILC system along the time-axis do not play a major role in convergence analysis. The iteration axis, on the other hand, is infinite. Therefore as the number of iterations increases it is intuitive that the ‘output profile’  $y_k(t)$  for  $t \in [0, T]$  can either converge or diverge, depending on the chosen learning mechanism. How can the question of convergence/stability then be approached mathematically? In answering this question the following example should be useful:

**Example 2.1** *Consider the following ILC algorithm*

$$u_{k+1}(t) = u_k(t) + [Ke_k](t) \quad (2.1)$$

where  $K$  is a ‘learning-gain operator’, and the plant model

$$y_{k+1}(t) = [Gu_{k+1}](t) + z_0(t) \quad (2.2)$$

for  $t \in [0, T]$ . In order to analyse the convergence properties of the algorithm, it is necessary to find how the tracking error  $e_k(t) := r(t) - y_k(t)$  evolves as a function of the iteration round  $k$ . In order to find this ‘evolution equation’, apply the control algorithm (2.1) to the plant model  $y_{k+1}(t) = [Gu_{k+1}](t) + z_0(t)$ , resulting in

$$[Gu_{k+1}](t) + z_0(t) = [Gu_k](t) + z_0(t) + [GKe_k](t) \quad (2.3)$$

The next step is to multiply (2.3) with  $-1$  and after that to add both  $r(t)$  on both sides of the equation. This results in

$$r(t) - [Gu_{k+1}](t) - z_0(t) = r(t) - [Gu_k](t) - z_0(t) - [GKe_k](t) \quad (2.4)$$

Using the process model (2.2) and the definition of the tracking error  $e_k(t)$  this equation can be written equivalently as

$$e_{k+1}(t) = [(I - GK)e_k](t) \quad (2.5)$$

or in more compact form

$$e_{k+1}(t) = [Le_k](t) \quad (2.6)$$

where  $L = (I - GK)$ . Hence  $L$  is the operator that maps  $e_k(\cdot)$  to  $e_{k+1}(\cdot)$ , and thus it has to be assumed that its mathematical properties somehow define whether or not the algorithm converges.

In the previous example  $L$  is the *learning operator* that maps the tracking error from the previous trial to the current trial. In fact most of the existing ILC algorithms in the research literature result in the general error evolution equation

$$e_{k+1} = Le_k \quad k = 0, 1, 2, \dots \quad (2.7)$$

or more generally

$$e_{k+1} = Le_k + b \quad k = 0, 1, 2, \dots \quad (2.8)$$

and therefore it is important to analyse in an abstract setting the conditions under which this kind of iterative process converges. Two different conditions are given for convergence in the next two sections. The first condition is a norm or a contraction mapping condition for the learning operator  $L$ , which guarantees that the tracking error converges to zero in the limit. Furthermore, if this norm condition is met, the tracking error monotonically decreases as the number of iterations increases. This is sometimes very important in practical applications. The second condition for convergence is a spectral radius condition. If the learning operator  $L$  satisfies this condition, it is guaranteed that as  $k \rightarrow \infty$ , the tracking error converges to zero, but no predictions can be made as to how the tracking error behaves during earlier iteration rounds. After introducing these two rather abstract tests it is shown how they can be applied on first-order iterative processes where the iterative process involves a dynamical system. It turns out, that due to the finite nature of the time-axis, the convergence conditions do *not* depend on the stability properties of the dynamical system along the time-axis. Furthermore, this observation can be used to design ILC algorithms that require a minimal amount of information about the system to be controlled. The final section of this chapter shows how high-order iterative processes can be approached with the convergence analysis techniques presented in the earlier sections of this chapter.

## 2.1 Contraction mapping condition

Let  $x_0$  be an arbitrary element of a normed space  $\mathcal{X}$  with a metric  $d$  and let  $T$  to be an operator  $T : \mathcal{X} \rightarrow \mathcal{X}$ . Consider now the iteration

$$x_{k+1} = Tx_k; \quad k = 0, 1, 2, \dots \quad (2.9)$$

The sequence  $\{x_k\}_{k \geq 0}$  will converge in the norm to a unique fixed point in  $\mathcal{X}$  if the two following conditions hold:

- i) The normed space  $\mathcal{X}$  is complete
- ii) The operator  $T$  is a contraction mapping, i.e. there exists  $0 < \alpha < 1$  so that

$$d(Tx, Ty) \leq \alpha d(x, y) \quad \forall x, y \in \mathcal{X} \quad (2.10)$$

The proof is standard and can be found, for example, from Pugh (2002). Note that the result is exactly the same for the modified iteration

$$x_{k+1} = Tx_k + b \quad (2.11)$$

In other words the convergence depends purely on  $T$  and the completeness of  $\mathcal{X}$ . However, the fixed point where the iteration converges to is different and is given by the equation

$$x_\infty = (I - T)^{-1}b \quad (2.12)$$

due the uniqueness of  $\lim_{k \rightarrow \infty} x_k = x_\infty$ . Because this condition is only a sufficient condition, a violation of these conditions does not necessarily imply that the iteration (2.9) (or its more general form (2.11)) would diverge.

Assume now that the metric space  $\mathcal{X}$  is in fact a complete normed space, i.e.  $\mathcal{X}$  is complete and the space is equipped with a norm  $\|\cdot\| : X \rightarrow \mathbb{R}_+$  where  $\mathbb{R}_+$  is defined to be the set of non-negative real numbers. In this case the metric  $d(x, y)$  becomes  $d(x, y) = \|x - y\|$  for an arbitrary  $x, y \in X$ . Assume further that the operator  $T$  is a linear and bounded, i.e.  $T(\alpha x) = \alpha T(x)$  and  $T(x_1 + x_2) = T(x_1) + T(x_2)$  (linearity), and there exists  $M \in \mathbb{R}$ ,  $M > 0$  so that for an arbitrary  $x \in \mathcal{X}$  it holds that  $\|Tx\| \leq M\|x\|$  (boundedness). In this case the condition (2.10) reads

$$d(Tx, Ty) = \|Tx - Ty\| = \|T(x - y)\| \leq \alpha \|x - y\| \quad (2.13)$$

Furthermore, it is a standard result for bounded linear operators that  $\|Tx\| \leq \|T\|\|x\|$  where  $\|T\|$  is the operator norm defined in the following way:

$$\|T\| := \sup_{x \in X, \|x\|=1} \|Tx\| \quad (2.14)$$

Hence the following estimate holds

$$\|Tx - Ty\| = \|T(x - y)\| \leq \|T\|\|x - y\| \quad (2.15)$$

and comparing this estimate with (2.10) it is clear that if

$$\|T\| < 1 \quad (2.16)$$

then the sequence  $x_{k+1} = Tx_k$  converges. In the ILC context it was assumed that the sequence of tracking errors satisfies  $e_{k+1} = Le_k$  where  $L$  is again the learning operator. If  $L$  is now a contraction mapping, then

$$\|e_{k+1}\| = \|Le_k\| \leq \|L\|\|e_k\| < \|e_k\| \quad (2.17)$$

if  $e_k \neq 0$ . Based on this estimate a learning operator that is a contraction mapping results in a sequence of tracking errors where the norm of each tracking error is smaller than the norm of the tracking error from the previous iteration. In a more mathematical terminology it is said that the algorithm results in monotonic convergence, and this a very desired property for an ILC algorithm. As the next example will demonstrate, whether or not a given operator is a contraction mapping is highly dependent on the norm:

**Example 2.2** Consider a matrix mapping  $L : \mathbb{R}^n \rightarrow \mathbb{R}^n$ . The space of  $\mathbb{R}^n$  can be equipped with several different norms. The most frequently used norms are  $l_1$ -norm,  $l_2$ -norm and  $l_\infty$ -norm defined, respectively, by the equations

$$\begin{aligned} \|v\|_1 &:= \sum_{i=1}^n |v_i| \\ \|v\|_2 &:= \sqrt{\sum_{i=1}^n |v_i|^2} \\ \|v\|_\infty &:= \max_{i \in I} |v_i| \end{aligned} \quad (2.18)$$

where  $I$  is the index-set  $I = \{1, 2, \dots, n\}$ ,  $v \in \mathbb{R}^n$  and arbitrary, and  $v = [v_1 \ v_2 \ \dots \ v_n]^T$ . The corresponding operator norms for  $L$  become (see Lancaster & Tismenetsky (1985))

$$\begin{aligned} \|L\|_1 &= \max_{j \in I} \sum_{i=1}^n |L_{ij}| \\ \|L\|_2 &= \sigma_{max}(L) \\ \|L\|_\infty &= \max_{i \in I} \sum_{j=1}^n |L_{ij}| \end{aligned} \quad (2.19)$$

where  $\sigma_{max}(L)$  is the largest singular value of  $L$ . Note that frequently  $\|L\|_\infty \neq \|L\|_2 \neq \|L\|_1$ , and consequently it has to be always made clear which particular norm is being used to measure the convergence properties of the algorithm.

Even though it is very desirable to require that a learning operator is a contraction mapping, in some cases of practical importance this is not the case. Hence another 'test' is required that determines when the sequence  $x_{k+1} = Tx_k$  converges asymptotically, even if the convergence is not monotonic. This test is given in following section using the spectral radius of  $T$ .

## 2.2 Spectral radius condition

Consider again the iterative process

$$x_{k+1} = Tx_k \quad k = 0, 1, 2, \dots \quad (2.20)$$

where  $T : \mathcal{X} \rightarrow \mathcal{X}$  and  $\mathcal{X}$  is assumed to be a normed space. Now define the spectral radius of  $T$  with the formula

$$\rho(T) := \lim_{n \rightarrow \infty} \|T^n\|^{\frac{1}{n}} \quad (2.21)$$

Then it can be shown (see for example Edwards & Owens (1982)) that if  $\rho(T) < 1$  the iterative process (2.20) converges to zero. This is due to the fact that formally

(i.e. the last equality is true but it has to be proved rigorously)

$$\lim_{k \rightarrow \infty} \|e_k\| = \lim_{k \rightarrow \infty} \|T^k e_0\| \leq \lim_{k \rightarrow \infty} \|T^k\| \|e_0\| = \lim_{k \rightarrow \infty} \rho(T)^k \|e_0\| \quad (2.22)$$

and if  $\rho(T) < 1$ , it follows immediately that  $\lim_{k \rightarrow \infty} e_k = 0$ , i.e. the iterative process converges *asymptotically* to zero. It is also easy to show that the spectral radius condition is both a necessary and a sufficient condition for asymptotic convergence for an arbitrary initial condition  $x_0 \in \mathcal{X}$ . However, as the next example taken from Amann (1996) illustrates, the spectral radius condition only guarantees that the norm of  $e_k$  is zero in the limit. During earlier iterations  $e_k$ s with a large norm can be generated by the iterative process (2.20), even if the initial condition  $e_0$  has a small norm.

**Example 2.3** Consider again a matrix mapping  $L : \mathbb{R}^n \rightarrow \mathbb{R}^n$ . Take  $n = 2$  and consider the matrix

$$L = \begin{bmatrix} 0 & \epsilon \\ 0 & 0 \end{bmatrix} \quad (2.23)$$

and the initial condition  $e_0 = [0 \ 1]^T$ . The spectral radius  $\rho(L)$  is in fact zero and hence the iteration  $e_{k+1} = Le_k$  converges. To be more precise,  $e_1 = [0 \ \epsilon]^T$  and  $e_k = [0 \ 0]^T$  for  $k \geq 2$ . Consequently the  $l_i$ -norm of  $e_1$  can be made arbitrarily large for  $i = 1, 2, \infty$  by increasing  $\epsilon$ , and still the iterative process converges after two iterations.

It can also be shown that

$$\|T\| \geq \rho(T) \quad (2.24)$$

i.e. the spectral condition is less restrictive. This is quite expected because, as discussed above, the spectral condition is a necessary and a sufficient condition for convergence, whereas the norm condition is only sufficient. See Edwards & Owens (1982) for a more thorough discussion on this topic.

**Remark 2.1** Note that for an arbitrary  $n \times n$ -matrix  $T$  the spectral radius  $\rho(T)$  is equal to

$$\rho(T) = \max_{i \in I} |\lambda_i(T)| \quad (2.25)$$

where  $I$  is again the index set  $I = \{1, 2, \dots, n\}$  and  $\lambda_i(T)$  for  $i \in I$  are the eigenvalues of  $T$ . See Lancaster & Tismenetsky (1985) for details.

### 2.3 Spectral radius condition and dynamical systems

In this section two examples will be given that illustrate that the convergence properties of an iterative process involving a dynamical system. The theoretical results presented in this section are adapted from Rogers & Owens (1992). As a first example consider the system

$$\begin{cases} x_{k+1}(t+1) = \Phi x_{k+1}(t) + \Gamma e_k(t), & x(0) = 0 \\ e_{k+1}(t) = C x_{k+1}(t) + D e_k(t), & t \in [0, 1, 2, \dots, T] \end{cases} \quad (2.26)$$

with  $x_{k+1}(\cdot) \in \mathbb{R}^n$ ,  $e_k(\cdot) \in \mathbb{R}^m$ . An equivalent 'integral' representation of this equation is given by

$$e_{k+1}(t) = C \sum_{i=0}^t \Phi^{t-i} \Gamma e_k(i-1) + D e_k(t) \quad (2.27)$$

(it is assumed that  $e_k(t) = 0$  for  $t < 0$ ) or in a more compact form  $e_{k+1}(t) = [T e_k](t)$  where  $T$  is defined in an obvious way. Furthermore, it is assumed that  $e_k(\cdot)$  comes from the space  $R^m[0, T]$ , which is the space of bounded, real  $m$ -vector-valued functions in the discrete time-interval  $t \in [0, 1, 2, \dots, T]$  with the norm

$$\|e(t)\| = \max_{t \in [0, 1, 2, \dots, T]} |e(t)| \quad (2.28)$$

where  $|\cdot|$  is a suitable norm in  $\mathbb{R}^n$ . In order to investigate the convergence properties of this iterative process, the spectral radius of  $T$  should be calculated. This has been already done in Rogers & Owens (1992) resulting in the following proposition:

**Proposition 2.1** *The iteration (2.27) converges asymptotically to zero, if and only if, the eigenvalues of the  $m \times m$  matrix  $D$  are inside the unit circle (or equivalently  $\rho(D) < 1$ ), and  $\rho(T) = \rho(D)$ .*

This result is at first sight quite surprising, because the stability of the iterative process depends only on  $D$ , and the  $\Phi$  matrix does not play any role. However as was explained at the beginning of this chapter, due to the finite nature of the trial length, stability along the time-axis is always guaranteed, and consequently the classical stability result in terms of  $\Phi$  (i.e. the eigenvalues of  $A$  have to be inside the unit circle) cannot be used here. If the iteration process is written in the following 'component form'

$$\begin{aligned} e_{k+1}(0) &= D e_k(0) \\ e_{k+1}(1) &= C \Gamma e_k(0) + D e_k(1) \\ e_{k+1}(2) &= C \Phi \Gamma e_k(0) + C \Gamma e_k(1) + D e_k(2) \\ &\vdots \\ e_{k+1}(T) &= C \Phi^{T-1} \Gamma e_k(0) + \dots + D e_k(T) \end{aligned} \quad (2.29)$$

the condition on the eigenvalues of  $D$  becomes very transparent. This is due to the fact that if the eigenvalues of  $D$  are inside the unit circle, based on the first line in (2.29)  $\lim_{k \rightarrow \infty} e_k(0) = 0$ . However, if  $e_k(0) = 0$ , then the second line in (2.29) is simply  $e_{k+1}(1) = D e_k(1)$ , and it also converges to zero because the eigenvalues of  $D$  are inside the unit circle. Continuing inductively,  $\lim_{k \rightarrow \infty} e_k(t) = 0$  for  $t \in [0, 1, 2, \dots, T]$ , showing that if the eigenvalues of  $D$  are inside the unit circle,  $e_k$  will converge to zero. Consequently  $D$  eventually squeezes component by component the vector  $e_k$  towards zero from left to right as  $k \rightarrow \infty$ . However, if  $\Phi$  is an unstable matrix it can generate vectors having a large norm during the earlier iterations (especially when the trial length is long), which can cause severe problems in practical applications.

**Remark 2.2** *It is important to realise that the asymptotic convergence result in Proposition 2.1 provides a certain degree of robustness. This is due to the fact that if the operator  $T$  has additive uncertainty of the form  $T + \gamma$ , there exists  $\delta > 0$  so that if  $\|\gamma\| < \delta$ , then the sequence  $\{e_k\}$  generated by*

$$e_{k+1} = (T + \gamma)e_k \quad (2.30)$$

*still converges asymptotically to the zero vector. The proof for this robustness property can be found from Rogers & Owens (1992).*

**Remark 2.3** *Note that the rigorous proof for Proposition 2.1 in Rogers & Owens (1992) relies heavily on the fact that the trial length is finite. If the trial length  $T \rightarrow \infty$ , the convergence condition requires the additional assumption that the plant  $(\Phi, \Gamma, C, D)$  is controllable and observable. Furthermore, the plant has to be stable in the classical sense (i.e. the eigenvalues of  $\Phi$  lie inside the unit circle) and it is required that*

$$\rho(L(z)) < 1 \quad \forall z \text{ with } |z| = 1 \quad (2.31)$$

*where  $L(z) := C(zI - \Phi)^{-1}\Gamma + D$ . In summary the finiteness of the trial length relaxes considerably the convergence conditions on the iterative process (2.26), because it results in a convergence test which is independent of  $\Phi$ .*

Note that the rather general result given in Proposition 2.1 can be used to generate the first rigorous proof for the 'Arimoto-algorithm' (see Example 1.1 in Chapter 1)

**Proposition 2.2** *Consider the ILC algorithm*

$$u_{k+1}(t) = u_k(t) + Ke_k(t+1) \quad (2.32)$$

*where  $K$  is an  $m \times m$  matrix and the algorithm is applied to the system*

$$\begin{cases} x_{k+1}(t+1) = \Phi x_{k+1}(t) + \Gamma u_{k+1}(t), & x_{k+1}(0) = 0 \\ y_{k+1}(t) = Cx_{k+1}(t), & t \in [0, 1, 2, \dots, T] \end{cases} \quad (2.33)$$

*where it is assumed that  $r(0) = 0$  (which implies that  $e_{k+1}(0) = 0$ ). The necessary and sufficient condition for asymptotic convergence, i.e.  $\lim_{k \rightarrow \infty} e_k(t) = 0$  for  $t \in [0, 1, 2, \dots, T]$ , is that the eigenvalues of  $I - CBK$  lie inside the unit circle.*

**Proof.** In order to analyse the convergence properties of this algorithm define

$$\begin{aligned} \Delta u_{k+1}(t) &:= u_{k+1}(t) - u_k(t) \\ \Delta x_{k+1}(t) &:= x_{k+1}(t) - x_k(t) \end{aligned} \quad (2.34)$$

and the system model can be written equivalently with these definitions as

$$\begin{cases} \Delta x_{k+1}(t+1) = \Phi \Delta x_{k+1}(t) + \Gamma \Delta u_{k+1}(t), & \Delta x_{k+1}(0) = 0 \\ e_{k+1}(t) = e_k(t) - C \Delta x_{k+1}(t), & t \in [0, 1, 2, \dots, T] \end{cases} \quad (2.35)$$

Inserting the control(2.32) into the system model (2.33) results in

$$\begin{cases} \Delta x_{k+1}(t+1) = \Phi \Delta x_{k+1}(t) + \Gamma K e_k(t+1), & \Delta x_{k+1}(0) = 0 \\ e_{k+1}(t) = e_k(t) - C \Delta x_{k+1}(t), & t \in [0, 1, 2, \dots, T] \end{cases} \quad (2.36)$$

Because the state equation depends on  $e_k(t+1)$ , it can be 'pulled through' into the measurement equation resulting in the equivalent IO-representation

$$\begin{cases} \Delta \tilde{x}_{k+1}(t+1) = \Phi \Delta \tilde{x}_{k+1}(t) + \Gamma K e_k(t), \Delta x_{k+1}(0) = 0 \\ e_{k+1}(t) = (I - C\Gamma K)e_k(t) - C\Phi \Delta \tilde{x}_{k+1}(t), \quad t \in [0, 1, 2, \dots, T] \end{cases} \quad (2.37)$$

This equation is now in the standard form of (2.26) and consequently, according to Proposition 2.1, the necessary and sufficient condition for convergence is that the eigenvalues of the  $I - C\Gamma K$  matrix must lie inside the unit circle.  $\square$

In summary the convergence properties of the structurally rather simple algorithm (2.32) depend only on  $C$  and  $\Gamma$ , and no knowledge is required from the dynamics of the system, i.e. the system matrix  $\Phi$ . However, as was explained earlier, the result guarantees only asymptotic convergence, and during earlier rounds the norm of the tracking error can be very large. Therefore this algorithm is not necessarily suitable for applications where large tracking errors cannot be tolerated. The key point in the convergence analysis is the finite nature of the trial length, because it guarantees that the asymptotic convergence result does not depend on  $\Phi$ , and consequently only  $C$  and  $\Gamma$  are needed to select  $K$ .

**Remark 2.4** *Note that no novelty value is claimed for Proposition 2.2. This same result has been derived by Moore (2000a), for example, but with a totally different technique.*

## 2.4 Convergence analysis of high-order algorithms

In the previous section the convergence properties of the iterative process  $x_{k+1} = Tx_k$  were analysed, where  $T: \mathcal{X} \rightarrow \mathcal{X}$  and  $\mathcal{X}$  is a suitable complete metric space. In the analysis the outcome of the iterative process for iteration  $k+1$  depends only on events during the previous iteration. However, as was explained in Chapter 1, in ILC it is possible to use the tracking error information and input function information for trial  $k+1$  from trials  $k, k-1, \dots, k+1-n$ . In this case the error evolution for  $e_{k+1}$  is typically of the form

$$e_{k+1} = L_1 e_k + L_2 e_{k-1} + \dots + L_n e_{k+1-n} \quad (2.38)$$

and it seems that the converge analysis presented in the previous sections cannot be exploited. Fortunately, a simple manipulation described in Edwards & Owens (1982), Amann *et al.* (1996), and Kaczorek (1993), converts this problem back to the standard problem  $x_{k+1} = Tx_k$ . Consider first the product space  $\mathcal{X}_n := \mathcal{X} \times \mathcal{X} \times \dots \times \mathcal{X}$  ( $n$  times) and define a 'state-vector'  $\vec{e}_k$  as

$$\vec{e}_k := [e_{k+1-n} \ \dots \ e_{k-1} \ e_k]^T \quad (2.39)$$

With this notation, the high-order error evolution equation (2.38) can be written equivalently as

$$\vec{e}_{k+1} = \begin{bmatrix} 0 & I & 0 & \dots & 0 \\ 0 & 0 & I & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & I \\ L_n & L_{n-1} & L_{n-2} & \dots & L_1 \end{bmatrix} \vec{e}_k \quad (2.40)$$

or in more compact form  $\vec{e}_{k+1} = L_e \vec{e}_k$  where  $L_e : \mathcal{X}_n \rightarrow \mathcal{X}_n$ , and consequently the methods from the previous section can be applied on the operator matrix  $L_e$ . However, if the elements of  $L_e$  are for example matrix operators without any special structure, the calculation of the spectral radius of  $L_e$  can be analytically almost impossible and computationally very intensive. Consequently, the representation of the learning operator as a matrix of operators does not always result in convergence results that can be easily tested analytically or with numerical methods.

## 2.5 Summary

In this chapter the convergence properties of iterative processes of the form  $x_{k+1} = Tx_k$  were reviewed, where  $T$  is a given operator. Two conditions were given: namely a contraction mapping condition and a spectral radius condition. The contraction mapping condition guarantees that  $\|x_{k+1}\| \leq \|x_k\|$ , which is sometimes a very desirable property in practical applications. The spectral radius condition, on the other hand, guarantees that  $\lim_{k \rightarrow \infty} x_k = 0$ , but the norm of  $x_k$  can be undesirably large during earlier iterations. These two abstract conditions are very important in the analysis of ILC algorithms, because most of the algorithms derived in this thesis result in the error evolution equation  $e_{k+1} = Le_k$  where  $L$  is the learning operator that maps the tracking error from iteration  $k$  to iteration  $k + 1$ . Finally, if the resulting error evolution is a high-order equation  $e_{k+1} = L_1 e_k + L_2 e_{k-1} + \dots + L_n e_{k-n+1}$ , it was shown how a simple manipulation can be used to convert this equation into a form which is again accessible to the contraction mapping and spectral radius condition.

### 3 Algebraic analysis of ILC systems

In this chapter the algebraic properties of ILC systems are analysed in the discrete-time case. As was explained in the first chapter, the starting point in ILC is a dynamical model

$$\begin{cases} x_{k+1}(t+1) = \Phi x_{k+1}(t) + \Gamma u_{k+1}(t), & x_{k+1}(0) = x_0 \\ y_{k+1}(t) = C x_{k+1}(t), & t \in [0, N-1] \end{cases} \quad (3.1)$$

and the objective is to find iteratively an input  $u^*(t)$  so that the output  $y^*(t)$  of the system would follow as accurately as possible a given reference signal  $r(t)$  on the finite time-interval  $t \in [0, N-1]$ . In this chapter only SISO systems are considered for notational simplicity. All the results in this chapter, however, can be extended in a straightforward manner to cover MIMO systems. Note also that for notation simplicity in (3.1) it is assumed (without loss of generality) that the sampling time is one time unit.

As was analysed in the first and second chapter, the iterative learning mechanism results in a two-dimensional system, where the iteration axis is infinite and the time axis is finite, and in the linear case the convergence properties of the system do not depend on ‘stability along the time-axis’. Because in the discrete-time case the time-interval consists of finite number of points, one way to approach the analysis of ILC systems could be to somehow hide the time axis all together from the plant model (3.1), and to construct an equivalent plant model where the only independent variable is the iteration axis  $k$ . This is done in the next section, where it is shown that both a causal and non-causal difference equation with zero initial (boundary) conditions can be replaced with an equivalent Toeplitz matrix mapping. The commutativity and invertibility properties of Toeplitz matrices are reviewed, and it turns out that the transfer function approach used commonly in ILC literature (where the transfer operator acts along the time-axis) can result in ambiguous results, whereas the matrix approach is a rigorous way to approach dynamical systems defined over a finite time-interval.

As a next step the matrix mapping for the process model (3.1) is established, and issues related to the relative degree of  $G(z) = C(zI - \Phi)^{-1}\Gamma$  and whether or not the reference signal  $r(t)$  belongs to the range of the matrix mapping are discussed. The matrix description of the plant model allows an ILC problem to be

considered as a tracking problem where the plant is multivariable and static, and the reference signal is a step function. This equivalent description of the ILC problem results in a general convergence theory for discrete-time ILC systems, which is the main contribution of this chapter. The convergence theory also suggests that time-varying ILC algorithms should be used to guarantee good transient behaviour, whereas in the current research literature it is more common to consider time-invariant algorithms. New optimality-based (in some cases time-varying) ILC algorithms are derived in the later chapters as a direct result of this observation.

### 3.1 Matrix representations of discrete-time dynamical systems

This section shortly analyses how a general time-invariant, discrete-time system defined over a finite-time interval can be equivalently described with a matrix mapping. Consider a discrete-time dynamical input-output system defined by the following convolution relation

$$y(t) = s_{N-1}u(t+N-1) + s_{N-2}u(t+N-2) \cdots + du(t) + r_1u(t-1) + \cdots + r_{N-1}u(t-N+1) \quad (3.2)$$

where  $t \in [0, 1, \dots, N-1]$  and  $s_i, d, r_i \in \mathbb{R}$ . This relation contains both causal and non-causal linear filters over a finite time-interval  $[0, N-1]$ . In the ILC framework  $N$  corresponds to the length of a trial and it is typically ‘a large number’. Note that in ILC the data from previous trials can be processed between the trials and hence it is possible to use also non-causal filters in ILC algorithms. In fact, as will be shown in this chapter, they are typically needed in order to guarantee convergence. In order to make (3.2) a mapping the time domain is enlarged to  $\mathbb{Z}$ , the set of integers, and it is assumed that  $u(t) = 0$  for  $t > N$  and  $t < 0$ . To make the range and domain of the mapping coincide, the result  $y(t)$  of the mapping (3.2) is padded with zeros for  $t > N$  and  $t < 0$  (these assumptions are frequently made in ILC papers, at least implicitly). With these assumptions it is easy to see that the convolution mapping combined with the ‘zero padding’ can be described equivalently with a matrix mapping

$$\vec{y} = G_e \vec{u} \quad (3.3)$$

where

$$G_e = \begin{bmatrix} d & s_1 & s_2 & \cdots & s_{N-1} \\ r_1 & d & s_1 & \cdots & s_{N-2} \\ r_2 & r_1 & d & \cdots & s_{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{N-1} & r_{N-2} & r_{N-3} & \cdots & d \end{bmatrix} \quad (3.4)$$

$\vec{y} := [y(0) \ y(1) \ \dots \ y(N-1)]^T$  and  $\vec{u} := [u(0) \ u(1) \ \dots \ u(N-1)]^T$  are so called ‘super-vectors’. The matrix  $G_e \in \mathbb{R}^{N \times N}$  is clearly a Toeplitz-matrix, i.e. it has constant

elements along the diagonals. In order to analyse series or parallel interconnections (the two basic interconnections needed in systems theory) of systems given by (3.3), it is clear that the proper way to do the analysis is to use the non-commutative ring structure of  $\mathbb{R}^{N \times N}$  matrices where especially the non-commutative multiplication makes calculations in this algebraic structure more complex. However, the non-commutativity of the multiplication can be relaxed in some cases: according to Gantmacher (1959), the product of two arbitrary matrices  $G_{e,1}G_{e,2}$ , that are lower-triangular and Toeplitz, commutes, (i.e.  $G_{e,1}G_{e,2} = G_{e,2}G_{e,1}$ ), and the resulting matrix is also a lower triangular Toeplitz matrix. This result also holds when the word ‘lower-triangular’ is replaced by the word ‘upper-triangular’. There are also many invertible elements available in the class of matrices given by (3.3) - one example is a lower-triangular  $G_e$  with the property  $d \neq 0$ , the inverse being again a lower-triangular Toeplitz-matrix. Hence in these special cases it is immediately known that the resulting system from either a series connection of two systems or an inversion of a system has a dynamical system interpretation.

The possibility of using both non-causal and causal filters makes the ‘z-transfer calculus’ (where the z-transform is taken along the time-axis) of ILC systems quite complicated even though a lot of ILC publications resort to this technique. The complication is due the fact that even if the time-interval of the system would be the whole  $\mathbb{Z}$  (i.e. the effect of truncation to finite interval is not considered), a non-causal and causal filter have typically a disjoint region of convergence, and multiplication of two transfer functions having a disjoint region convergence is meaningless in terms of convolution operation in the time-domain. However, as was shown in this section, the matrix representation is still a rigorous way to analyse an ILC system even if the ILC system contains both causal and non-causal subsystems.

**Example 3.1** Consider the transfer functions

$$G_1(z) = \frac{1}{z + \alpha}, \quad G_2(z) = z \quad (3.5)$$

If it is assumed that both transfer functions come from the commutative field (field of fractions)  $\mathbb{C}(z)$ , which is the assumption frequently taken in the ILC literature, then by definition they commute, i.e.

$$G_1(z)G_2(z) = G_2(z)G_1(z) \quad (3.6)$$

Suppose now that the trial length is four time units. In this case the matrix representation for  $G_1(z)$  becomes

$$G_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -\alpha & 1 & 0 & 0 \\ \alpha^2 & -\alpha & 1 & 0 \end{bmatrix} \quad (3.7)$$

and for  $G_2(z)$

$$G_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.8)$$

The multiplication  $G_1G_2$  (i.e. a series connection) results in

$$G_1G_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -\alpha & 1 & 0 \\ 0 & \alpha^2 & -\alpha & 1 \end{bmatrix} \quad (3.9)$$

whereas the other series connection  $G_2G_1$  is equal to

$$G_2G_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\alpha & 1 & 0 & 0 \\ \alpha^2 & -\alpha & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.10)$$

Consequently by assumption  $G_1(z)G_2(z) = G_2(z)G_1(z)$  but  $G_1G_2 \neq G_2G_1$ , which demonstrates in a clear fashion that the transfer function approach cannot be used in ILC without mathematical ambiguities.

### 3.2 The process model and invertibility

Since the system (3.1) is defined over a finite time-interval, it can be represented equivalently with a matrix equation  $\vec{y}(k) = G_e\vec{u}(k) + \vec{d}$ , where

$$G_e = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ C\Gamma & 0 & 0 & \dots & 0 \\ C\Phi\Gamma & C\Gamma & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C\Phi^{N-2}\Gamma & C\Phi^{N-3}\Gamma & \dots & \dots & 0 \end{bmatrix}, \quad \vec{d} = \begin{bmatrix} Cx_0 \\ C\Phi x_0 \\ C\Phi^2 x_0 \\ \vdots \\ C\Phi^{N-1}x_0 \end{bmatrix} \quad (3.11)$$

The elements  $C\Phi^j B$  of the matrix  $G_e$  are the Markov parameters of the plant (3.1), and  $d$  is the initial condition response. Furthermore, the ‘super-vectors’  $\vec{u}(k)$  and  $\vec{y}(k)$  are defined as

$$\begin{aligned} \vec{u}(k) &:= [u_k(0) \ u_k(1) \ \dots \ u_k(N-1)]^T \\ \vec{y}(k) &:= [y_k(0) \ y_k(1) \ \dots \ y_k(N-1)]^T \end{aligned} \quad (3.12)$$

The tracking error  $\vec{e}(k+1)$  is defined with the equation

$$\vec{e}(k+1) := \vec{r} - \vec{y}(k+1) = r - \left( G_e\vec{u}(k+1) + \vec{d} \right) = \left( r - \vec{d} \right) - G_e\vec{u}(k+1) \quad (3.13)$$

and consequently the possibly non-zero initial condition response can be embedded into the reference signal, and from now on (without loss of generality) it is assumed that  $\vec{d} = 0$ , or equivalently,  $x_{k+1} = 0$ . Also note that time-varying process state-space models can be represented with a matrix mapping. The only difference is that, in this case, the diagonals of (3.11) are no longer constant. Furthermore, in

Moore (2000b) it has been shown that a subclass of nonlinear systems exists that can be written in a similar matrix form to (3.11).

In order to investigate whether or not there exists an input  $u^*$  that results in perfect mapping, as a first observation note that  $e(0) = r(0) - Cx(0) = 0$ , and consequently a necessary condition for the existence of  $u^*$  is that  $r(0) = Cx(0)$  or, in other words, the reference signal must belong to the range of the plant. Assume now that  $C\Gamma \neq 0$ . In this case the input sequence  $u_k$  affects only the output values  $y_k(t)$  for  $t \geq 1$ . Consequently the matrix mapping  $G_e$  in (3.11) can be replaced with the ‘lifted’ mapping  $G_{e,l}$  given by the equation

$$G_{e,l} = \begin{bmatrix} C\Gamma & 0 & 0 & \dots & 0 \\ C\Phi\Gamma & C\Gamma & 0 & \dots & 0 \\ C\Phi^2\Gamma & C\Phi\Gamma & C\Gamma & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C\Phi^{N-2}\Gamma & C\Phi^{N-3}\Gamma & \dots & \dots & C\Gamma \end{bmatrix} \quad (3.14)$$

which now maps the lifted input sequence  $\vec{u}_l(k) := [u_k(0) \ u_k(1) \ \dots \ u_k(N-2)]^T$  to the lifted output sequence  $\vec{y}_l(k) := [y_k(1) \ y_k(2) \ \dots \ y_k(N-1)]^T$ . Furthermore, given an arbitrary reference signal  $r$  for which  $r(0) = Cx(0)$ , there now exists a unique input function  $u^*$  which results in perfect tracking, because the lifted plant  $G_{e,l}$  is invertible (under the assumption  $C\Gamma \neq 0$ ), and

$$\vec{u}_l^* = G_{e,l}^{-1} \vec{r}_l \quad (3.15)$$

where  $\vec{r}_l := [r(1) \ r(2) \ \dots \ r(N-1)]^T$ . If  $C\Gamma = 0$  and  $C\Phi^{m-1}\Gamma$  is the first non-zero Markov parameter (i.e.  $m$  is the relative degree of the plant), the same lifting technique can be used to generate a matrix representation of the dynamical system, where in (3.14) the diagonal term is the first non-zero Markov parameter  $C\Phi^{m-1}\Gamma$  and the rest of the elements are defined in a similar manner as in (3.14). In this case the matrix  $G_e$  becomes a mapping from  $\vec{u}_l(k) := [u_k(0) \ u_k(1) \ \dots \ u_k(N-1-m)]$  to  $\vec{y}_l(k) := [y_k(m) \ y_k(1) \ \dots \ y_k(N-1)]$ . Furthermore, for invertibility it has to be assumed that  $r(0) = Cx(0)$ ,  $r(1) = C\Phi x_0$ ,  $\dots$ ,  $r(m-1) = C\Phi^{m-1}x_0$ .

In this chapter the lifted plant could be used as a starting point for analysis by making the necessary assumptions. However, in order to demonstrate the effect of initial conditions in a precise manner, the original plant model  $G_e$  in (3.11) is used instead. For simplicity it is assumed that  $C\Gamma \neq 0$ , i.e. the relative degree of the plant is one, but all the results in this chapter can straightforwardly be extended for plants with higher relative degree.

### 3.3 Polynomial description of ILC systems

Now consider as a motivational example the following ILC law suggested in Moore (1993)

$$u_{k+1}(t) = u_k(t) + \gamma e_k(t+1) \quad (3.16)$$

which is used to control the system described in (3.1). The control law can be written in the super-vector notation as  $\vec{u}(k+1) = \vec{u}(k) + G_{e,c}\vec{e}(k)$  where

$$G_c = \begin{bmatrix} 0 & \gamma & 0 & \dots & 0 \\ 0 & 0 & \gamma & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \gamma \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix} \quad (3.17)$$

and the process model becomes in the super-vector notation as  $\vec{y}(k) = G_e\vec{u}(k)$  where  $G_e$  is given by (3.11). Time-varying process state-space models can also be represented with a matrix mapping. The only difference is that, in this case, the diagonals of (3.11) are no longer constant. Furthermore, in Moore (2000b) it has been shown that there exists a subclass of nonlinear systems, that can be written in a similar matrix form as (3.11).

Now define the following set  $(\mathbb{R}^N)^{\mathbb{Z}}$  of all functions from  $\mathbb{Z} \rightarrow \mathbb{R}^N$  and the operator  $w^{-1} : (\mathbb{R}^N)^{\mathbb{Z}} \rightarrow (\mathbb{R}^N)^{\mathbb{Z}}$  by  $(w^{-1}\vec{v})(k) = \vec{v}(k-1)$  for an arbitrary  $\vec{v}(k) \in (\mathbb{R}^N)^{\mathbb{Z}}$ . By using the  $w^{-1}$ -operator the control law can be written as

$$(I - w^{-1}I)\vec{u}(k) = w^{-1}G_c\vec{e}(k) \quad (3.18)$$

where  $\vec{e} := [r(0) - y(0) \ r(1) - y(1) \ \dots \ r(N-1) - y(N-1)]^T$  is the tracking error. Solving for  $\vec{e}(k)$  gives (multiplying from the left with the process model and noting that the reference signal is independent of the iteration round)

$$(I - w^{-1}L)\vec{e}(k) = 0 \quad (3.19)$$

where

$$L = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 - \gamma C\Gamma & 0 & \dots & 0 \\ 0 & -\gamma C\Phi\Gamma & 1 - \gamma C\Gamma & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & -\gamma C\Phi^{N-2}\Gamma & -\gamma C\Phi^{N-3}\Gamma & \dots & 1 - \gamma C\Gamma \end{bmatrix} \quad (3.20)$$

However, both equations (3.18) and (3.19) come from the structure of  $\mathbb{R}^{N \times N}[w^{-1}]$ , i.e. the ring of  $N \times N$ -matrices having coefficients from the polynomial ring  $\mathbb{R}[w^{-1}]$  where  $w^{-1}$  is the delay operator defined earlier. This is the standard algebraic structure for discrete-time multivariable linear systems, and both the analysis and synthesis of these systems are a well-established area in systems theory, see for example Blomberg & Ylinen (1983) and Kucera (1979).

In order to analyse the convergence properties of (3.19) the following standard result from discrete-time multivariable linear systems theory is needed:

**Proposition 3.1** *With arbitrary initial condition, the equation  $A(w^{-1})\vec{v}(k) = 0$ , where  $A(w^{-1}) \in \mathbb{R}^{N \times N}[w^{-1}]$ , satisfies  $\lim_{k \rightarrow \infty} \vec{v}(k) = 0$  if and only if the elements of the set  $P$ ,  $P = \{p_i \in \mathbb{C} \mid \det(A(p_i^{-1})) = 0\}$  are inside the unit circle (the complex numbers  $p_i \in \mathbb{C}$  are conventionally identified as the poles of the system).*

For a rigorous proof see Blomberg & Ylinen (1983). However, for the convenience of the reader the main steps of the proof are provided below to show that the convergence condition in Proposition 1 is a sufficient one. A simple counter-example can be easily constructed to prove that the convergence condition in the proposition is also a necessary one.

**Sketch of the proof.** Let  $S$  be the set that contains the solutions of the equation  $A(w^{-1})\vec{v}(k) = 0$ . Take an arbitrary unimodular matrix  $Q(w^{-1})$  (i.e.  $\det Q(w^{-1}) = c, c \in \mathbb{C}$ ) and consider the set

$$\tilde{S} = \{\vec{v}(k) \in (\mathbb{R}^N)^{\mathbb{Z}} \mid Q(w^{-1})A(w^{-1})\vec{v}(k) = 0\} \quad (3.21)$$

It can be shown that it always holds that  $\tilde{S} = S$ , and therefore multiplying  $A(w^{-1})$  from the left with an arbitrary unimodular matrix does not change the original solution set  $S$ . Furthermore, the unimodular matrix can always be chosen so that  $\tilde{A}(w^{-1}) := Q(w^{-1})A(w^{-1})$  is a lower-triangular polynomial matrix (triangularisation can be achieved by carrying out elementary row operations on the matrix  $A(w^{-1})$ ). Denote the elements of  $\tilde{A}(w^{-1})$  with  $\tilde{a}_{ij}(w^{-1})$ . Using this notation the equation  $\tilde{A}(w^{-1})v(k) = 0$  can be written component-wise as

$$\begin{cases} \tilde{a}_{11}(w^{-1})v(k, 0) = 0 \\ \tilde{a}_{22}(w^{-1})v(k, 1) + \tilde{a}_{21}(w^{-1})v(k, 0) = 0 \\ \vdots \\ \tilde{a}_{ss}(w^{-1})v(k+1, s-1) + \sum_{i=1}^{s-1} \tilde{a}_{si}(w^{-1})v(k, i-1) = 0 \end{cases} \quad (3.22)$$

for  $s = 3, \dots, N$ . The first equation is an autonomous difference equation, and  $v(k, 0)$  converges to zero, if the solutions of the equation  $\{p \in \mathbb{C} \mid a_{11}(p^{-1}) = 0\}$  are inside the unit circle. The second equation is driven by  $v(k, 0)$  and the output is  $v(k, 1)$ . However, if the solutions  $\{p \in \mathbb{C} \mid \tilde{a}_{11}(p^{-1}) = 0\}$  are inside the unit circle, the driving function  $v(k, 0)$  converges to zero. Therefore if the solutions  $\{p \in \mathbb{C} \mid \tilde{a}_{22}(p^{-1}) = 0\}$  are inside the unit circle,  $\lim_{k \rightarrow \infty} v(k, 1) = 0$ . Using an induction argument it is seen that if the solutions of  $\{p \in \mathbb{C} \mid \tilde{a}_{ss}(p^{-1}) = 0\}$  are inside the unit circle then  $\lim_{k \rightarrow \infty} v(k, s) = 0, s = 3, \dots, N$ . Furthermore,

$$\det(Q(w^{-1})A(w^{-1})) = \det(Q(w^{-1})) \det(A(w^{-1})) = c \det A(w^{-1}) \quad (3.23)$$

but because  $\tilde{A}(w^{-1}) = Q(w^{-1})A(w^{-1})$  is a lower-triangular matrix, it holds that

$$\det(Q(w^{-1})A(w^{-1})) = \det \tilde{A}(w^{-1}) = c \tilde{a}_{11}(w^{-1}) \tilde{a}_{22}(w^{-1}) \dots \tilde{a}_{NN}(w^{-1}) \quad (3.24)$$

and therefore the sufficient condition for convergence becomes that the elements of the set  $P = \{p_i \in \mathbb{C} \mid \det(A(p_i^{-1})) = 0\}$  are inside the unit circle.  $\square$

Based on Proposition 1 the convergence condition for the algorithm (3.16) is given by the following

**Proposition 3.2** *A necessary and sufficient condition for the convergence of the algorithm (3.16) on the time-interval  $t \in [1, N-1]$  with an arbitrary initial input function  $\vec{u}(0)$  is that  $|1 - \gamma C \Gamma| < 1$ . In addition  $e_k(0) = r(0)$  for  $k = 0, 1, 2, \dots$*

**Proof.** From (3.20) it can be seen that the poles of the system (3.19) (‘iteration axis poles’) are given by  $p_1 = 1$ ,  $p_i = (1 - \gamma CT)$  for  $i = 2, \dots, N$ . Hence even if  $\gamma$  is selected so that  $|1 - \gamma CT| < 1$ , (3.19) is *marginally* stable. However, due to the special nature of (3.20) (first column and first row), the condition  $|1 - \gamma CT| < 1$  guarantees that  $\lim_{k \rightarrow \infty} e_k(t) = 0$  for  $t = 1, 2, 3, \dots, N - 1$  and from (3.20)  $e_k(0) = r(0)$  for all  $k$ .  $\square$

The discussion so far also results in the following general conclusions about convergence analysis of ILC systems: according to (3.11), the original system to be controlled can be regarded as a *static* linear multivariable system. Furthermore, a linear (possibly time-varying) ILC controller can always be written in the  $\mathbb{R}^{N \times N}[w^{-1}]$ -domain. Also, if it is assumed that the initial conditions of the controller are zero, the controller can be written as a linear matrix fractional representation in  $\mathbb{R}^{N \times N}(w^{-1})$  (see Blomberg & Ylinen (1983)). This guarantees that  $\vec{e}(k)$  can be solved as a function of the reference signal  $\vec{r}$  and the standard techniques from linear, discrete-time, multivariable control theory can be used to analyse convergence. As an example of how the theory developed so far can be used, consider the following (also derived in Owens *et al.* (2002) and Moore (2000a))

**Proposition 3.3** *Let a given ILC law be of the form*

$$\begin{aligned} \vec{u}(k) = & H_1 \vec{u}(k-1) + H_2 \vec{u}(k-2) + \dots + H_r \vec{u}(k-r) \\ & + K_0 \vec{e}(k) + K_1 \vec{e}(k-1) + \dots + K_s \vec{e}(k-s) \end{aligned} \quad (3.25)$$

where it is assumed that  $H_i \in \mathbb{R}^{N \times N}$  commute with the process model, i.e.  $G_e H_i = H_i G_e$  and that  $(I + G_e K_0)$  is invertible. Then a sufficient and necessary condition for zero convergence under arbitrary initial conditions  $\vec{u}(-1), \vec{u}(-2), \dots, \vec{u}(-r)$  is that  $\sum_{i=1}^r H_i = I$  and the solutions of

$$\det(A(w^{-1})) = \det(I + C_1 w^{-1} + \dots + C_d w^{-d}) = 0 \quad (3.26)$$

are inside the unit circle.

**Proof.** Let  $d = \max(s, r)$ . Then the error evolution equation becomes

$$\begin{aligned} \vec{e}(k) + C_1 \vec{e}(k-1) + C_2 \vec{e}(k-2) \dots C_d \vec{e}(k-d) \\ = (I + G_e K_0)^{-1} (I - H_1 - H_2 - \dots - H_r) \vec{r} \end{aligned} \quad (3.27)$$

where  $C_i \in \mathbb{R}^{N \times N}$  and are given by the equation

$$C_i = G_e K_i - H_i \quad (3.28)$$

for  $i = 1, 2, \dots, d$ . Here it is defined that if  $i > r$  then  $H_i = 0$  and if  $i > s$  then  $K_i = 0$ . Based on left-hand side of (3.27) a necessary condition for error convergence to zero becomes  $\sum_{i=1}^r H_i = I$ . Furthermore, based on Proposition 3.1 it is clear that a sufficient condition for convergence is that the solutions  $p_i$  (i.e. poles along the iteration axis) of the characteristic equation

$$\det(A(w^{-1})) = \det(I + C_1 w^{-1} + \dots + C_d w^{-d}) = 0 \quad (3.29)$$

lie inside the unit circle in the complex plane.  $\square$

Note that the assumption  $G_e H_i = H_i G_e$  requires that the matrices operating on previous inputs are generated by causal LTI filters and the invertibility of  $(I + G_e K_0)$  is guaranteed as long as  $K$  is generated by a causal LTI filter.

The synthesis problem in the discrete-time ILC case is equivalent to finding a standard multivariable control law as a linear matrix fraction representation for a *static* multivariable plant (3.11) when the output of the system is required to follow a ‘multi-channel’ step-function  $\vec{r}$ . This discussion immediately shows that the ILC synthesis problem in the discrete-time case can be approached with standard techniques, with  $H_2$ -theory and  $H_\infty$ -theory being two possible options. However, from the implementation point of view (especially when the ratio between trial length and sampling time is high) it is important that when the resulting control law from synthesis is given by  $(I + H_1 w^{-1} + \dots + H_r w^{-r})\vec{u}(k) = (K_0 + K_1 w^{-1} + \dots + K_s w^{-s})\vec{e}(k)$ , the matrices  $H_i, K_i \in \mathbb{R}^{N \times N}$  have a dynamical system interpretation (i.e. they can be realised as recursive equations along the time-axis). Otherwise computationally impractical lookup tables have to be used in the real-time implementation of the ILC law. Note that these two conclusions were made for the first time in a more restricted form in Moore (2000a) and Owens *et al.* (2002).

### 3.4 Controller design

Consider again the control law (3.25) which can be written in more compact form as  $m(w^{-1})\vec{u}(k) = n(w^{-1})\vec{e}(k)$  and assume again that  $G_{e,p} H_i = H_i G_{e,p}$  and that the system is supposed to track a reference signal  $\vec{r}(k)$ . Under these assumptions the error evolution equation becomes

$$(m(w^{-1}) + G_{e,p} n(w^{-1}))\vec{e}(k) = m(w^{-1})\vec{r}(k) \quad (3.30)$$

which results in the following

**Proposition 3.4** *A necessary condition for zero convergence with the algorithm (3.25) is that*

$$\vec{r}(k) \in \text{Ker}(m(w^{-1})) \quad (3.31)$$

**Proof.** By definition (see Blomberg & Ylinen (1983))  $\text{Ker}(m(w^{-1})) = \{z(k) \in (\mathbb{R}^N)^{\mathbb{Z}} \mid (m(w^{-1})z)(k) = 0\}$  and  $(\mathbb{R}^N)^{\mathbb{Z}}$  is the set of all functions from  $\mathbb{Z} \rightarrow \mathbb{R}^N$ . Hence if  $\vec{r}(k) \in \text{Ker}(m(w^{-1}))$ , the error evolution equation (3.30) gives

$$(m(w^{-1}) + G_{e,p} n(w^{-1}))\vec{e}(k) = m(w^{-1})\vec{r}(k) = 0 \quad (3.32)$$

which completes the proof.  $\square$

In the standard ILC case the reference is a step-function iteration-wise,  $\vec{r}(k) = \vec{r}$ , and hence a necessary condition for zero convergence is that  $(I - w^{-1}I)$  has to

be a factor of  $m(w^{-1})$ . In other words, the controller has to include an integrator for zero convergence. This observation also shows that ILC can be used to learn or achieve perfect tracking of more complicated reference signals as long as the reference signal belongs to  $\text{Ker}(m(w^{-1}))$ , possibly resulting in a high-order ILC algorithm. Another way to state this observation is to say that the controller has to have an internal model of the reference signal in order to achieve perfect tracking, which is the well-known internal model principle first published in Francis & Wonham (1975). Note that if  $m(w^{-1})$  contains an internal model of  $\bar{r}(k)$ , based on (3.32) the behaviour of the tracking error is determined from the equation  $(m(w^{-1}) + G_e n(w^{-1}))\bar{e}(k) = 0$ . Furthermore, in this case Proposition 1 states that it is guaranteed that  $\lim_{k \rightarrow \infty} \bar{e}(k) = 0$  if and only if the zeros of  $\det(m(w^{-1}) + G_e n(w^{-1}))$  are inside the unit circle. In summary, the design of an ILC controller consists of selecting  $m(w^{-1})$  and  $n(w^{-1})$  so that the polynomial matrix  $m(w^{-1}) + G_e n(w^{-1})$  is stable in the sense of Proposition 1. Furthermore, for perfect tracking  $m(w^{-1})$  has to contain the internal model of  $\bar{r}(k)$  as a factor.

What, then, does a good controller in the standard ILC case look like? As an example consider again the Arimoto-type control law  $u_{k+1}(t) = u_k(t) + \gamma e_k(t+1)$ . As was shown in the previous section the poles of the error evolution equation are  $(p_1 = 1)$  and  $(p_i = 1 - \gamma CT)$  for  $i = 2, \dots, N$ . The existence of these  $N - 1$  multiple poles shows that the solution for  $\bar{e}(k)$  is a linear combination of terms of the form  $k^i(1 - \gamma CT)^k$  for  $i \in \{0, 1, 2, \dots, N - 2\}$ . Now if the term  $(1 - \gamma CT) \neq 0$  (in the worst case close to unity), the term  $k^i$  will become dominant during the earlier iteration rounds, and consequently the norm of the error (in some suitable topology) can potentially grow to a very large number during earlier rounds before the term  $(1 - \gamma CT)^k$  starts to dominate. A natural remedy for this problem would be to use a *time-varying* control law

$$u_{k+1}(t) = u_k(t) + \gamma(t)e_k(t+1) \quad (3.33)$$

This algorithm will give the poles  $p_1 = 1$  and  $p_i = 1 - \gamma(i)CT$  for  $i = 2, \dots, N$  and for convergence  $|1 - \gamma(t)CT| < 1$ ,  $1 \leq t \leq N - 1$  is required. In this case  $\bar{e}(k)$  would be a linear combination of terms  $(1 - \gamma(t)CT)^k$ . Note here that the algebraic approach can be used to analyse time-varying control laws, whereas the rather common frequency-based stability analysis would fail. Furthermore, frequency-based analysis gives typically only sufficient conditions for convergence, whereas the conditions from the algebraic approach are both necessary and sufficient. However, the time-varying control law does not necessarily result in monotonic convergence in norm in a  $l_1, l_2$  or  $l_\infty$ -topology, i.e.  $\|\bar{e}(k+1)\|_i \leq \|\bar{e}(k)\|_i$  for  $i = 1, 2, \infty$ : this is due to the fact that the error evolution equation can be written time-wise as follows (this same development was done in terms of the input signal  $u_{k+1}(t)$  in

Moore (1998a))

$$\begin{aligned}
e_{k+1}(0) &= e_k(0) \\
e_{k+1}(1) &= (1 - \gamma(1))C\Gamma e_k(1) \\
e_{k+1}(2) &= (1 - \gamma(2))C\Gamma e_k(2) - \gamma(2)C\Phi\Gamma e_k(1) \\
&\vdots \\
e_{k+1}(s) &= (1 - \gamma(s))C\Gamma e_k(s) \\
&\quad - \sum_{i=1}^{s-1} \gamma(s)C\Phi^i\Gamma e_k(s - i)
\end{aligned} \tag{3.34}$$

for  $s = 3, \dots, N - 1$ . Hence if the terms  $\gamma(s)C\Phi^i\Gamma$  do not rapidly converge to zero (in other words the original plant has a slowly decaying impulse response), the error will accumulate to the end of the trial during the earlier iteration rounds due to the heavy interaction between ‘channels’. This will possibly result in bad transient behaviour, where the norm of the error can be extremely large during earlier iteration rounds. However, when compared to the time-invariant control law, the quality of the transient behaviour with the time-varying control law should be better, because the terms  $k^i$  disappear from the solution of the error evolution equation. This same phenomenon of bad transient behaviour was noticed with the time-invariant control law  $u_{k+1}(t) = u_k(t) + \gamma e_k(t + 1)$  in Longman (2000) where the author describes the convergence mechanism (3.34) as a convergence wave that starts from  $t = 0$  and gradually works its way through each time instant up to  $t = N - 1$ . As an interesting coincidence note that one of the rare algorithms available that has guaranteed exponential rate of convergence is in fact time-varying (see Amann *et al.* (1996)).

### 3.5 Simulation examples

In order to illustrate the theoretical results presented in this chapter consider again the continuous-time model

$$(p^2 + 5p + 1)y(t) = (p + 1)u(t) \tag{3.35}$$

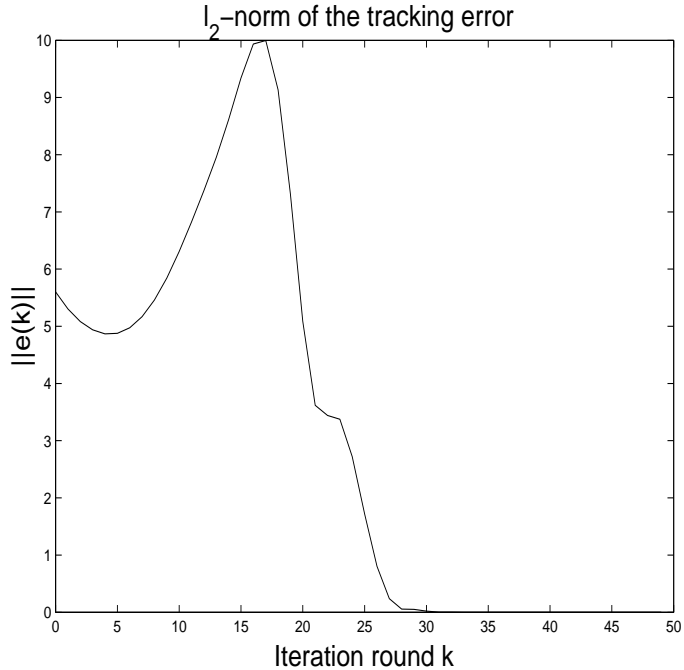
where  $p := \frac{d}{dt}$  and  $t \in [0, 6]$ . This model is sampled with 0.1 seconds, resulting in a discrete-time model (3.1) with

$$\Phi = \begin{bmatrix} 1.5595 & -0.6065 \\ 1 & 0 \end{bmatrix} \quad \Gamma = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} \tag{3.36}$$

$$C = [ 0.1643 \quad -0.1486 ]$$

The reference signal is  $r(t) = \sin(t)$  for  $t = [0, 0.1, \dots, 6]$ . As a first simulation example the control law  $u_{k+1}(t) = u_k(t) + \gamma e_k(t + 1)$  is used with  $\gamma = 10$  resulting in  $1 - \gamma C\Gamma = 0.1785$ . Fig. 3.1 shows the  $l_2$ -norm of the error as a function of the iteration round when the initial guess is  $u_0(t) = 0$ . The figure demonstrates how the ‘convergence-wave’ causes a rapid increase in the  $l_2$ -norm of the error during earlier rounds, but decreases asymptotically to zero as  $k \rightarrow \infty$ .

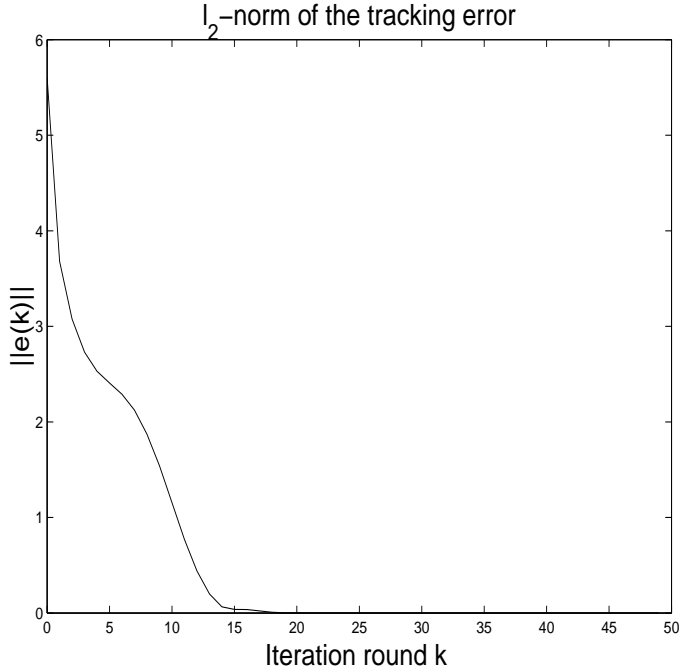
As a second simulation example (with the same initial guess  $u_0(t) = 0$ ) a time-varying control law  $u_{k+1}(t) = u_k(t) + \gamma(t)e_k(t + 1)$  is used where  $\gamma(t)$  increases from 5 to 11 with a fixed increment 0.1. This is done in order to obtain numerically different values for the closed-loop system poles. Fig. 3.2 again shows the  $l_2$ -norm of the error as a function of the iteration round. The time-varying control law is seen to provide a better transient behaviour than the time-invariant control law in the previous example.



**Figure 3.1.** The  $l_2$ -norm of the error as a function of iteration round with control law  $u_{k+1}(t) = u_k(t) + \gamma e_k(t + 1)$ .

### 3.6 Summary

In this chapter the algebraic properties of discrete-time ILC systems were analysed. As a starting point it was shown how (possibly non-causal) difference equations can be replaced with equivalent matrix mappings in the ILC case, and therefore the finite time-axis can be 'hidden' in the discrete-time case, and the complexities



**Figure 3.2.** The  $l_2$ -norm of the error as a function of iteration round with control law  $u_{k+1}(t) = u_k(t) + \gamma(t)e_k(t+1)$ .

resulting from 2-D nature of the original ILC problem can be avoided. In particular, the plant to be controlled can be described equivalently with a lower-triangular Toeplitz matrix, i.e. it can be seen as a static linear multivariable system.

In ILC it is assumed that the reference signal is independent of the iteration axis  $k$ , and hence it is a 'multi-channel' step function along the iteration axis. The internal model principle, on the other hand, says that the controller has to include an internal model of the reference signal in order to achieve perfect tracking (necessary condition), which is now an integrator. Therefore in the discrete-time case the ILC design problem is to construct a feedback controller for the static multivariable plant where the controller transfer function matrix has to have an integrator as a factor in order to guarantee asymptotic convergence. In addition, the controller has to be chosen so that the resulting closed-loop system is stable.

The simulation examples of this chapter have shown how time-varying algorithms can make the difference in terms of the monotonicity of the norm of tracking error. This phenomenon was explained by demonstrating that a time-invariant algorithm results in repeated closed-loop poles (along the iteration axis), resulting in a bad transient behaviour due to the terms  $k^{n-1}p_i^k$  in the solution of the tracking error, where  $p_i$  is a pole repeated  $n$  times. With a time-varying algorithm, by

contrast, it is possible to spread the poles in numerically different places in the complex plane, and therefore to remove the  $k^{n-1}$  term from the response. Note, however, that in the simulation example the time-varying learning-gain  $\gamma(t)$  was selected by resorting to an ‘educated guess’. In order to come up with something better in terms of algorithm tuning, in the following chapters several new (in some cases time-varying) ILC algorithms are derived, where the free parameters of an algorithm are selected by taking them to be a solution of a suitable optimisation problem.

## 4 Norm Optimal ILC

### 4.1 A review of earlier work on optimality and ILC

As was explained in Chapter 1, the fundamental idea of the ILC concept is to iteratively learn the input function that results in perfect tracking, at least asymptotically. In addition, it is preferable that the tracking error during each iteration is not too large, because a large tracking error can result in reduced product quality or even destroy the system in question. The definition of a large tracking error is highly application dependent, but a suitable norm can certainly be used to measure it. Therefore a natural way to approach the problem is to insist that the squared norm of the tracking error  $\|e_{k+1}\|^2$  is kept small during each iteration. This requirement, is an optimisation problem

$$\min_{u_{k+1} \in \mathcal{U}} \|e_{k+1}\|^2 \quad (4.1)$$

with the constraint equation  $e_{k+1} = r - Gu_{k+1}$ . In (4.1)  $\mathcal{U}$  is the set of allowed inputs. The issue here is that, if the reference signal belongs to the range of the plant, the solution of the optimisation problem is simply  $u_{k+1} = G^{-1}r$ . Hence the algorithm would converge in one iteration. In practice, however, the plant model  $G$  is never equal to the true plant, because of either unmodelled high-order dynamics or nonlinearities. Consequently, the cost function has to be modified somehow to increase the robustness of the algorithm. In the research literature, several different modifications exist. A short review of the existing methods is presented next to make it clear how the new methods derived in this thesis are related to the existing methods. Note that currently there are several hundred publications on ILC, and it is not clear how to give a balanced review of all these publications. Consequently the author has attempted to select a representative set of publications that are relevant to work reported in the later chapters of this thesis. For a more general review on ILC publications before 1998 see Amann (1996) and Moore (1998b).

### 4.1.1 Gradient methods for discrete-time plants

One of the earliest (possibly the first) publication that combines ILC and optimisation can be found in Togai & Yamano (1985). The publication considers only MIMO discrete-time plants. The authors propose the following optimisation problem

$$\begin{aligned} \min_{u(t)} J(u(t)) \\ J(u(t)) = \frac{1}{2}e_k(t+1)^T e_k(t+1) \end{aligned} \quad (4.2)$$

where the plant model (i.e. the constraint equation) is

$$\begin{aligned} x_k(t+1) &= Ax_k(t) + Bu_k(t) \\ y_k(t) &= Cx_k(t) \end{aligned} \quad (4.3)$$

and the tracking error is defined as  $e_k(t) := x_M(t) - x_k(t)$ , i.e. in terms of the state of the system. Furthermore,  $x_M(t)$  is generated by a reference model that has to be chosen by the algorithm designer. The authors suggest that this optimisation problem (4.2) can be solved iteratively by using a suitable optimisation method. Note that this approach is quite special in the sense that the idea is to minimise that tracking error for each time-instant, and not the tracking error of the whole trial ‘at one go’ (i.e.  $\|e_k\|^2$ ). In their paper the authors consider the steepest-descent method, the Newton-Raphson method and the Gauss-Newton method to iteratively solve this optimisation problem. For the steepest-descent method the optimal algorithm becomes

$$u_{k+1}(t) = u_k(t) + KB^T e_k(t+1) \quad (4.4)$$

where  $K$  is a constant matrix which defines the step size. The Newton-Raphson method on the other hand results in the algorithm

$$u_{k+1}(t) = u_k(t) + \frac{\|e_k(t+1)\|^2}{\|B^T e_k(t+1)\|^2} B^T e_k(t+1) \quad (4.5)$$

and the Gauss-Newton method results in

$$u_{k+1}(t) = u_k(t) + (B^T B)^{-1} B^T e_k(t+1) \quad (4.6)$$

The authors do not provide any results on how the different algorithms perform in terms of convergence or convergence speed. However, this publication can be seen as one of the first to investigate the possibility of combining ILC and optimisation. They also seem to be the first authors to realise that the ‘non-causal’ algorithm  $u_{k+1}(t) = u_k(t) + Ke_k(t+1)$  should be used in order to lift the plant model so that zero tracking error can be achieved for a large variety of LTI discrete-time plants.

### 4.1.2 Soft constraints in the control magnitude

In Tao *et al.* (1994) LTI discrete-time plants are considered. The authors propose to solve the minimisation problem

$$\begin{aligned} \min_{u_{k+1} \in \mathcal{U}} J(u_{k+1}) \\ J(u_{k+1}) = e_{k+1}^T Q e_{k+1} + u_{k+1}^T R u_{k+1} \end{aligned} \quad (4.7)$$

where the constraint equation is  $e_{k+1} = r - G u_{k+1}$ . The solution to this optimisation problem is

$$u_{k+1} = (R + G^T W G)^{-1} G^T Q r \quad (4.8)$$

and, after some algebraic manipulations, it can be shown than an equivalent implementation of (4.8) is given by

$$u_{k+1} = u_k + (R + G^T Q G)^{-1} G^T (e_k - R u_k) \quad (4.9)$$

In order to guarantee that the input function is smooth, the authors add a ‘soft constraint’ by modifying the cost function into

$$J(u_{k+1}) = e_{k+1}^T Q e_{k+1} + u_{k+1}^T R u_{k+1} + \Delta[u_{k+1}]^T R_2 \Delta[u_{k+1}] \quad (4.10)$$

where

$$\begin{aligned} \Delta u_{k+1}(t) &:= u_{k+1}(t+1) - u_{k+1}(t) \\ \Delta[u_{k+1}] &:= [u_{k+1}(0) \ \Delta u_{k+1}(1) \ \dots \ \delta u_{k+1}(N-1)]^T \end{aligned} \quad (4.11)$$

and  $N$  is the length of the trial. Note that the  $\Delta[u_{k+1}]$  corresponds to the derivative  $\frac{d}{dt} u_{k+1}(t)$  in the continuous-time case, and therefore the idea is to find an input signal that gives good tracking but is constrained in terms of amplitude and smoothness. The optimal input is derived in the paper, and it is not repeated here due to space limitations. The authors do not provide convergence analysis for this algorithms but equation (4.8) shows that the algorithm does not contain an internal model of the reference signal, and therefore the algorithm will not converge to zero tracking error. A similar observation holds for the algorithm with the soft constraint. Therefore, if perfect tracking is required, the cost function should be modified so that the optimal solution contains the internal model of the reference. A method for doing this will be shown in Section 4.2.

### 4.1.3 Steepest-descent method for continuous-time systems

In Togai & Yamano (1985) the possibility of applying the standard steepest-descent algorithm to ILC is analysed. The starting point is a continuous-time LTI system  $y(t) = [G u](t)$ . The standard steepest-descent algorithm is given by the formula (see Luenberger (1969))

$$u_{k+1} = u_k + \epsilon_{k+1} \delta_{k+1} \quad (4.12)$$

where  $\delta_{k+1}$  is a vector determining the direction of the update during iteration  $k+1$  and  $\epsilon_{k+1}$  is the step-length. In steepest-descent  $\delta_{k+1}$  is chosen to be the negative

gradient of the cost functional to be minimised, and in ILC, as explained above, the natural choice for cost function is  $J(u_{k+1}) = \|e_{k+1}\|^2$ ,  $e_{k+1} = r - Gu_{k+1}$ . This results in

$$u_{k+1} = u_k + \epsilon_{k+1} G^* e_k \quad (4.13)$$

At each trial the step-length is selected to minimise the tracking error during the next iteration, resulting in the following update law

$$\epsilon_{k+1} = \frac{\|G^* e_k\|^2}{\|GG^* e_k\|^2} \quad (4.14)$$

This algorithm will converge to zero tracking error for an arbitrary continuous-time LTI plant if  $G$  is stable, controllable and observable, and the convergence is monotonic. The update law, however, requires that a mathematical model of the plant exists, and consequently the authors also analyse the robustness properties of the algorithm. This work is not repeated here, but the reader is invited to study Chapter 7, where the convergence and robustness properties of this same algorithm and its 'robust modification' are analysed in the discrete-time case.

#### 4.1.4 Newton-Raphson method

In Gorinevsky (1992) the possibility of applying ILC to nonlinear systems was investigated. The starting point is a possibly nonlinear plant model

$$y = G(u) \quad (4.15)$$

where  $G(\cdot)$  is the (nonlinear) input-output mapping of the plant and the objective is to minimise the performance criterion

$$J = \|e\|^2 + \rho \|u\|^2 \quad (4.16)$$

If the plant is nonlinear but 'smooth enough', a small deviation  $\delta$  from the input  $u$  will result in

$$G(u_o + \delta) = G(u_o) + G'(u_o)\delta + O(\|\delta\|). \quad (4.17)$$

for a 'small'  $\delta$ , ( $G'$  is the Frechet derivative of  $G$ ) and the idea is to use this linearised model to minimise the cost function (4.16), resulting in

$$u_{k+1} = u_k - (\rho I + G'(u_k)^T G'(u_k))^{-1} (G'(u_k)^T e_k + \rho u_k) \quad (4.18)$$

Because  $G'(u_k)$  is not known, the authors suggest that an on-line identification routine is used to find an estimate for  $G'(u_k)$  based on previous trial data. The authors apply the algorithm to a two-link robot system, and the results are compared to a conventional inverse dynamics compensation controller. In a few iterations the learning controller achieves a smaller tracking error than the non-learning controller.

### 4.1.5 Constrained optimisation and ILC

A more recent work Gunnarsson & Norrlöf (2001), considers the cost function

$$J(u_{k+1}) = e_{k+1}^T W_e e_{k+1} + u_{k+1}^T W_u u_{k+1} \quad (4.19)$$

where the constraints are the plant equation  $y_{k+1} = G_e u_{k+1}$  (a discrete-time LTI model) and an inequality constraint  $(u_{k+1} - u_k)^T (u_{k+1} - u_k) \leq \delta$ .  $W_e$  and  $W_u$  are positive-definite and symmetric weighting matrices that can be used to find a balance between the tracking accuracy and the input function magnitude. The authors adjoin the inequality constraint into the cost function with a Langrange multiplier  $\lambda$ , which results in the modified cost function

$$\tilde{J}(u_{k+1}) = e_{k+1}^T W_e e_k + u_{k+1}^T W_u u_{k+1} + \lambda (u_{k+1} - u_k)^T (u_{k+1} - u_k) \quad (4.20)$$

and a straightforward differentiation exercise results in the following input update-law

$$u_{k+1} = Q(u_k + L e_k) \quad (4.21)$$

where

$$Q = (W_u + \lambda I + G_e^T W_e G_e)^{-1} (\lambda I + G_e^T W_e G_e) \quad (4.22)$$

and

$$L = (\lambda I + G_e^T W_e G_e)^{-1} G_e^T W_e \quad (4.23)$$

The authors suggest that the optimal value for  $\lambda$  is not solved explicitly, but it is used as a tuning parameter. Furthermore, they claim that the use of  $W_u > 0$  is useful, both for robustness reasons and when dealing with non-minimum phase systems. If  $W_u \neq 0$ , the algorithm does not contain the internal model of the reference signal, and therefore perfect tracking cannot be achieved. The authors apply the algorithm to an industrial robot, and after four iterations the algorithm has learned an input signal which results in near perfect tracking.

In summary, several researchers have suggested that optimisation is an efficient technique for designing ILC algorithms. However, the work neglects the fact that in order to achieve perfect tracking, the algorithm has to include an internal model of the reference signal, which is an integrator along the iteration axis. This can be achieved at least in two ways: the first way is to fix the structure of the algorithm to  $u_{k+1} = u_k + \delta_{k+1}$  and select  $\delta_{k+1}$  to minimise a suitable cost function, as is done for example in the steepest-descent approach in Section 4.1.3. The other possibility is to define a cost function where the optimising solution automatically includes the internal model of the reference signal. This is the approach taken in the next section, resulting in a rather general theory of Norm-Optimal Iterative Learning Control (NOILC).

## 4.2 Derivation and Convergence Analysis for Norm-Optimal ILC

In this section the concept of Norm-Optimal Iterative Learning Control (NOILC) is introduced and its mathematical properties are derived. The material presented in this section is based on Amann (1996). Only the main result in terms of convergence analysis will be presented. If the reader is interested in a more detailed analysis, he/she should consult the references Amann (1996) and Amann *et al.* (1998).

The starting point in NOILC is to write the plant model in the operator form

$$y = Gu + z_0 \quad (4.24)$$

where  $G$  is the system input-output (convolution) operator,  $u \in \mathcal{U}$  and  $y, z_0 \in \mathcal{Y}$  where  $\mathcal{Y}$  and  $\mathcal{U}$  are the input and output spaces respectively. In NOILC both  $\mathcal{U}$  and  $\mathcal{Y}$  are chosen to be real Hilbert spaces. Furthermore,  $G$  is assumed to be a linear and bounded operator from  $\mathcal{U}$  to  $\mathcal{Y}$ . In (4.24),  $z_0$  describes the effect of non-zero initial conditions on the plant output and, as was shown in Chapter 3, it can be assumed that  $z_0 = 0$  without loss of generality. As was explained earlier, the ultimate goal of any ILC algorithm is to solve in the limit the optimisation problem

$$\min_{u \in \mathcal{U}} \|e\|^2 \quad (4.25)$$

with the constraint equation  $e = r - Gu$ . From now on it is assumed that the reference signal belongs to the range of the plant  $G$ , and hence the optimising solution  $u^*$  satisfies  $e^* = r - Gu^* = 0$ .

Theoretically, if  $G$  and  $r$  are known, and  $r$  belongs to the range of  $G$ , and  $G$  is injective, the unique optimising solution  $u^*$  can be solved directly with the equation  $u^* = G^{-1}r$ . In practice, however,  $G$  is never known precisely, and alternative ways of solving the optimisation problem (4.25) have to be considered. In NOILC the optimal  $u^*$  is generated by solving the following *sequence* of optimisation problems

$$\min_{u_{k+1} \in \mathcal{U}} J_{k+1}(u_{k+1}) \quad (4.26)$$

where

$$J_{k+1}(u_{k+1}) = \|e_{k+1}\|^2 + \|u_{k+1} - u_k\|^2 \quad (4.27)$$

subject to the constraint equation  $e_{k+1} = r - Gu_{k+1}$ . The norms are assumed to be induced norms from the inner products  $\langle \cdot, \cdot \rangle_{\mathcal{Y}}$  in  $\mathcal{Y}$  and  $\langle \cdot, \cdot \rangle_{\mathcal{U}}$  in  $\mathcal{U}$  respectively, in other words  $\|e_{k+1}\|_{\mathcal{Y}}^2 = \langle e_{k+1}, e_{k+1} \rangle_{\mathcal{Y}}$  and  $\|u_{k+1} - u_k\|_{\mathcal{U}}^2 = \langle u_{k+1} - u_k, u_{k+1} - u_k \rangle_{\mathcal{U}}$ . The first term in (4.27) reflects the design criterion of  $\|e_{k+1}\|^2$  being small during every repetition. The second term penalises the input functions  $u_{k+1}$  that are ‘too’ different from the input  $u_k$ . There are at least two arguments that explain why this term should be useful in the cost function: the first argument is that the resulting algorithm will be of the form  $u_{k+1} = u_k + \delta_{k+1}$ , and hence the algorithm will contain the internal model of  $r_k = r$ , which is necessary for zero convergence according to Proposition 3.4 in Chapter 3. The second argument is that this term will result in an input sequence  $\{u_k\}$  that is smooth along the iteration axis  $k$ ,

which possibly results in a robust algorithm. Note that (4.27) is equal to (4.20) if  $W_u$  is selected to be zero in (4.20), which establishes an interesting connection between the work in Gunnarsson & Norrlöf (2001) and the work presented in this chapter. As a remark on the originality of the cost function (4.27), note that in Lee *et al.* (1996b) the same cost function was considered, and this publication was also published in 1996: therefore the idea of using this particular cost function can be traced back to at least two independent sources.

To get a flavour of the properties of the resulting algorithm, let  $u_{k+1}^*$  be the optimal solution for iteration round  $k + 1$  and  $e_{k+1}^*$  the corresponding optimal tracking error for iteration  $k + 1$ . With this notation the non-optimal choice  $u_{k+1} = u_k$  gives the following interlacing result

$$\|e_{k+1}^*\|^2 \leq J(u_{k+1}^*) \leq \|e_k^*\|^2 \quad (4.28)$$

and hence the algorithm will result in at least monotonic convergence, i.e.  $\|e_{k+1}^*\| \leq \|e_k^*\|$ . Note that for this interlacing result only the boundedness of  $G$  and the existence of a optimising solution  $u_{k+1}^*$  (which can be non-unique) is required, and linearity of  $G$  does not play any role. Hence it can be expected that the NOILC algorithm can also work for nonlinear plant models, and simulation work carried out in Amann (1996) supports this conjecture.

In order to show rigorously that the algorithm converges to the correct solution  $u^*$  in the linear case, the first step is to solve the optimisation problem (4.26). This achieved by calculating the Frechet derivative of (4.26) and setting it to zero, which results in a necessary condition for optimality. The Frechet derivative can be calculated from the equation

$$\langle J_{k+1}(u_{k+1}), \delta u_{k+1} \rangle = \frac{d}{d\epsilon} J_{k+1}(u_{k+1} + \epsilon \delta u_{k+1}) = 0 \quad (4.29)$$

In order to solve this equation, the adjoint operator  $G^*$  of  $G$  is needed in the calculations.  $G^*$  is defined by the equation

$$\langle v, Gu \rangle_{\mathcal{Y}} = \langle G^*v, u \rangle_{\mathcal{U}} \quad (4.30)$$

for an arbitrary  $v \in \mathcal{Y}$  and  $u \in \mathcal{U}$ , and it can be shown that  $G^*$  exists and is linear and bounded. The first term in  $J_{k+1}(u_{k+1} + \epsilon \delta u_{k+1})$  is given by

$$\begin{aligned} & \|r - G(u_{k+1} + \epsilon \delta u_{k+1})\|_{\mathcal{Y}}^2 \\ &= \langle r - G(u_{k+1} + \epsilon \delta u_{k+1}), r - G(u_{k+1} + \epsilon \delta u_{k+1}) \rangle_{\mathcal{Y}} \\ &= \langle e_{k+1} - \epsilon G \delta u_{k+1}, e_{k+1} - \epsilon G \delta u_{k+1} \rangle_{\mathcal{Y}} \\ &= \|e_{k+1}\|_{\mathcal{Y}}^2 - 2\epsilon \langle \delta u_{k+1}, G^* e_{k+1} \rangle_{\mathcal{U}} + \epsilon^2 \|G \delta u_{k+1}\|_{\mathcal{Y}}^2 \end{aligned} \quad (4.31)$$

and second term in  $J_{k+1}(u_{k+1} + \epsilon \delta u_{k+1})$  is given by

$$\|u_{k+1} + \epsilon \delta u_{k+1} - u_k\|_{\mathcal{U}}^2 = \langle u_{k+1}, u_{k+1} \rangle_{\mathcal{U}} - 2\epsilon \langle \delta u_{k+1}, u_{k+1} - u_k \rangle_{\mathcal{U}} + \epsilon^2 \|\delta u_{k+1}\|_{\mathcal{U}}^2 \quad (4.32)$$

Consequently the optimal solution  $u_{k+1}$  satisfies

$$\frac{d}{d\epsilon} J_{k+1}(u_{k+1} + \epsilon \delta u_{k+1}) = \langle \delta u_{k+1}, u_{k+1} - u_k - G^* e_{k+1} \rangle_{\mathcal{U}} = 0 \quad (4.33)$$

Because (the first) variation  $\delta u_{k+1}$  is an arbitrary element of  $\mathcal{U}$  (the optimisation problem is unconstrained with respect to  $u_{k+1}$ ), and  $\langle u, v \rangle = 0$  for an arbitrary  $u \in \mathcal{U}$  if and only if  $u = 0$  (a property of an inner product), the necessary condition for the optimal solution  $u_{k+1}$  is that it has to satisfy the equation

$$u_{k+1} = u_k + G^* e_{k+1} \quad (4.34)$$

This condition equation is also sufficient, because the second variation  $\frac{d^2}{d\epsilon^2} J_{k+1}(u_{k+1} + \epsilon \delta u_{k+1})$  is given by

$$\begin{aligned} \frac{d^2}{d\epsilon^2} J_{k+1}(u_{k+1} + \epsilon \delta u_{k+1}) &= \langle \delta u_{k+1}^T, (I + G^* G) \delta u_{k+1} \rangle_{\mathcal{U}} \\ &= \|\delta u_{k+1}\|_{\mathcal{U}}^2 + \|G \delta u_{k+1}\|_{\mathcal{U}}^2 > 0 \end{aligned} \quad (4.35)$$

for a non-zero  $\delta u_{k+1}$ . Hence the cost function is globally convex (high-order variations are zero), and  $u_{k+1} = u_k + G^* e_{k+1}$  is the optimising solution.

Multiplying (4.34) with  $G$  gives

$$y_{k+1} = y_k + GG^* e_{k+1} \quad (4.36)$$

and the multiplication of this equation with  $-1$  and the addition of  $r$  on both sides together with some algebraic manipulations results in

$$(I + GG^*) e_{k+1} = e_k \quad (4.37)$$

The operator  $I + GG^*$  in (4.37) is positive, because for an arbitrary non-zero  $v \in \mathcal{Y}$

$$\begin{aligned} \langle (I + GG^*)v, v \rangle_{\mathcal{Y}} &= \langle v, v \rangle_{\mathcal{Y}} + \langle GG^*v, v \rangle_{\mathcal{Y}} = \langle v, v \rangle_{\mathcal{Y}} + \langle G^*v, G^*v \rangle_{\mathcal{U}} \\ &= \|v\|_{\mathcal{Y}}^2 + \|G^*v\|_{\mathcal{U}}^2 \geq \|v\|_{\mathcal{Y}}^2 \end{aligned} \quad (4.38)$$

and hence, due to positivity,  $(I + GG^*)$  is invertible and the error evolution equation becomes

$$e_{k+1} = (I + GG^*)^{-1} e_k = L e_k \quad (4.39)$$

where  $L := (I + GG^*)^{-1}$  is the learning operator. Using the learning operator  $L$  the following proposition can be derived:

**Proposition 4.1** *Assume that  $r \in \mathcal{R}g(G)$  where  $\mathcal{R}g(G)$  is the range of the operator  $G$ . Then  $e_k$  in (4.39) satisfies  $\lim_{k \rightarrow \infty} \|e_k\| = 0$ .*

**Proof.** Due to optimality  $J_{k+1}$  is a monotonically decreasing sequence of positive numbers, and hence the limit  $\lim_{k \rightarrow \infty} J_k = J_{\infty}$  exists. The interlacing result  $\{\|e_k\|\}$  is also a monotonically decreasing sequence of positive numbers and, therefore,  $\lim_{k \rightarrow \infty} \|e_k\|$  exists. However,  $\|e_{k+1}\| = \|e_k\|$  if and only if  $\|u_{k+1} - u_k\| = 0$ . Hence  $\lim_{k \rightarrow \infty} \|u_{k+1} - u_k\| = 0$  and  $\lim_{k \rightarrow \infty} J_{k+1} = \lim_{k \rightarrow \infty} \|e_k\|^2$ . From  $\lim_{k \rightarrow \infty} \|u_{k+1} - u_k\| = 0$  it follows that

$$0 = \lim_{k \rightarrow \infty} \langle u, u_{k+1} - u_k \rangle_{\mathcal{U}} = \lim_{k \rightarrow \infty} \langle u, G^* e_{k+1} \rangle_{\mathcal{U}} = \lim_{k \rightarrow \infty} \langle Gu, e_{k+1} \rangle_{\mathcal{Y}} \quad (4.40)$$

i.e.  $\langle y, u_{k+1} - u_k \rangle_y = 0$  as  $k \rightarrow \infty$  for an arbitrary  $y \in \mathcal{R}g(G)$ . Using the operator  $\hat{L} := (I + GG^*)$ , the optimal value for the cost function  $J_{k+1}$  can be written in the form

$$J_{k+1} = \langle e_{k+1}, (I + GG^*)e_{k+1} \rangle = \left\langle \hat{L}^{-k-1}e_0, \hat{L}^{k+1}(I + GG^*)e_{2k+1} \right\rangle = \langle e_0, e_{2k+1} \rangle \quad (4.41)$$

If  $r \in \mathcal{R}g(G)$ , then  $e_0 = r - Gu_0 \in \mathcal{R}g(G)$ , and therefore  $e_0 = Gu$  for some  $u \in \mathcal{U}$ . This gives

$$J_{k+1} = \langle e_0, e_{2k+1} \rangle = \langle Gu, e_{2k+1} \rangle = \langle u, G^*e_{2k+1} \rangle \quad (4.42)$$

and applying (4.40) on (4.42) shows that  $\lim_{k \rightarrow \infty} J_{k+1} = \lim_{k \rightarrow \infty} \|e_k\| = 0$ .  $\square$

Hence, in this general case, the norm of the error will converge monotonically to zero. However, if  $GG^*$  is a positive operator, according to the following proposition the convergence is *geometric*:

**Proposition 4.2** *If the operator  $GG^*$  satisfies  $GG^* \geq \sigma^2 I$ , then the following estimate holds*

$$\|e_{k+1}\| \leq \frac{1}{1 + \sigma^2} \|e_k\| \quad (4.43)$$

and the norm of the tracking error converges geometrically to zero.

**Proof.** Taking the inner product between  $e_{k+1}$  and  $e_k$  and using (4.37) gives

$$\begin{aligned} \langle e_{k+1}, e_k \rangle &= \langle e_k, (I + GG^*)e_{k+1} \rangle = \langle e_{k+1}, e_{k+1} \rangle + \langle e_{k+1}, GG^*e_{k+1} \rangle \\ &\geq \langle e_{k+1}, e_{k+1} \rangle + \sigma^2 \langle e_{k+1}, e_{k+1} \rangle = (1 + \sigma^2) \|e_{k+1}\|^2 \end{aligned} \quad (4.44)$$

Further, from Cauchy-Schwarz,  $\langle e_{k+1}, e_k \rangle \leq \|e_k\| \|e_{k+1}\|$  resulting in the following estimate

$$\|e_{k+1}\| \|e_k\| \geq (1 + \sigma^2) \|e_{k+1}\|^2 \quad (4.45)$$

Dividing this equation with  $(1 + \sigma^2) \|e_{k+1}\|$  gives the estimate (4.43).  $\square$

The convergence analysis of the input sequence  $\{u_k\}$  is given by the following

**Proposition 4.3** *The input sequence  $\{u_k\}$  has the following property:*

$$\lim_{k \rightarrow \infty} \|G^*(r - Gu_k)\|_{\mathcal{U}} = 0 \quad (4.46)$$

If  $GG^*$  has a bounded inverse, the input sequence  $\{u_k\}$  converges in norm to  $u_\infty = (G^*G)^{-1}G^*r$ .

**Proof.** As was shown earlier,  $\{u_{k+1} - u_k\}$  converges in norm to zero. Furthermore,  $u_{k+1} = u_k + G^*e_{k+1}$  can be written as

$$u_{k+1} - u_k = G^*(r - Gu_{k+1}) \quad (4.47)$$

which in the limit gives (4.46). If  $G^*G$  has a bounded inverse, which is equivalent to the condition  $G^*G > \sigma^2 I$ , then  $u_\infty = (G^*G)^{-1}G^*r$ .  $\square$

**Remark 4.1** In Amann (1996) it has been shown that if the original plant is described using a continuous-time model,  $\sigma$  in Proposition 4.2 and Proposition 4.3 is equal to zero. This is because the space  $L[0, T]$  contains functions of ‘infinite frequency’, and these functions belong to the kernel of  $G$ . If the model is described using a discrete-time model, then  $\sigma$  is the smallest (non-zero) singular value of the matrix  $G_e$  defined in (3.14) in Chapter 3, and consequently in this case the convergence is geometric and the input sequence  $u_k$  converges to an unique vector that results in perfect tracking.

As was explained in the remark above, in the continuous-time case  $G^*G$  is not strictly positive (i.e. there does not exist  $\sigma^2 > 0$  so that  $G^*G \geq \sigma^2 I$ ), and it might happen that  $\{e_k\}$  converges to zero tracking error in the limit, but the input sequence  $u_k$  does not converge to a fixed point. This can be avoided by using the following ‘relaxed’ version

$$u_{k+1} = \alpha u_k + G^* e_{k+1} \quad (4.48)$$

where  $\alpha \in (0, 1)$ . With this algorithm the limit of the error sequence is not zero but rather

$$e_\infty = \left( I + \frac{GG^*}{1 - \alpha} \right)^{-1} r \quad (4.49)$$

and by taking a value of  $\alpha$  close but not equal to unity, an acceptable tracking accuracy will be achieved together with convergence for the input sequence  $\{u_k\}$ .

### 4.3 Causal implementation for continuous-time systems

Consider the continuous-time linear time-invariant plant model (possibly a MIMO system)

$$\begin{aligned} \dot{x}_k(t) &= Ax_k(t) + Bu_k(t) & x_k(0) &= x_0 \\ y_k(t) &= Cx_k(t) \end{aligned} \quad (4.50)$$

where  $t \in [0, T]$  and without loss of generality it is selected that  $x_0 = 0$ . With these assumptions, the input-output behaviour of the dynamical system (4.50) can be equivalently written as  $y(t) = [Gu](t)$  where  $G : \mathcal{U} \rightarrow \mathcal{Y}$  is defined with the formula

$$[Gu](t) = \int_0^t C e^{A(t-\tau)} B u(\tau) d\tau \quad (4.51)$$

The inner products are selected to be

$$\langle u_1, u_2 \rangle_{\mathcal{U}} = \int_0^T u_1^T(t) R u_2(t) dt \quad (4.52)$$

and

$$\langle y_1, y_2 \rangle_{\mathcal{Y}} = \int_0^T y_1^T(t) Q y_2(t) dt + y_1^T(T) F y_2(T) \quad (4.53)$$

where  $R, Q$  and  $F$  are symmetric positive definite matrices. These definitions, together with the constraint equation  $e(t) = r(t) - [Gu](t)$ , result in the following differential equation for the optimal  $u_{k+1}$

$$\begin{aligned} \dot{\psi}_{k+1}(t) &= -A^T \psi_{k+1}(t) - C^T Q e_{k+1}(t) & \psi_{k+1}(T) &= C^T F e_{k+1}(T) \\ u_{k+1}(t) &= u_k(t) + R^{-1} B^T \psi_{k+1}(t) \end{aligned} \quad (4.54)$$

This equation is anti-causal due to the terminal condition  $\psi_{k+1}(T) = C^T F e_{k+1}(T)$  and thus cannot be used to implement the algorithm. However, if it is guessed from the outset that a causal implementation of (4.54) is given by the equation

$$u_{k+1}(t) = u_k(t) - R^{-1} B^T [K(t)(x_{k+1}(t) - x_k(t)) - \xi_{k+1}(t)] \quad (4.55)$$

It can be shown that this implementation is equivalent to (4.54) if  $K(t)$  satisfies the Riccati-equation

$$\dot{K}(t) = -A^T K(t) + K(t)A + K(t)BR^{-1}B^T - C^T QC, \quad K(T) = C^T FC \quad (4.56)$$

and  $\xi_{k+1}(t)$  satisfies the differential equation

$$\dot{\xi}_{k+1}(t) = -(A - BR^{-1}B^T K(t))^T \xi_{k+1}(t) - C^T Q e_k(t) \quad (4.57)$$

which is driven by tracking error from trial  $k$ . In summary the algorithm consists of the following steps:

- 1) Select suitable values for  $R$  and  $Q$  and simulate  $K(t)$  in reverse time using equation (4.56).
- 2) Select an initial guess  $u_0$  and feed into the real plant. Record the input  $u_0(\cdot)$ , the corresponding tracking error  $e_0(\cdot)$ , and state  $x_0(\cdot)$  and set the iteration index  $k$  equal to one.
- 3) Solve the predictive term  $\xi_k(\cdot)$  in reverse time using equation (4.57).
- 4) Run an experiment by using the control law (4.60). Record the input  $u_k(\cdot)$ , corresponding tracking error  $e_k(\cdot)$ , and state  $x_k(\cdot)$ . Set  $k \rightarrow k + 1$  and go to step 3.

**Remark 4.2** *It is important to note that in the implementation of the algorithm the Riccati equation for  $K(t)$  has to be solved only once, whereas the predictive term  $\xi_k(t)$  has to be solved between each trial via numerical integration. Furthermore, the implementation requires a full knowledge of the state of the system and, if this is not available, a state observer has to be added to the algorithm.*

**Remark 4.3** *The inner product (4.53) might at first sight look slightly exotic. However, the addition of the term  $y_1^T F y_2$  guarantees that the algorithm results in uniform zero tracking error even if  $CB = 0$ , i.e. the relative degree of the plant is more than one. See Amann (1996) and Amann et al. (1998) for details.*

#### 4.4 Causal implementation for discrete-time systems

Consider now the following discrete-time, linear, time-invariant system (possibly a MIMO system)

$$\begin{aligned} x_{k+1}(t+1) &= \Phi x_{k+1}(t) + \Gamma u_{k+1}(t) & x_{k+1}(0) &= x_0 \\ y(t) &= Cx(t) \end{aligned} \quad (4.58)$$

where  $t \in [0, 1, \dots, T]$  (i.e. without loss of generality unity sampling time is assumed) and again without loss of generality it is assumed that  $x_0 = 0$ . As in the previous section, it can be shown that the optimal input  $u_{k+1}$  is generated by the differential equation

$$\begin{aligned} p_{k+1}(t) &= \Phi^T p_{k+1}(t+1) + C^T Q e_{k+1}(t+1), & p_{k+1}(N) &= 0 \\ u_{k+1}(t) &= u_k(t) + R^{-1} \Gamma^T p_{k+1}(t) \end{aligned} \quad (4.59)$$

which is clearly non-causal. However, resorting to a similar technique as in the previous section, it can be shown that an equivalent causal implementation exists, and is given by the equation

$$u_{k+1}(t) = u_k(t) - (R + \Gamma^T K(t) \Gamma)^{-1} \Gamma^T K(t) \Phi [x_{k+1}(t) - x_k(t)] + R^{-1} \Gamma^T \xi_{k+1}(t) \quad (4.60)$$

where  $K(t)$  and  $\xi_{k+1}(t)$  are generated by the equations

$$\begin{aligned} K(t) &= \Phi^T K(t+1) (I + \Gamma R^{-1} \Gamma^T K(t+1))^{-1} \Phi + C^T Q C \\ \xi_{k+1}(t) &= (I + K(t) \Gamma R^{-1} \Gamma^T)^{-1} (\Phi^T \xi_{k+1}(t+1) + C^T Q e_k(t+1)) \end{aligned} \quad (4.61)$$

where the boundary conditions are  $K(N) = 0$  and  $\xi_{k+1}(N) = 0$ . The steps needed in the running of the algorithm are equivalent to those of the continuous-time case of the previous section. The causal implementation for the relaxed algorithm (4.48) has been reported in Amann (1996) both for the continuous-time case and discrete-time case, and it is not repeated here due to space limitations.

#### 4.5 Predictive Norm Optimal Iterative Learning Control

So far in this chapter the standard NOILC algorithm has been introduced and its convergence properties have been analysed. However, intuition suggests that faster convergence might be achieved if future tracking errors and future input differences are also added to the cost function. This has already been done in Amann *et al.* (1998) as part of a predictive receding horizon control design:

$$J_{k+1,n}(\vec{u}_{k+1}) = \sum_{i=1}^n \lambda^{i-1} (\|e_{k+1,i}\|^2 + \|u_{k+1,i} - u_{k+1,i-1}\|^2) \quad (4.62)$$

where  $\vec{u}_{k+1} := [u_{k+1,1} \ u_{k+1,2} \ \dots \ u_{k+1,n}]$ . Here  $u_{k+1,j}$  refers to the input for iteration  $k+j$  calculated during iteration  $k+1$  and  $e_{k+1,j}$  refers to the (predicted)

tracking error for iteration  $k + j$  calculated during iteration  $k + 1$ . Furthermore, in the following material the definition  $\vec{e}_{k+1} := [e_{k+1,1} \ e_{k+1,2} \ \dots \ e_{k+1,n}]$  is used. Finally,  $u_{k+1}$  refers to the input control fed into the real plant during iteration  $k + 1$  and  $e_{k+1} = r - Gu_{k+1}$  is the resulting tracking error from this particular choice of input function.

The proposed criterion (4.62) includes the error not only from the current trial but also the predicted error from the next  $n - 1$  trials ( $n$  is known as the prediction horizon) as well as the corresponding changes in the input. The weight parameter  $\lambda > 0$  determines the importance of more distant (future) errors and incremental inputs. Furthermore, just as in Generalized Predictive Control Camacho & Bordons (1998), a receding horizon principle is proposed in Amann *et al.* (1998). In this approach, at trial  $k + 1$  only the input  $u_{k+1,1}$  is used as an input into the plant, and the optimization is repeated again at the next trial.

By including more future signals into the performance criterion (4.62), it is argued by the authors that the algorithm should become less ‘short sighted’, and faster convergence should be obtained when compared to non-predictive algorithm resulting from the minimisation of (4.26). This was rigorously proved in Amann *et al.* (1998) when the input  $u_{k+1,1}$  is used as the ‘true’ input. In addition, the resulting algorithm from (4.62) has a causal implementation if the original plant can be described with a state-space representation.

In this section, it is noted that it is possible to use the receding horizon principle for any other input  $u_{k+1,j}$  or, more generally, the implemented control input function can be taken as a positive convex combination of the inputs  $u_{k+1,j}$ . It is then an important question whether or not a higher convergence rate can be achieved with these choices when compared to the choice where  $u_{k+1,1}$  is used as the input. To provide preliminary support for the idea, it is of interest to examine the receding horizon control algorithm obtained from the minimisation of (4.62) and in implementation of the control  $u_{k+1} = u_{k+1,j}$ . It follows from optimality that

$$\lambda^{j-1} \|e_{k+1,j}\|^2 \leq \min_{\vec{u}_{k+1}} J_{k+1,j}(\vec{u}_{k+1}) \leq \|u_\infty - u_k\|^2 \quad (4.63)$$

where  $u_\infty$  is the control input that generates the reference  $r$  exactly. The first inequality is a direct consequence of the positivity of the terms in  $J_{k+1,j}$ . The second inequality follows from optimality and the choice of the input sequence  $u_{k+1,l} = u_\infty$ ,  $1 \leq l \leq n$ . Equation (4.63) provides some insight into the effect of the choice of  $\lambda$  and  $n$  via rearrangement

$$\|e_{1,j}\|^2 \leq \frac{\|u_\infty - u_0\|^2}{\lambda^{j-1}} \quad (4.64)$$

That is, at the first trial, increasing  $\lambda > 1$  or  $j$  leads to an increased reduction in the tracking error. In particular, this result suggests that the choice of  $j = n$  and  $\lambda > 1$  may lead to the smallest tracking error. As  $n \rightarrow \infty$ , the error becomes zero. For fixed  $n > 1$ , the error goes to zero as  $\lambda \rightarrow \infty$ . The main theme in the following sections is to analyse in detail the relationship between convergence speed, the weighting parameter  $\lambda$  and the prediction horizon  $n$ .

## 4.6 Derivation of the error evolution equations

In order to analyse the effect of  $\lambda$  and  $n$  on convergence speed, the first step is to establish the error evolution equation between  $e_{k+1,j}$  and  $e_k$  when using the input  $u_{k+1,j}$  for  $j = 1, 2, \dots, n$ . It is a straightforward task (see Amann *et al.* (1998)) to show that in matrix form the optimising solution of (4.62) is given by

$$N_{n,e}(\lambda)\vec{u}_{k+1} = \tilde{u}_k + G_e^* \vec{e}_{k+1} \quad (4.65)$$

where  $\tilde{u}_k = [0 \ 0 \ \dots \ 0 \ u_k]^T$  and  $N_{n,e}(\lambda)$  is given by

$$N_{n,e}(\lambda) = \begin{bmatrix} I & -I & 0 & \dots & 0 \\ -\lambda I & I + \lambda I & -I & \dots & 0 \\ 0 & -\lambda I & I + \lambda I & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & -I \\ 0 & 0 & \dots & -\lambda I & I + \lambda I \end{bmatrix} \quad (4.66)$$

and  $G_e^*$  is given by

$$G_e^* = \begin{bmatrix} G^* & 0 & 0 & \dots & 0 \\ 0 & G^* & 0 & \dots & 0 \\ 0 & 0 & G^* & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & G^* \end{bmatrix} \quad (4.67)$$

Based on (4.65), the error evolution equation can be written as

$$L_{n,e}(H, \lambda) \begin{bmatrix} e_{k+1,n} \\ e_{k+1,n-1} \\ e_{k+1,n-2} \\ \vdots \\ e_{k+1,2} \\ e_{k+1,1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ e_k \end{bmatrix} \quad (4.68)$$

where

$$L_{n,e}(H, \lambda) = \begin{bmatrix} I + H & -I & 0 & \dots & 0 \\ -\lambda I & S(\lambda, H) & -I & \dots & 0 \\ 0 & -\lambda I & S(\lambda, H) & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & -I \\ 0 & 0 & \dots & -\lambda I & S(\lambda, H) \end{bmatrix} \quad (4.69)$$

and  $S(\lambda, H) := I + \lambda I + H$ . Since the elements in (4.69) commute with each other

and are linear and bounded, and both  $I + H$  and  $I + \lambda I + H$  are strictly positive operators, and therefore invertible, the elements of (4.69) behave in algebraic sense as the elements of a field. Therefore the inverse of  $L_{n,e}(H, \lambda)$  can be calculated with Cramer's rule MacLane & Birkoff (1971) in the following way

$$L_{n,e}(H, \lambda)^{-1} = (\det(L_{n,e}(H, \lambda)))^{-1} \text{adj}(L_{n,e}(H, \lambda)) \quad (4.70)$$

However, because the interest in this section is to find the relationship between  $e_{k+1,j}$  and  $e_k$  it can be seen that the operator mapping  $e_k$  to  $e_{k+1,j}$  is given by

$$\det(L_{n,e}(H, \lambda))^{-1} (-1)^{2n-j+1} M(H, \lambda) \quad (4.71)$$

for  $j = 1, 2, \dots, n$  where

$$M(H, \lambda) = |L_{n,e}(H, \lambda)^{(n, n-j+1)}| \quad (4.72)$$

and  $|L_{n,e}(H, \lambda)^{(n, n-j+1)}|$  is the minor of  $L_{n,e}(H, \lambda)$  at position  $(n, n-j+1)$ . These minors can be used to establish the following

**Proposition 4.4** *The functional dependence from  $e_k$  to  $e_{k+1,j}$  is given by*

$$e_{k+1,j} = p_n(H, \lambda)^{-1} p_{n-j}(H, \lambda) e_k \quad (4.73)$$

where the operator  $p_j(H, \lambda)$  is defined by the recursive formula

$$\begin{aligned} p_j(H, \lambda) &= (I + \lambda + H)p_{j-1}(H, \lambda) \\ &\quad - \lambda p_{n-2}(H, \lambda) \end{aligned} \quad (4.74)$$

and the initial condition for the recursion is  $p_0(H, \lambda) = 0$  and  $p_1(H, \lambda) = (I + H)$ .

**Proof.** As a preliminary remark, it is observed that  $L_{n,e}(H, \lambda)$  can be written recursively as

$$L_{n,e}(H, \lambda) = \begin{bmatrix} & & & & 0 \\ & & & & 0 \\ & & & & \vdots \\ & & L_{n-1,e}(H, \lambda) & & -I \\ 0 & 0 & \dots & -\lambda I & S(\lambda, H) \end{bmatrix} \quad (4.75)$$

or even as

$$L_{n,e}(H, \lambda) = \begin{bmatrix} & & & & 0 & 0 \\ & & & & 0 & 0 \\ & & & & \vdots & \vdots \\ & & L_{n-2,e}(H, \lambda) & & -I & 0 \\ 0 & 0 & \dots & -\lambda I & S(\lambda, H) & -I \\ 0 & 0 & \dots & 0 & -\lambda I & S(\lambda, H) \end{bmatrix} \quad (4.76)$$

From (4.75) and (4.76) it is seen that

$$\begin{aligned} \det(L_{n,e}(H, \lambda)) &= (I + \lambda I + H) \det(L_{n-1,e}(H, \lambda)) \\ &\quad - \lambda \det(L_{n-2,e}(H, \lambda)) \end{aligned} \quad (4.77)$$

with  $\det(L_{0,e}(H, \lambda)) = I$  and  $\det(L_{1,e}(H, \lambda)) = I + H$ . To shorten the notation, the  $n$ 'th operator obtained from the iteration (4.77) is denoted as  $p_n(H, \lambda)$ , i.e.  $p_n(H, \lambda) := \det(L_{n,e}(H, \lambda))$ . It is known from Amann *et al.* (1998) that  $p_n(H, \lambda)$  for  $n = 1, 2, 3, \dots$  are bounded, self-adjoint, strictly positive, linear operators and thus it is mathematically correct to use (4.70) to calculate the inverse of  $L_{n,e}(H, \lambda)$ .

To calculate the functional dependence from  $e_k$  to  $e_{k+1,1}$  the cofactor of  $L_{n,e}(H, \lambda)$  is needed from position  $(n, n)$ . But (4.77) points out that it is  $\det(L_{n-1,e}(H, \lambda)) = p_{n-1}(H, \lambda)$ . Hence

$$e_{k+1,1} = p_n(H, \lambda)^{-1} p_{n-1}(H, \lambda) e_k \quad (4.78)$$

A similar argument shows that for  $e_{k+1,j}$ ,  $j \in \{1, 2, \dots, n\}$

$$e_{k+1,j} = p_n(H, \lambda)^{-1} p_{n-j}(H, \lambda) e_k \quad (4.79)$$

establishing the function dependence between  $e_{k+1,j}$  and  $e_k$ .  $\square$

As the error evolution equation between  $e_{k+1,j}$  and  $e_k$  is known it is now possible to discover if the inputs  $u_{k+1,j}$  for  $j = 2, 3, \dots, n$  or, more generally, convex combinations of the inputs, result in faster convergence than that observed when  $u_{k+1,1}$  alone is used. This is done in the next section by carefully analysing the properties of the family of learning operators given in (4.74).

## 4.7 Partial orderings for the learning operators

In this section partial orderings are introduced for the operators

$L_{n,j}(H, \lambda) = p_n(H, \lambda)^{-1} p_{n-j}(H, \lambda)$  for  $j \in \{1, \dots, n\}$  to demonstrate the effect of the prediction horizon  $n$  and weighting factor  $\lambda$  on the rate of convergence in the following way Kreyszig (1978): let  $T_1$  and  $T_2$  be bounded linear self-adjoint operators on a complex Hilbert space  $Y$ . Then  $T_1 \geq T_2$  if and only if  $\langle T_1 e, e \rangle \geq \langle T_2 e, e \rangle$  (or  $T_1 > T_2$  if and only if  $\langle T_1 e, e \rangle > \langle T_2 e, e \rangle$ ) for an arbitrary non-zero element  $e \in Y$ . Another important concept from functional analysis in this section is a positive operator Kreyszig (1978): a bounded self-adjoint linear operator  $T$  is positive if and only if  $T \geq 0$  (or  $T$  is strictly positive if and only if  $T > 0$ ). This implies that  $T_1 \geq T_2$  if and only if  $T_1 - T_2 \geq 0$  (or  $T_1 > T_2$  if and only if  $T_1 - T_2 > 0$ ).

In this section it will be shown how the partial orderings are related to convergence rate. The partial orderings demonstrate that by using a positive convex combination of the inputs  $\{u_{k+1,1} \dots u_{k+1,n}\}$  (i.e. using the input  $u_{k+1} := \sum_{i=1}^n \alpha_i u_{k+1,i}$ , where  $\sum_{i=1}^n \alpha_i = 1$  and  $\alpha_i \geq 0$ ) as the true input for the plant, a quicker convergence rate can be obtained than using merely  $u_{k+1} = u_{k+1,1}$  as the true input. In addition, the quickest convergence rate is achieved with the choice

$u_{k+1} = u_{k+1,n}$ . Furthermore, the convergence is geometric for a positive convex combination if for  $\forall e \in Y, e \neq 0, \langle He, e \rangle \geq \sigma^2 \|e\|^2$  where  $\sigma^2 > 0$  and almost geometric if  $\sigma^2 = 0$ .

Most of the proofs are based on a standard result in functional analysis Kreyszig (1978): namely that, if two bounded self-adjoint linear operators  $S$  and  $T$  on a complex Hilbert space  $Y$  are positive and commute, then their product  $ST = TS$  is positive.

The analysis begins by collecting from Amann *et al.* (1998) the following useful properties of the family of learning operators  $L_n(H, \lambda) = L_{n,1}(H, \lambda)$ :

**Property 4.1** *Assume that  $H \geq 0, \|H\| < \infty, 0 < \lambda < \infty$ . Then  $L_n(\lambda, H) > 0, n = 0, 1, 2, \dots$*

From this on it always assumed that  $\|H\| < \infty$ , because  $G$  is supposed to be a linear dynamical system defined over a *finite-time*, and hence it is bounded, implying  $H = GG^*$  is bounded.

**Property 4.2** *Assume that  $H \geq 0, \|H\| < \infty, 0 < \lambda < \infty$ . Then  $0 < L_{n+1}(H, \lambda) < L_n(H, \lambda) < I$  for  $H > 0, \lambda > 0, n = 0, 1, 2, \dots$ , and the operators  $L_n(H, \lambda)$  are self-adjoint and commute with each other for  $n = 0, 1, 2, \dots$*

**Property 4.3** *Assume that  $H > 0, 0 < \lambda < \lambda'$ . Then  $L_n(H, \lambda) > L_n(H, \lambda'), n = 1, 2, 3, \dots$*

The first step is to show that a similar partial ordering exists as in Property 4.2 for the operators  $L_{n,i}(H, \lambda)$  as a function of  $i$ .

**Proposition 4.5** *Assume that  $H > 0$ . Then  $L_n = L_{n,1} > L_{n,2} > \dots > L_{n,n-1} > L_{n,n}$ .*

**Proof.** First note that for  $j \in \{1, \dots, n\}$

$$\begin{aligned} L_{n,j} &= p_n^{-1} p_{n-j} \\ &= p_n^{-1} p_{n-1} p_{n-1}^{-1} \dots p_{n-j+1} p_{n-j+1}^{-1} p_{n-j} \\ &= L_n L_{n-1} \dots L_{n-j+1} \end{aligned} \quad (4.80)$$

Furthermore  $L_{n,k} = L_n L_{n-1} \dots L_{n-k+1} > 0$  and  $I - L_{n-k} > 0$  based on Property 4.2. Thus  $L_{n,k} > L_n L_{n-1} \dots L_{n-k+1} L_{n-k} = L_{n,k+1}$  for  $k \in \{1, \dots, n-1\}$ .  $\square$

This result suggests that by including the information from the inputs  $u_{k+1,j}$ ,  $j = \{2, \dots, n\}$  in the calculation of the current input signal quicker convergence can be achieved when compared to the case where merely  $u_{k+1}$  is used. One interesting approach to achieve this inclusion would be to select the current input as a positive convex combination of the input signals, i.e.  $u_{k+1} = \sum_{j=1}^n \alpha_j u_{k+1,j}$ , where  $\alpha_j \geq 0, \sum_{j=1}^n \alpha_j = 1$ . In this case the error evolution is governed by the following equation

$$\begin{aligned} e_{k+1} &= p_n^{-1} (\alpha_1 p_{n-1} + \alpha_2 p_{n-2} + \dots + \alpha_n p_0) e_k \\ &= (\alpha_1 L_{n,1} + \alpha_2 L_{n,2} + \dots + \alpha_n L_{n,n}) e_k \\ &= L_{n,c}(H, \lambda, \boldsymbol{\alpha}) e_k \end{aligned} \quad (4.81)$$

where  $L_{n,c}(H, \lambda, \boldsymbol{\alpha})$  reflects the fact that the properties of the new learning operator  $L_{n,c}(H, \lambda, \boldsymbol{\alpha})$  are also dependent on the choice of  $\boldsymbol{\alpha} := \{\alpha_1, \dots, \alpha_n\}$ . The next

proposition shows that by taking an arbitrary convex combination of  $\alpha_i$ , a learning operator  $L_{n,c}(H, \lambda, \alpha)$  is obtained which lies between  $L_{n,1}(H, \lambda)$  and  $L_{n,n}(H, \lambda)$  in the sense of the partial ordering.

**Proposition 4.6** *Assume that  $H > 0$ . Then  $L_n > L_{n,c} > L_{n,n}$  if  $L_{n,c} \neq L_n$  and  $L_{n,c} \neq L_{n,n}$ .*

**Proof.**

$$\begin{aligned} L_{n,n} &= \sum_{i=1}^n \alpha_i L_{n,i} < \sum_{i=1}^n \alpha_i L_{n,i} = L_{n,c} \\ &< \sum_{i=1}^n \alpha_i L_{n,1} = L_{n,1} = L_n \end{aligned} \quad (4.82)$$

□

The next step is to analyse whether or not  $\lambda$  introduces a partial ordering for the learning operator. This question is answered in the following proposition:

**Proposition 4.7** *Let  $\lambda' > \lambda$ ,  $\lambda', \lambda > 0$  and  $H > 0$ . Then  $L_{n,c}(H, \lambda', \alpha) < L_{n,c}(H, \lambda, \alpha)$ .*

**Proof.** Take an arbitrary  $j \in \{1, \dots, n-1\}$ . From Property 4.3 it is known that  $L_{n-j}(H, \lambda') < L_{n-j}(H, \lambda)$ . Thus  $L_{n-j+1}(H, \lambda)L_{n-j}(H, \lambda) > L_{n-j+1}(H, \lambda)L_{n-j}(H, \lambda')$ . However, because  $L_{n-j+1}(\lambda, H) - L_{n-j+1}(H, \lambda') > 0$ , it implies that

$$\begin{aligned} L_{n-j}(H, \lambda')L_{n-j+1}(H, \lambda) &> \\ L_{n-j}(H, \lambda')L_{n-j+1}(\lambda', H) & \end{aligned} \quad (4.83)$$

Thus

$$\begin{aligned} L_{n-j+1}(H, \lambda)L_{n-j}(H, \lambda) &> \\ L_{n-j+1}(H, \lambda')L_{n-j+1}(H, \lambda') & \end{aligned} \quad (4.84)$$

This argument is repeated inductively and thus

$$\begin{aligned} L_n(H, \lambda)L_{n-1}(H, \lambda) \dots L_{n-j}(H, \lambda) &> \\ L_n(H, \lambda')L_{n-1}(H, \lambda') \dots L_{n-j}(H, \lambda') & \end{aligned} \quad (4.85)$$

and consequently

$$\begin{aligned} L_{n,c}(H, \lambda, \alpha) &= \sum_{i=1}^n \alpha_i L_{n,i}(H, \lambda) > \\ \sum_{i=1}^n \alpha_i L_{n,i}(H, \lambda') &= L_{n,c}(H, \lambda', \alpha) \end{aligned} \quad (4.86)$$

which completes the proof. □

The next step is to analyze whether or not the prediction horizon  $n$  introduces a partial ordering for the operators. However, because for an arbitrary set of  $\{\alpha_1, \dots, \alpha_n\}$  it is not clear how to select a new set of  $\{\alpha'_1, \dots, \alpha'_{n+1}\}$  so that the comparison would make logical sense, a special case is considered where for a fixed  $\alpha \in \mathbb{R}_+$  and an arbitrary  $n \in \mathbb{N}_+$  it is always selected that  $\alpha_1 = \alpha$ ,  $\alpha_i = 0$  for  $i \in \{2, \dots, n-1\}$  and  $\alpha_n = 1 - \alpha$ , i.e.  $L_{n,c}(H, \lambda, \alpha) = \alpha L_1(H, \lambda) + (1 - \alpha)L_n(H, \lambda)$ . This specific selection gives the following proposition:

**Proposition 4.8** *Assume that  $H > 0$ . Then  $L_{n+1,c}(H, \lambda, \alpha) < L_{n,c}(H, \lambda, \alpha)$ .*

**Proof.** Because  $I - L_{n+1}(H, \lambda) > 0$ , this implies that

$$\frac{L_n(H, \lambda)L_{n-1}(H, \lambda) \dots L_1(H, \lambda)}{L_{n+1}(H, \lambda)L_n(H, \lambda) \dots L_1(H, \lambda)} > 0 \quad (4.87)$$

and hence

$$\begin{aligned} L_{n,c}(H, \lambda, \alpha) &= \alpha L_{n,1}(H, \lambda) + (1 - \alpha)L_{n,n}(H, \lambda) \\ &> \alpha L_{n+1,1}(H, \lambda) + (1 - \alpha)L_{n+1,n+1}(H, \lambda) \\ &= L_{n+1,c}(H, \lambda, \alpha) \end{aligned} \quad (4.88)$$

because based on Property 4.2,  $L_{n,1}(H, \lambda) = L_n(H, \lambda) > L_{n+1}(H, \lambda) = L_{n+1,1}(H, \lambda)$ . The rest of this section shows how the partial orderings are related to the convergence properties of the learning operator. As a starting point the following proposition is needed from Amann *et al.* (1998):

**Proposition 4.9** *Let  $\sigma \geq 0$  be such that  $\langle e, He \rangle \geq \sigma^2 \langle e, e \rangle \forall e \in Y$ . Then*

$$L_n(H, \lambda) \leq l_n(\sigma, \lambda) \quad \forall \lambda > 0, n = 0, 1, 2, \dots \quad (4.89)$$

where

$$l_{n+1}(\sigma, \lambda) = \frac{1}{1 + \sigma^2 + \lambda - \lambda l_n(\sigma, \lambda)}, \quad l_0 = 1 \quad (4.90)$$

and the error sequence is bounded by  $\|e_{k+1}\| \leq l_n(\sigma, \lambda)\|e_k\|$ .

Note that if  $\sigma > 0$  (which holds for the important case of discrete-time systems) then  $0 < l_n(\sigma, \lambda) < 1$ ,  $n \in \{1, 2, \dots\}$  but if  $\sigma = 0$  then  $l_n(\sigma, \lambda) = 1$  for  $n \in \{1, 2, \dots\}$ . For a bounded self-adjoint operator  $T$  on a complex Hilbert-space  $Y$  it is valid that  $\|T\| = \sup_{\|x\|=1} |\langle Tx, x \rangle|$  implying that  $\|L_n(H, \lambda)\| \leq l_n(\sigma, \lambda)$ . Thus  $L_n(H, \lambda)$  is a contraction if  $\sigma > 0$ . Furthermore in this case each Proposition from 4.5 to 4.8 can be rewritten so that  $L_n(H, \lambda)$  is replaced with  $l_n(\sigma, \lambda)$  giving the following qualitative conclusions:

- i) The lowest convergence rate is achieved by using  $u_{k+1,1}$  as an true input and the quickest convergence rate is achieved by using  $u_{k+1,n}$  as the true input. By using a convex combination of the inputs the convergence rate lies between the convergence rate given by  $u_{k+1,1}$  and  $u_{k+1,n}$ . The convergence is geometric in each case.
- ii) Increasing  $\lambda$  increases the convergence rate for an arbitrary positive convex combination of  $\{u_{k+1,1} \dots u_{k+1,n}\}$ .
- iii) If the coefficients  $\alpha_i$  of  $L_{n,c}(H, \lambda, \alpha)$  are selected as in Proposition 4.8 then  $\|L_{n+1,c}(H, \lambda, \alpha)\| < \|L_{n,c}(H, \lambda, \alpha)\|$ . This shows that by increasing the prediction horizon a quicker convergence rate is achieved.

The underlying problem, however, is that for a strictly proper continuous-time system the lower bound in Proposition 4.9 is in fact zero (an operator having a strictly positive lower bound on an infinite-dimensional space  $Y$  cannot be compact but  $H$  is a compact operator). Thus in the previous propositions and properties in this section  $<$  has to be replaced with  $\leq$  when  $H \geq 0$ . In particular,  $\|L_{n,c}(H, \lambda, \alpha)\| \leq 1$ , and thus strict geometric convergence is not possible to achieve. Consequently it would be interesting to know which error functions are left unaffected by  $L_{n,c}(H, \lambda, \alpha)$ . This question is answered in the following proposition for  $L_n(H, \lambda)$ :

**Proposition 4.10**  $L_n(H, \lambda)e = e$  if and only if  $e \in \text{Ker}(H)$ .

**Proof.** Suppose that  $e \in \text{Ker}(H)$ . Then  $L_0(H, \lambda) = Ie = e$ . Assume now that  $e \in \text{Ker}(H)$  implies that  $L_n(H, \lambda)e = e$ . Then

$$L_{n+1}(H, \lambda)e = (I + \lambda I + H - \lambda L_n(H, \lambda))^{-1} e \quad (4.91)$$

or (because  $L_{n+1}(H, \lambda)$  is invertible and the operators commute)

$$L_{n+1}(H, \lambda)(I + \lambda I + H - \lambda L_n(H, \lambda))e = e \quad (4.92)$$

but based on the induction assumption and the fact that  $He = 0$

$$L_{n+1}(H, \lambda)e = e \quad (4.93)$$

Assume now that there exists  $e \in Y$  such that  $L_1(H, \lambda)e = (I + H)^{-1}e = e$ . This shows that  $(I + H)e = e$  implying  $He = 0$  and hence  $e \in \text{Ker}(H)$ . Assume now that there exists  $e \in Y$  such that  $L_n(H, \lambda)e = e$ . This implies that  $\langle L_n(H, \lambda)e, e \rangle = \langle e, e \rangle = \|e\|^2$ . However, Property 4.2 shows that  $I \geq L_1(H, \lambda) \geq L_n(H, \lambda)$  showing that

$$\langle e, e \rangle \geq \langle L_1(H, \lambda)e, e \rangle \geq \langle L_n(H, \lambda)e, e \rangle = \langle e, e \rangle \quad (4.94)$$

which implies that  $\langle L_1(H, \lambda)e, e \rangle = \langle e, e \rangle$  and hence  $L_1(H, \lambda)e = e$  and  $e \in \text{Ker}(H)$ .  $\square$

The extension to  $L_{n+1,c}(H, \lambda, \alpha)$  is straightforward: from Proposition 4.10 it is clear that if  $e \in \text{Ker}(H)$  then  $L_{n,c}(H, \lambda, \alpha)e = e$ . On the other it is known from Proposition 4.6 that  $I \geq L_1(H, \lambda) \geq L_{n,c}(H, \lambda, \alpha)$ . Thus, if  $L_{n,c}(H, \lambda, \alpha)e = e$ , a similar argument as in Proposition 4.10 shows that  $L_1(H, \lambda)e = e$ , implying  $e \in \text{Ker}(H)$ . Furthermore, it can be shown that  $\text{Ker}(H) = \text{Ker}(G^*) = (\text{Rg}(G))^\perp$ . Thus it is seen that if  $r \notin \text{Ker}(G^*) = (\text{Rg}(G))^\perp$  then the algorithm converges even when  $\sigma^2 = 0$  in Proposition 4.9. The nature of this convergence is shown in the next section.

## 4.8 Almost geometric convergence

As mentioned in the previous section, geometric convergence cannot be achieved for a strictly proper continuous-time plant because  $H$  is compact operator in an infinite-dimensional space. However, as  $H$  is a compact self-adjoint operator on a Hilbert space  $Y$ , then  $H$  has the following spectral decomposition (see Kreyszig (1978))

$$He = \sum_j \sigma_j^2 \langle e, \psi_j \rangle \psi_j \quad (4.95)$$

where  $\{\sigma_j^2\}$  are the eigenvalues of  $H$  and  $\{\psi_j\}$  are the corresponding eigenvectors. In addition,  $\sigma_i^2 \rightarrow 0$  and 0 belongs to the spectrum of  $H$ . Furthermore, if  $Y$  is separable and infinite-dimensional (for example  $L_2[0, T]$ ) then the eigenvectors

form a complete orthonormal basis for  $Y$ . This means that an arbitrary vector  $e \in Y$  can be written as  $e = \sum_j \langle e, \psi_j \rangle \psi_j$ . Thus, when using the standard algorithm (i.e.  $u_{k+1,1}$  is the true input), the evolution of the error signal can be written as

$$e_{k+1} = L_n(H, \lambda)e_k = \sum_j l_n(\sigma_j, \lambda) \langle e_k, \psi_j \rangle \psi_j \quad (4.96)$$

where  $l_n(\sigma_j, \lambda)$  is obtained from the iteration (4.90). If a convex combination of the inputs is used as the true input for the plant, then the error evolution is governed by

$$e_{k+1} = L_{n,c}(H, \lambda)e_k = \sum_j (\alpha_1 l_{n,1}(\sigma_j, \lambda) + \alpha_2 l_{n,2}(\sigma_j, \lambda) + \dots + \alpha_{n-1} l_{n,n-1}(\sigma_j, \lambda) + \alpha_n l_{n,n}(\sigma_j, \lambda)) \langle e_k, \psi_j \rangle \psi_j \quad (4.97)$$

where

$l_{n,j}(\sigma_i, \lambda) = l_n(\sigma_i, \lambda) l_{n-1}(\sigma_i, \lambda) \dots l_{n-j+1}(\sigma_i, \lambda)$  for  $j \in \{1, \dots, n\}$ . In the standard case where the true input is  $u_{k+1,1}$  the following proposition from Amann *et al.* (1998) shows the ‘almost geometric convergence’ property.

**Proposition 4.11** *For any  $\epsilon > 0$ , let  $m$  be an integer such that the approximation error satisfies  $\|e_0 - \sum_{j=1}^m \langle e_0, \psi_j \rangle \psi_j\| < \epsilon$ . Then the following bound holds:*

$$\|e_k\| = l_n(\sigma_m, \lambda)^k \|e_0\| + \epsilon \quad (4.98)$$

If the true input is a convex combination of  $\{u_{k+1,1} \dots u_{k+1,n}\}$  a modification of Proposition 4.11 shows that the equality in (4.98) can be replaced with

$$\|e_k\| \leq \left( \sum_{i=1}^n \alpha_i l_{n,i}(\sigma_m, \lambda) \right)^k \|e_0\| + \epsilon \quad (4.99)$$

Thus if  $r(t) \notin R_g(G)^\perp$  then the previous propositions guarantee almost geometric convergence. Furthermore, the convergence is dependent on the projection of  $r(t)$  on the eigenvectors of  $L_n(H, \lambda)$  (note that  $L_{n,c}(H, \lambda, \alpha)$  has the same eigenvectors as  $L_n(H, \lambda)$ ) in the sense that if  $r(t)$  can be approximated accurately with eigenvectors related to large eigenvalues of  $L_n(H, \lambda)$  the convergence is faster than when compared to the case where the approximation  $r(t)$  requires also eigenvectors related to small eigenvalues of  $L_n(H, \lambda)$ . Furthermore, if  $G$  is a dynamical system having a low-pass filter characteristic, it can be argued that the eigenvectors related to large eigenvalues correspond to ‘low-frequency’ signals, whereas the eigenvectors related to small eigenvalues correspond to ‘high-frequency signals’. With this interpretation, error convergence is slower at high frequencies, but this is normally the range where the reference  $r$  has only a small amplitude. Slow convergence at high frequencies is hence not a substantial practical problem.

## 4.9 A causal implementation of the predictive algorithm

The causal implementation of the receding-horizon predictive algorithm has already been done in Amann (1996) and Amann *et al.* (1998) for the continuous-time case. In this section this derivation is shortly reviewed and it is also shown how the receding horizon implementation has to be modified in order to cover the new convex combination case. As a starting point, the following extended plant matrices are defined

$$\begin{aligned} A_n &= \text{diag}\{A, A, \dots, A\}, \quad B_n = \text{diag}\{B, B, \dots, B\}, \\ C_n &= \text{diag}\{C, C, \dots, C\} \end{aligned} \quad (4.100)$$

with  $n$  entries. Furthermore, the following extended weighting matrices are needed:

$$Q_n = \text{diag}\{Q, \lambda Q, \dots, \lambda^{n-1}Q\}, \quad F_n = \text{diag}\{F, \lambda F, \dots, \lambda^{n-1}F\} \quad (4.101)$$

and

$$R_n = \begin{bmatrix} (1 + \lambda)R & -\lambda R & 0 & \dots & 0 & 0 \\ -\lambda R & (\lambda + \lambda^2)R & -\lambda^2 R & \dots & 0 & 0 \\ 0 & -\lambda^2 R & (\lambda^2 + \lambda^3)R & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \lambda^{n-2} & (\lambda^{n-2} + \lambda^{n-1})R & -\lambda^{n-1}R \\ 0 & 0 & 0 & \dots & -\lambda^{n-1}R & \lambda^{n-1}R \end{bmatrix} \quad (4.102)$$

With these definitions the next step is to use a similar method as with the standard algorithm, i.e. it is guessed that the causal algorithm

$$\Delta \vec{u}_{k+1}(t) = -R_n^{-1} B_n^T \left( K_n(t) \Delta \vec{x}_{k+1}(t) - \vec{\xi}_{k+1}(t) \right) \quad (4.103)$$

where

$$\Delta \vec{u}_{k+1} := \begin{bmatrix} u_{k+1,1}(t) \\ u_{k+1,2}(t) \\ \vdots \\ u_{k+1,n}(t) \end{bmatrix} - \begin{bmatrix} u_k(t) \\ u_k(t) \\ \vdots \\ u_k(t) \end{bmatrix} \quad (4.104)$$

and

$$\Delta \vec{x}_{k+1} := \begin{bmatrix} x_{k+1,1}(t) \\ x_{k+1,2}(t) \\ \vdots \\ x_{k+1,n}(t) \end{bmatrix} - \begin{bmatrix} x_k(t) \\ x_k(t) \\ \vdots \\ x_k(t) \end{bmatrix} \quad (4.105)$$

is equivalent to the non-causal algorithm (4.65). This is true if the feedback gain matrix satisfies the following Riccati differential equation

$$\dot{K}_n = -A_n^T K_n - K_n A_n + K_n B_n R_n^{-1} B_n^T K_n - C_n^T Q_n C_n; \quad K_n(T) = C_n^T F_n C_n \quad (4.106)$$

and the predictive term  $\vec{\xi}_{k+1}$  is generated by the differential equation

$$\dot{\vec{\xi}}_{k+1} = -(A_n - B_n R_n^{-1} B_n^T K_n) \vec{\xi}_{k+1} - C_n^T Q_n \begin{bmatrix} e_k \\ e_k \\ \vdots \\ e_k \end{bmatrix} \quad (4.107)$$

with the terminal condition  $\vec{\xi}_{k+1}(T) = C_n^T F_n [e_k^T(T) \ e_k^T(T) \ \dots \ e_k^T(T)]^T$ . As was explained earlier, in the standard receding horizon approach only the first input vector is fed into the real plant, i.e.  $u_{k+1} = u_{k+1,1}$ , and the rest of the input signals are ‘virtual’ inputs that only exist inside the ILC controller. In the new approach proposed in this section, by contrast, a convex combination of the inputs  $u_{k+1}, \dots, u_{k+n}$  is calculated, i.e.  $u_{k+1} = \sum_{i=1}^n \alpha_i u_{k+1,i}$ , and the resulting input function is fed into the real plant. Fig. 4.1 shows the flow diagram of the modified predictive NOILC algorithm. In the flow diagram the block ‘C’ takes inside the input vector  $\vec{u}_{k+1}$  and calculates the convex combination  $u_{k+1} = \sum_{i=1}^n \alpha_i u_{k+1,i}$  which is fed into the ‘real plant’. Note that the ‘plant models’ in the diagram are mathematical models of the true plant and they reside inside a computer that implements the modified predictive NOILC algorithm.

## 4.10 A numerical example

Consider a SISO-system

$$(p^2 + 5p + 6)y(t) = (p + 1)u(t) \quad (4.108)$$

where  $p := \frac{d}{dt}$ . This system is sampled with sampling time  $T_s = 0.1$  seconds using a zero-order hold and the reference signal is  $r(t) = e^{t/20} \sin(t)$ ,  $t \in [0, 0.1, \dots, 20]$ .

### 4.10.1 The effect of $\alpha_i$

The prediction horizon of the norm-optimal algorithm was taken to be  $n = 5$  and the weighting parameter  $\lambda = 2$ . The algorithm was run using three different approaches: the first approach was the previously used receding-horizon approach, i.e.  $u_{k+1} = u_{k+1,1}$ . The second approach used a convex combination  $u_{k+1} = \sum_{i=1}^5 \alpha_i u_{k+1,i}$  where  $\alpha_i = 0.2$ . The third approach was selected to be  $u_{k+1} = u_{k+1,5}$ . Fig. 4.2 shows the singular values of the learning operators as function of ‘frequency’ (or strictly speaking as function of the vector  $v_i$  related to the singular value  $\sigma_i$ ), which is a  $200 \times 200$  matrix. From this figure it is clear that the smallest singular values are obtained with the choice  $u_{k+1} = u_{k+1,5}$ , the slowest with  $u_{k+1} = u_{k+1,1}$ , and the convex combination  $u_{k+1} = \sigma_{i=1}^n 0.2 u_{k+1,i}$  gives something in between. Furthermore, the maximum benefit from using a



and the convex combination  $u_{k+1} = \sum_{i=1}^n 0.2u_{k+1,i}$  gives something in between.

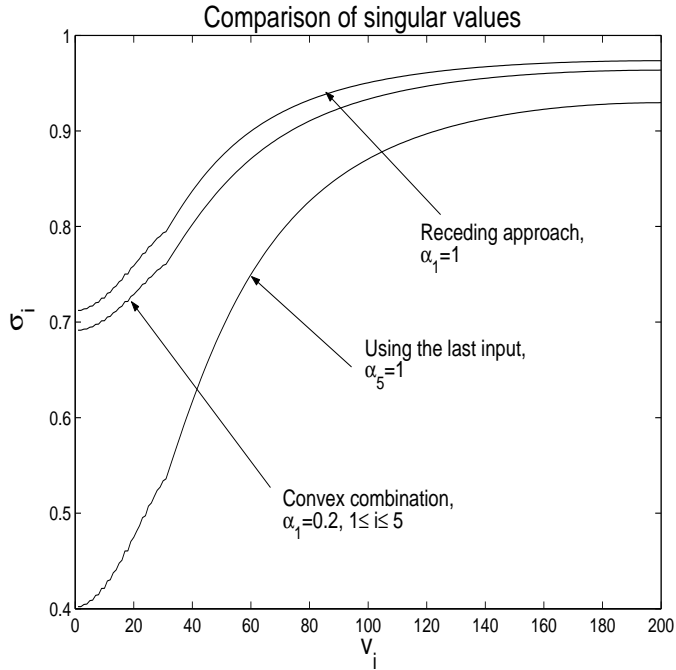


Figure 4.2. Comparison of singular values for different values of  $\alpha_i$ .

#### 4.10.2 The effect of weighting parameter $\lambda$

In order to investigate the effect of  $\lambda$  on convergence, take  $\lambda = 1$ ,  $\lambda = 2$  and  $\lambda = 5$  in the convex combination case  $u_{k+1} = \sum_{i=1}^n 0.2u_{k+1,i}$ . Fig. 4.4 shows the singular values of the learning operator for the different values of  $\lambda$ , and as expected, an increase in  $\lambda$  will decrease the amplitude of the singular values.

Fig. 4.5, on the other hand, shows the norm of the tracking error as a function of the iteration round. Again, as expected, an increase in  $\lambda$  will increase the convergence speed.

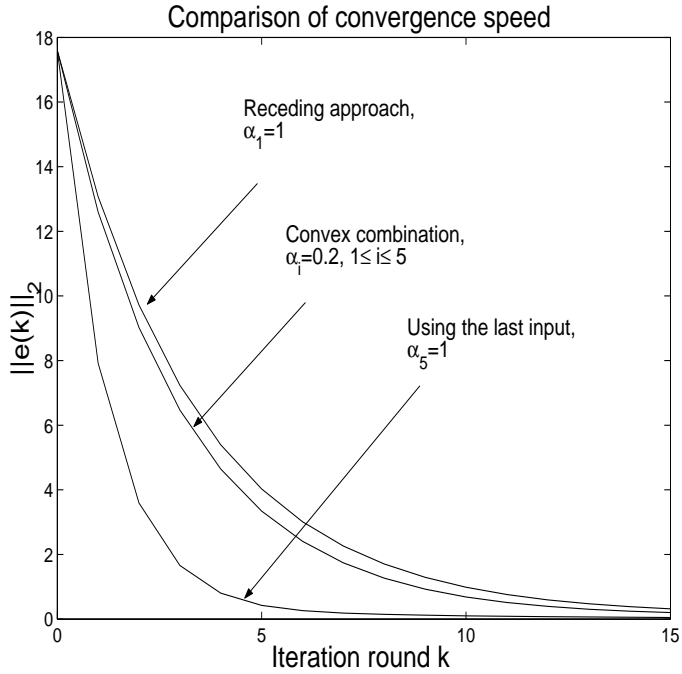


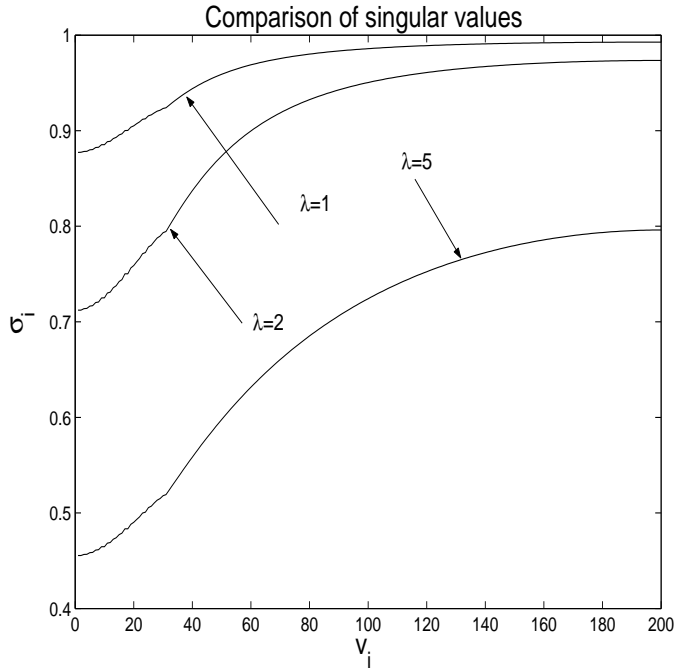
Figure 4.3. Comparison of convergence speed for different values of  $\alpha_i$ .

#### 4.10.3 Effect of prediction horizon $n$

Consider now the choice  $u_{k+1} = 0.5u_{k+1,1} + 0.5u_{k+1,n}$  for  $n = 2, 3, 4, 5$  with  $\lambda = 2$ . Fig. 4.6 shows again an expected result, i.e. an increase in the prediction horizon decreases the magnitude of the singular values. Fig. 4.7 shows the corresponding tracking behaviour where an increase in the prediction horizon increases the convergence speed.

### 4.11 Summary

In this section NOILC was introduced as one possible way of approaching Iterative Learning Control. The benefits from using either the non-predictive or predictive version are clear: the tracking error will converge monotonically to zero tracking error for an arbitrary LTI plant. Furthermore, if the plant is a discrete-time plant, the convergence is *geometric*. The derivations in this chapter were made assuming that there are no uncertainties in the plant model. This is done in order to understand how the algorithm performs in the nominal case, which has to be

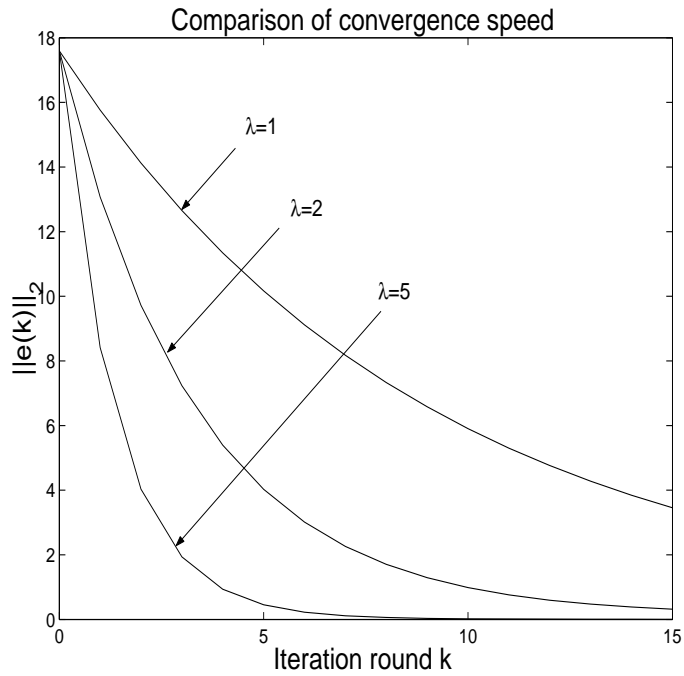


**Figure 4.4.** Comparison of singular values for different values of weighting parameter  $\lambda$ .

starting point for any further analysis. Any robustness analysis is expected to be complex.

Note, however, that in the discrete-time case the learning operator  $L_{n,j}$  is a matrix. Furthermore, in the absence of uncertainty the eigenvalues of  $L_{n,j}$  are inside the unit circle. It is a well-known result that the eigenvalues of a matrix are continuous functions of the elements of the matrix. Therefore, even if the plant parameters are perturbed slightly, the eigenvalues of the learning operator will remain inside the unit circle, resulting at least in asymptotic convergence. In summary, in the discrete-time case it is guaranteed that there is a certain amount of robustness available against parametric uncertainty in the plant model, but how to quantify this in a useful design framework is still an open problem for future research.

As was shown in this chapter, the fastest convergence speed is achieved by using the last input  $u_{k+1,n}$  as the input that is fed into the real plant. However, this should be the least robust choice because it relies most on the predictions of the plant model. Therefore the convex combination approach proposed in this chapter should provide a straightforward mechanism to find a balance between convergence speed and robustness.



**Figure 4.5.** Comparison of convergence speed for different values of weighting parameter  $\lambda$ .

The main disadvantage of NOILC is the rather complex real-time implementation of the algorithm (especially in the predictive case). Therefore in the next chapters on ILC the main theme is to find algorithms that are easy to implement but at the same time these algorithms should result at least in monotonic convergence.

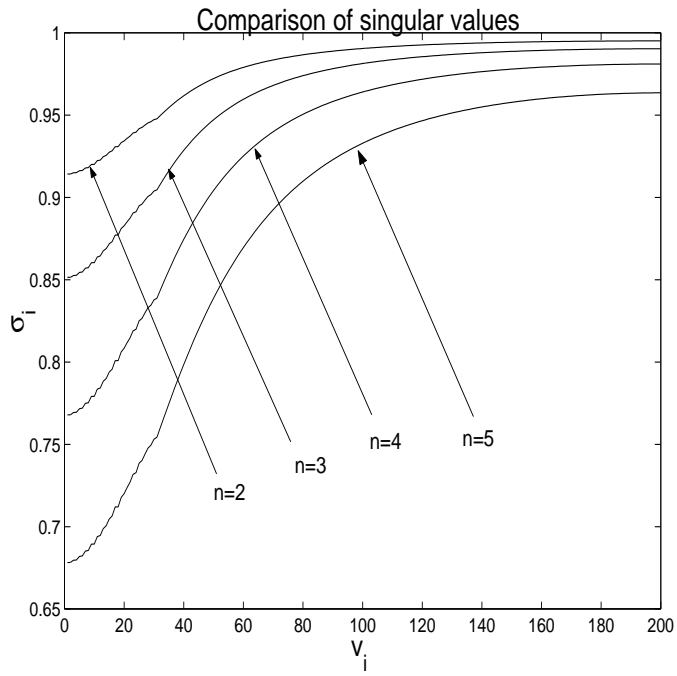


Figure 4.6. Comparison of singular values as function of the prediction horizon  $n$ .

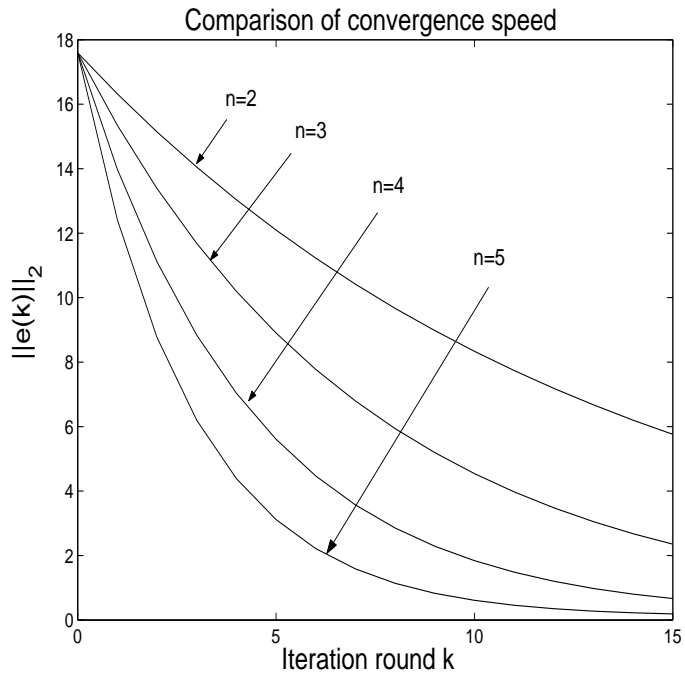


Figure 4.7. Comparison of convergence speed as function of the prediction horizon  $n$ .

## 5 Parameter Optimal ILC

In the previous section the possibility of using optimisation techniques in ILC were discussed, and it was shown how the NOILC approach and its predictive modification result in geometric convergence for an arbitrary linear (possibly time-varying) plant. The implementation of the different versions of the algorithm is, however, a non-trivial task. This is due to the fact that both the feedback and feedforward term are generated by a time-varying noncausal dynamical system, and they have to be solved by using numerical integration. In addition, the feedback term requires a full knowledge of the state of the system, which either requires instrumentation that can measure the states (possibly resulting in an expensive measurement setup) or the inclusion of a state-observer into the NOILC algorithm. Consequently it is an important question whether or not there exists structurally (in terms of implementation and instrumentation) simpler algorithms that would still result at least in monotonic convergence. A natural starting point for answering this question is the Arimoto-type ILC algorithm

$$u_{k+1}(t) = u_k(t) + \gamma e_k(t+1) \quad (5.1)$$

because it is a feedforward algorithm (i.e. does not require any additional instrumentation or other 'real-time components') and the corrective term is simply a learning gain multiplied with the shifted tracking error from the previous trial. The algorithm is applied on the SISO-plant model

$$\begin{cases} x_{k+1}(t+1) = \Phi x_{k+1}(t) + \Gamma u_{k+1}(t), & x_{k+1}(0) = 0 \\ y_{k+1} = C x_{k+1}(t) \end{cases} \quad (5.2)$$

and it is assumed that  $\Gamma C \neq 0$  and  $t \in [0, N]$ . As was shown in Chapter 3, the algorithm (5.1) will result in asymptotic convergence to zero tracking-error when applied to the plant model (5.2) if  $|1 - \gamma C \Gamma| < 1$ . However, monotonic convergence would be preferable, but it is not clear how to select  $\gamma$  so that this would be achieved. One possibility is to borrow ideas from the NOILC algorithm that was proposed in the previous chapter, where in the discrete-time case during each iteration the optimisation problem

$$\begin{aligned} \min_{u_{k+1} \in \mathcal{U}} J(u_{k+1}) \\ J(u_{k+1}) = \|e_{k+1}\|^2 + \|u_{k+1} - u_k\|^2 \end{aligned} \quad (5.3)$$

has to be solved under the constraint  $y_{k+1} = G_e u_{k+1}$  where  $G_e$  is the lifted plant model (3.14) from Chapter 3 for (5.2), i.e.

$$G_e = \begin{bmatrix} C\Gamma & 0 & 0 & \dots & 0 \\ C\Phi\Gamma & C\Gamma & 0 & \dots & 0 \\ C\Phi^2\Gamma & C\Phi\Gamma & C\Gamma & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C\Phi^{N-1}\Gamma & C\Phi^{N-2}\Gamma & \dots & \dots & C\Gamma \end{bmatrix} \quad (5.4)$$

and

$$\begin{aligned} u_k &:= [u_k(0) \ u_k(1) \ \dots \ u_k(N-1)]^T \\ y_k &:= [y_k(1) \ y_k(1) \ \dots \ y_k(N)]^T \end{aligned} \quad (5.5)$$

In the algorithm (5.1), the norm of the input difference is  $\|u_{k+1} - u_k\| = \gamma^2 \|e_k\|^2$ , and hence the input difference is proportional to the learning gain  $\gamma$ . Therefore, there is a possibility that the algorithm

$$u_{k+1}(t) = u_k(t) + \gamma_{k+1} e_{k+1}(t+1) \quad (5.6)$$

where  $\gamma_{k+1}$  is selected to be the solution of the optimisation problem

$$\begin{aligned} \min_{\gamma_{k+1} \in \mathbb{R}} J_{k+1}(\gamma_{k+1}) \\ J_{k+1}(\gamma_{k+1}) = \|e_{k+1}\|^2 + w\gamma_{k+1}^2 \end{aligned} \quad (5.7)$$

with  $w > 0$  would have similar convergence properties as the NOILC algorithm (the constraint equation is again  $y_{k+1} = G_e u_{k+1}$ ). In the next section this optimisation problem is solved and the convergence properties of the resulting algorithm are analysed. Furthermore, new variants of this basic algorithm are derived, which result in a class of ILC algorithms that in this thesis are termed as the class of Parameter Optimal Iterative Learning Control (POILC) algorithms.

## 5.1 The feedforward algorithm

The results in this section are taken from Owens & Feng (2002) and Owens & Feng (2003). As a starting point the update law for the optimal  $\gamma_{k+1}$  is given in the following

**Proposition 5.1** *The solution of the optimisation problem (5.7) is given by*

$$\gamma_{k+1} = \frac{e_k^T G_e e_k}{w + e_k^T G_e^T G_e e_k} \quad (5.8)$$

**Proof.** Using the error evolution equation  $e_{k+1} = (I - \gamma_{k+1} G_e) e_k$  in (5.7) the cost function can be written as

$$J_{k+1}(\gamma_{k+1}) = e_k^T (I - \gamma_{k+1} G_e)^T (I - \gamma_{k+1} G_e) e_k + w\gamma_{k+1}^2 \quad (5.9)$$

Differentiating this equation with respect to  $\gamma_{k+1}$  (a necessary condition for optimality) and setting the equation to zero gives

$$\frac{1}{2} \frac{dJ_{k+1}(\gamma_{k+1})}{d\gamma_{k+1}} = -e_k^T G_e e_k + \gamma_{k+1} (w + e_k^T G_e^T G_e e_k) = 0 \quad (5.10)$$

and solving for  $\gamma_{k+1}$  results in (5.8). Because the second derivative of  $J(\gamma_{k+1})$  is  $e^T G_e^T G_e e_k > 0$ , the cost function is convex and therefore the optimal  $\gamma_{k+1}$  is given by (5.8)  $\square$

The next proposition shows that the algorithm results in monotonic convergence:

**Proposition 5.2** *The algorithm (5.1) where  $\gamma_{k+1}$  is calculated using (5.8) results in monotonic convergence, i.e.  $\|e_{k+1}\| \leq \|e_k\|$  and  $\|e_{k+1}\| = \|e_k\|$  if and only if  $\gamma_{k+1} = 0$ .*

**Proof.** Let  $\gamma_{k+1}^*$  be the optimal learning gain for iteration  $k+1$ . The non-optimal choice  $\gamma_{k+1} = 0$  results in  $\|e_{k+1}\|^2 + w(\gamma_{k+1}^*)^2 = J(\gamma_{k+1}^*) \leq J_{k+1}(0) = \|e_k\|^2$ . Furthermore,  $\|e_{k+1}\|^2 \leq \|e_{k+1}\|^2 + w(\gamma_{k+1}^*)^2$ , which implies  $\|e_{k+1}\|^2 \leq \|e_k\|^2$ . This inequality also shows that  $\|e_{k+1}\| = \|e_k\|$  if and only if  $\gamma_{k+1} = 0$ .  $\square$

The next proposition characterises the behaviour of the sequence of optimal learning gains  $\{\gamma_k^*\}$  in the limit:

**Proposition 5.3** *The sequence of optimal learning gains  $\{\gamma_k^*\}$  generated by (5.8) satisfies  $\lim_{k \rightarrow \infty} \gamma_k^* = 0$ .*

**Proof.** Take an arbitrary  $k \in \mathbb{N}_+$ . The interlacing result in the previous proposition shows that  $0 \leq \|e_{k+1}\|^2 \leq \|e_k\|^2 - w\gamma_{k+1}^{*2}$ . Applying induction on this inequality results in  $0 \leq \|e_0\|^2 - w \sum_{i=1}^{k+1} (\gamma_i^*)^2$ . Because  $k$  is arbitrary, this implies that  $\lim_{k \rightarrow \infty} \gamma_k^* = 0$ .  $\square$

These two propositions result in the following converge result for the sequence of tracking errors  $\{e_k\}$ :

**Proposition 5.4** *Consider the algorithm (5.1) with the update law (5.8) for  $\gamma_{k+1}$ . Suppose that  $G_e + G_e^T$  is positive-definite, in other words  $\exists \sigma \in \mathbb{R}$ ,  $\sigma > 0$  so that  $v^T G_e v = v^T (\frac{G_e + G_e^T}{2}) v \geq \sigma v^T v$  for an arbitrary  $v \in \mathbb{R}^N$ . Then the resulting error sequence satisfies  $\lim_{k \rightarrow \infty} e_k = 0$ .*

**Proof.** Note that if  $e_k \neq 0$ , the positivity of  $G_e$  implies

$$\gamma_{k+1} = \frac{e_k^T G_e e_k}{w + e_k^T G_e^T G_e e_k} \geq \sigma \frac{e_k^T e_k}{w + e_k^T G_e^T G_e e_k} > 0 \quad (5.11)$$

However, because according to Proposition 5.3  $\lim_{k \rightarrow \infty} \gamma_k = 0$ , inequality (5.11) implies that  $\lim_{k \rightarrow \infty} e_k = 0$ .  $\square$

**Remark 5.1** Note that if the matrix  $G_e + G_e^T$  is negative-definite, a similar analysis as in Proposition 5.4 shows that the algorithm will converge monotonically to zero tracking error. However, if  $G_e + G_e^T$  is negative-definite, it can be always made positive-definite by multiplying  $G_e$  with  $-1$ . Therefore in this chapter (without loss of generality) the statement ' $G_e + G_e^T$  is not a positive-definite matrix' means that  $G_e + G_e^T$  is either a semi-definite or a non-definite matrix.

If  $G_e^T + G_e$  is not a positive-definite matrix, the following proposition shows that the algorithm can converge to a non-zero solution:

**Proposition 5.5** Assume that  $G_e^T + G_e$  is not a positive-definite matrix. Then it is possible for the algorithm (5.1) to converge to a non-zero tracking error.

**Proof.** If  $G_e + G_e^T$  is not a positive-definite matrix, there exists a non-zero vector  $v$  so that  $v^T G_e v = 0$ . Take  $u_0$  so that  $v = r - G_e u_0$  (such  $u_0$  exists because  $G_e$  is invertible) and obviously  $e_0 = r - G_e u_0$ . The update law (5.8) implies that  $u_1 = u_0$ , and therefore  $e_1 = e_0$ , and by induction  $e_k = e_0$  for  $k \geq 1$ , showing that algorithm has converged to a non-zero tracking error.  $\square$

Consequently zero tracking error in the limit is guaranteed when  $G_e + G_e^T$  is a positive-definite matrix, but in other cases the limiting tracking error can be a non-zero vector. A more detailed convergence analysis of this algorithm can be found in Owens & Feng (2002) and Owens & Feng (2003).

**Remark 5.2** Assume that  $G_e + G_e^T$  is not positive-definite. If the algorithm produces a tracking error  $e_k$  so that  $e_k^T (G_e + G_e^T) e_k = 0$ , then  $u_{k+1} = u_k$  and  $e_{k+1} = e_k$ . Therefore the algorithm will always converge to a tracking error  $e_\infty \in S$  where  $S = \{v | v^T (G_e + G_e^T) v = 0\}$ .  $v = 0$  is obviously a member of this set but the set also contains elements  $v \neq 0$ .

## 5.2 Predictive algorithm

Consider now the following predictive feedback algorithm

$$u_{k+1}(t) = u_k(t) + \gamma e_{k+1}(t+1) \quad (5.12)$$

where  $\gamma \in \mathbb{R}$ ,  $\gamma > 0$ . This implementation of the algorithm is non-causal, but by using the state-space model  $(\Phi, \Gamma, C)$  from (5.2) that generates  $G_e$ , the algorithm can be written as

$$u_{k+1}(t) = M^{-1} \{u_k(t) + \gamma r(t+1) - \gamma C \Phi x_{k+1}(t)\} \quad (5.13)$$

where  $M = (1 + \gamma C \Gamma)^{-1}$ . Note that other more advanced prediction methods could be used, Model Predictive Control and Kalman Filtering being two possible options. The corresponding error evolution equation for the algorithm (5.13) is given by

$$(I + \gamma G_e) e_{k+1} = e_k \quad (5.14)$$

**Proposition 5.6** *Assume that  $G_e + G_e^T$  is a positive-definite matrix. Then the algorithm (5.13) results in geometric convergence*

$$\|e_{k+1}\| \leq \frac{1}{1+\sigma} \|e_k\| \quad (5.15)$$

where  $\sigma$  is the smallest singular value of  $\frac{G_e + G_e^T}{2}$ .

**Proof.** The inner-product of (5.14) with  $e_{k+1}$  gives

$$\begin{aligned} e_{k+1}^T e_{k+1} + \gamma_{k+1} e_{k+1}^T G_e e_{k+1} &= e_{k+1}^T e_{k+1} + \gamma_{k+1} e_{k+1}^T \left( \frac{G_e + G_e^T}{2} \right) e_{k+1} \\ &= e_{k+1}^T e_k \end{aligned} \quad (5.16)$$

Using  $\frac{G_e + G_e^T}{2} \geq \sigma$  (where  $\sigma$  can be selected to be the smallest singular value of  $\frac{G_e + G_e^T}{2}$ ) and the Cauchy-Schwarz inequality (i.e.  $|x^T y| \leq \|x\| \|y\|$  for an arbitrary  $x$  and  $y$ ) in (5.16) one obtains the estimate

$$\|e_{k+1}\| \leq \frac{1}{1+\gamma\sigma} \|e_k\| \quad (5.17)$$

and hence the algorithm results in geometric convergence for a positive  $\gamma$ .  $\square$

If the  $G_e + G_e^T$  is not a positive-definite matrix, the following proposition characterises the convergence behaviour:

**Proposition 5.7** *The algorithm (5.13) will converge asymptotically to zero tracking error for any  $\gamma > 0$ .*

**Proof.** The eigenvalues of  $(I + \gamma G_e)^{-1}$  are  $(1 + \gamma C\Gamma)^{-1}$ , and because it is assumed that  $C\Gamma > 0$ , then the tracking error from algorithm (5.13) converges asymptotically to zero for  $\gamma > 0$ , because eigenvalues of the learning operator  $(I + \gamma G_e)^{-1}$  satisfy  $|1/(1 + \gamma C\Gamma)| < 1$ .  $\square$

Note that in Proposition 5.6, convergence is achieved in one iteration, and in  $N$  iterations in Proposition 5.7, when  $\gamma \rightarrow \infty$ . However, in practice this is not feasible due to modelling uncertainties. In order to find a balance between robustness and convergence speed, it is suggested that between trials the following optimisation problem is to be solved in order to find a suitable  $\gamma$ :

$$J_{k+1} = (\gamma_{k+1}) = \|e_{k+1}\|^2 + w\gamma_{k+1}^2 \quad (5.18)$$

and the optimal  $\gamma_{k+1}$  is implemented with the algorithm

$$u_{k+1}(t) = M_{k+1}^{-1} \{u_k(t) + \gamma_{k+1} r(t+1) - \gamma_{k+1} C\Gamma x_{k+1}(t)\} \quad (5.19)$$

where  $M_{k+1} = (1 + \gamma_{k+1} C\Gamma)^{-1}$ . It is straightforward to show that the optimal  $\gamma_{k+1}$  is given by (using the same mechanism as in Proposition 5.1)

$$\gamma_{k+1} = \frac{e_{k+1}^T G_e e_{k+1}}{w + e_{k+1}^T G_e^T G_e e_{k+1}} \quad (5.20)$$

Equation (5.20), however, is implicit in the sense that  $e_{k+1}$  is a function of  $\gamma_{k+1}$  and hence it can be only used for analysis. One way to solve the optimal  $\gamma_{k+1}$  'in practice' is to use a line search method where the value of the cost function (5.18) is calculated for given  $\gamma$  using a simulation model of the plant in question. The convergence behaviour of the algorithm is analysed in the following

**Proposition 5.8** *Consider the algorithm (5.19) with the update law (5.20) for  $\gamma_{k+1}$ . If the plant  $G_e$  is positive, then  $\lim_{k \rightarrow \infty} e_k = 0$ ,  $\lim_{k \rightarrow \infty} \gamma_k = 0$  and the error evolution satisfies the following inequality*

$$\|e_{k+1}\| \leq \frac{1}{1 + \gamma_{k+1}\sigma} \|e_k\| \quad (5.21)$$

which implies geometric convergence to zero tracking error.

**Proof.** By optimality and the non-optimal choice  $\gamma_{k+1} = 0$ , (5.18) results in the inequality  $\|e_{k+1}\|^2 \leq \|e_{k+1}\|^2 + w\gamma_{k+1}^2 \leq \|e_k\|^2$ , showing monotonic convergence. Furthermore, a similar inductive argument as in Proposition 5.3 shows that  $\lim_{k \rightarrow \infty} \gamma_{k+1} = 0$ . Because it is assumed that  $G_e + G_e^T \geq \sigma I$ , the update law (5.20) shows that  $\lim_{k \rightarrow \infty} e_k = 0$ . Furthermore, (5.15) implies (5.21) where  $\gamma$  is replaced with  $\gamma_{k+1}$ .  $\square$

In summary the optimisation based algorithm results in *geometric* convergence under the positivity assumptions, which is a very desirable property of an ILC algorithm. Furthermore, because  $\lim_{k \rightarrow \infty} \gamma_k = 0$ , it is clear from (5.21) that the convergence becomes slower as the number of iterations increases, resulting in a 'cautious algorithm'.

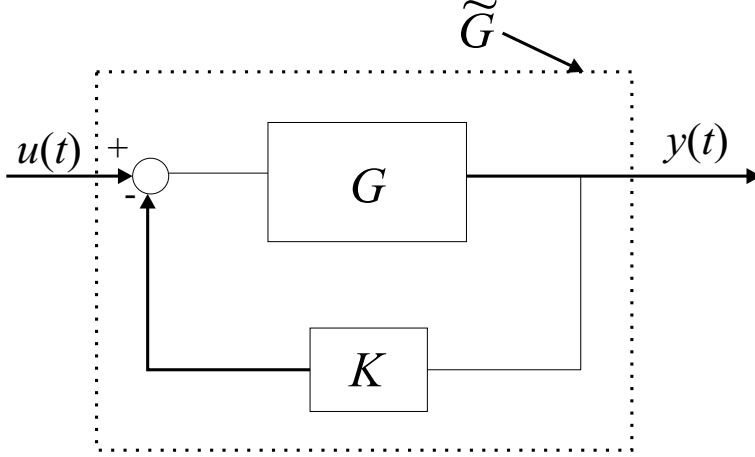
**Remark 5.3** *Note that a similar analysis that is given in Proposition 5.5 for the feedforward algorithm holds trivially for the predictive algorithm, i.e. if the matrix  $G_e + G_e^T$  is not positive-definite, it is possible for the algorithm to converge to a non-zero tracking error vector. Also Remark 5.2 holds in the predictive feedback case.*

**Remark 5.4** *One way to avoid the convergence to a non-zero tracking error is use to adaptive feedback algorithm until  $\gamma_{k+1}$  is 'reasonably small'. After that  $\gamma_{k+1}$  is frozen, and due to Proposition 5.7 the algorithm will converge from the initial condition  $e_{k+1}$  asymptotically to zero tracking error.*

### 5.3 Conditioning and positivity

In this section it will be shown that even if the original plant is not positive (i.e.  $G_e + G_e^T$  is not a positive-definite matrix), a suitable feedback law can be used to transform the plant to be positive. The method presented in this section is adopted from Owens *et al.* (2001), where a new conditioning method was developed in order to obtain positivity for continuous-time repetitive systems. The method is based on an additional feedback loop around the system given by  $u(t) = -Ky(t)$ , where

$K \in \mathbb{R}$ , which conditions the system so that it becomes positive. This approach is illustrated in Fig. 5.1, where the feedback systems forms a new ‘conditioned’ plant model  $\tilde{G}$ .



**Figure 5.1. Feedback conditioning.**

In order to extend this method to discrete-time ILC-systems, recall the following definition from Desoer & Vidyasagar (1975):

**Definition 5.1** A time-invariant system with an impulse response  $g(t)$ ,  $g \in l_1$  is called *positive-real*, if

$$\sum_{t=0}^{\infty} u(t)^T \sum_{i=0}^t g(t-i)u(i) \geq \gamma \|u\|_2^2 \quad (5.22)$$

where  $u \in l_2$  is arbitrary and  $\gamma \geq 0$ . If  $\gamma > 0$  then the system is called *strictly positive real*.

Note that if a system defined over infinite time-axis  $t \in [0, \infty)$  is positive real, then its restriction to a finite time-axis is positive real, because the set  $S = \{u \in l_2 | u(t) = 0, t > T\}$  is a subset of  $l_2$ . It is also straightforward to show that if the time-axis is finite, then condition (5.22) is equivalent to

$$u^T G_e u \geq \gamma \|u\|_2^2 \quad (5.23)$$

where  $G_e$  is given by

$$G_e = \begin{bmatrix} g(0) & 0 & 0 & \dots & 0 \\ g(1) & g(0) & 0 & \dots & 0 \\ g(2) & g(1) & g(0) & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g(T-1) & g(T-2) & \dots & g(1) & g(0) \end{bmatrix} \quad (5.24)$$

Note that the positivity of the lifted system has to be tested, i.e.  $h(0) = C\Gamma$ ,  $h(1) = C\Phi\Gamma$  etc. It is rather straightforward to show that an equivalent condition for strict positivity is that the  $Z$ -transform  $G(z)$  of the impulse response  $g(t)$  satisfies

- 1)  $G(z)$  is analytic for  $|z| > 1$ ,
- 2)  $\operatorname{Re}[G(e^{j\theta T_s})] > 0$ ,

where  $T_s$  is the sampling time and  $\theta \in [0, 2\pi/T_s]$ . Note that the condition  $\operatorname{Re}[G(e^{j\theta T_s})] > 0$  is equivalent to  $G(e^{j\theta T_s})^* + G(e^{j\theta T_s}) > 0$ . From now on, in order to shorten notation, it is assumed that  $T_s = 1$ . Note also that, because the positivity of the lifted plant  $G_e$  is analysed, this corresponds in the infinite time-axis case to the positivity of the transfer-function  $zG(z)$ .

**Proposition 5.9** *Assume that  $zG(z)$  has relative degree zero and is minimum phase. Then there exists a feedback gain  $K$  so that the closed-loop system  $\tilde{G}(z) := zG(z)(KzG(z) + 1)^{-1}$  satisfies conditions 1) and 2).*

**Proof.** The closed-loop feedback system (i.e. with the additional feedback loop  $u(t) = -Ky(t)$ ) can be written as

$$\begin{aligned} y(z) &= (1 + KzG(z))^{-1}(zG(z))u(z) \\ &= ((zG(z))^{-1} + K)^{-1}u(z) \end{aligned} \quad (5.25)$$

and the goal is to select  $K$  so that the transfer function  $((zG(z))^{-1} + K)^{-1}$  is strictly positive. As starting point it can be shown that the strict positivity of  $((zG(z))^{-1} + K)$  implies the strict positivity of  $((zG(z))^{-1} + K)^{-1}$ . Furthermore, the inverse of  $zG(z)$  can be written as

$$L(z) := (zG(z))^{-1} = C\Gamma^{-1} - z^{-1}(C\Gamma)^{-2}C\Phi\Gamma + H(z) \quad (5.26)$$

where  $H(z)$  is strictly proper and stable because  $zG(z)$  was assumed to have relative degree zero and to be minimum phase. This gives

$$\begin{aligned} \tilde{L} &:= L(e^{j\theta})^* + L(e^{j\theta}) \\ &= 2C\Gamma^{-1} - e^{-j\theta}((C\Gamma)^{-2}C\Phi\Gamma - ((C\Gamma)^{-2}C\Phi\Gamma)^T) + H(e^{j\theta}) + H(e^{j\theta})^* \end{aligned} \quad (5.27)$$

However, because only SISO-systems are considered in this chapter,  $(C\Gamma)^{-2}C\Phi\Gamma - ((C\Gamma)^{-2}C\Phi\Gamma)^T = 0$  and hence

$$\tilde{L} + 2F_0 = 0 \quad (5.28)$$

where  $F_0(\theta) = -(C\Gamma^{-1} + 1/2(H(e^{j\theta}) + H(e^{j\theta})^*))$ . Because  $H(z)$  is strictly proper and stable,  $F_0(\theta)$  is bounded for each  $\theta$  and thus  $K$  can be chosen so that  $K > F_0(\theta)$  for all  $\theta$ . With this choice  $\tilde{L} + 2K > 0$ , showing that  $\tilde{G}(z)$  will be strictly positive and the condition 2) is met. Furthermore, because it was assumed that  $zG(z)$  has a relative degree zero and is minimum phase, there always exists  $K$ ,  $K > F_0$  such that the closed-loop system is stable, i.e condition 1) is also satisfied. This is due the fact that the closed-loop poles approach the open-loop zeros as  $K$  is increased (see Ogata (1973)). Because  $zG(z)$  is minimum phase (the zeros of  $zG(z)$  are the zeros of  $G(z)$  and  $z = 0$ ) and has relative degree zero, each open-loop zero of  $G(z)$

is inside the unit circle. □

As was mentioned earlier a plant that is positive over the infinite time-axis  $[0, \infty)$  is also going to be positive over the finite time-axis  $[0, T]$  in the sense of (5.23). Consequently Proposition 5.9 shows that if the feedback law is applied on the plant  $G_e$  with  $K$  sufficiently large, the closed-loop system  $\tilde{G}_e$  is positive in the sense of (5.23).

In order to implement the control law  $u(t) = -Ky(t)$  on the original plant, it is important to notice that the design was done for the plant model  $y_k = G_e u_k$  where  $u_k = [u_k(0) \ u_k(1) \ \dots \ u_k(N-1)]^T$ ,  $y_k = [y_k(1) \ y_k(2) \ \dots \ y_k(N)]^T$ . Hence the control law  $u_k = -Ky_k$  reads  $u_{k+1}(t) = -Ky_{k+1}(t+1)$  when it is implemented on the original 'real' plant. This implementation is of course noncausal, but by using the plant model it can be written as

$$u_{k+1}(t) = -(1 + KCT)^{-1} KC\Phi x(t) \quad (5.29)$$

The drawback of (5.29) is that the implementation of it requires full knowledge of the state of the system. However, if this information is not available, an optimal observer can be always designed to give a state estimate.

**Remark 5.5** *In Proposition 5.9 it is assumed that the zeros of the  $G(z)$  are inside the unit circle. In order to investigate when this property holds, consider an  $n$ 'th order continuous-time system  $G(s)$  with  $m$  zeros  $z_i, i = 1, 2, \dots, m$ . Let  $G_T(z)$  be the corresponding sampled system where the sampling is done by using zero-order hold and the sampling time is  $T$ . According to Astrom & Wittenmark (1984) the system  $G_T(z)$  will have the following properties as  $T \rightarrow 0$ :*

- 1) *The discrete-time system  $G_T(z)$  will have  $n-1$  zeros,*
- 2)  *$m$  zeros of  $G_T(z)$  will be given by  $e^{z_i T}$ ,*
- 3) *The remaining  $r = n - 1 - m$  zeros will be the zeros of a polynomial  $P_r(z)$ , which for  $r = 1$  has the zero  $z = -1$  and for  $r > 1$  has zeros outside the unit circle.*

*Consequently if the original continuous-time system has a relative degree 2 or more, the discrete-time system will have a zero outside the unit circle for high sampling frequencies, and therefore the conditioning technique cannot be used.*

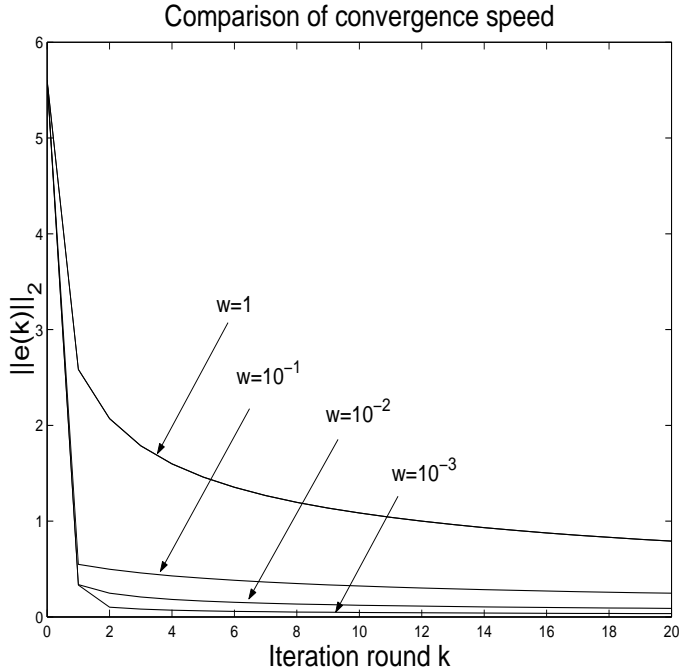
## 5.4 Simulation examples

Consider a continuous time system

$$(p^2 + 5p + 6)y(t) = (p + 1)u(t) \quad (5.30)$$

where  $p := \frac{d}{dt}$ . This plant is sampled with zero order hold using sampling time  $T_s = 0.1$  seconds and the length of the trial is 6 seconds. The reference signal is  $r(t) = \sin(t)$ . It is easy to check numerically that the resulting matrix  $G_e + G_e^T$  is

positive-definite. Fig. 5.2 shows the convergence behaviour with the parameter-optimal feedforward algorithm for different values of  $w$ , and as theory suggests, the convergence is geometric. Fig. 5.3 on the other hand shows the convergence behaviour with feedback algorithm (5.19) for different values of  $w$ , and the convergence is, as theory suggests, geometric.



**Figure 5.2.**  $\|\tilde{e}(k)\|$  as a function of iteration round  $k$  with the feedforward algorithm.

As another example consider the following discrete-time model

$$(q^2 - 1.5595q + 0.6065)y(t) = (0.0163q - 0.0006)u(t) \quad (5.31)$$

where  $q$  is the standard forward shift operator. The sampling time, length of the trial, and reference signal are the same as in the first example. The Nyquist-diagram of  $(zG(z))^{-1}$  (note that  $zG(z)$  is the plant whose positivity has to be analysed) is shown in Fig. 5.4. From the figure it can be seen that the minimum value of the real part of  $(zG(z))^{-1}$  is approximately  $-10$  and hence the control law  $u(t) = -15y(t+1)$  is used to build up a new modified system  $\tilde{G}(z)$ . The Nyquist diagram of this modified plant is shown in Fig. 5.5 and clearly the modified system is positive. Fig. 5.6 shows the norm of the tracking error as a function of

iteration round for both the original plant and the modified plant with feedforward algorithm  $u_{k+1}(t) = u_k(t) + \gamma_{k+1}e(t+1)$  where  $\gamma$  is taken to be the solution of the optimisation problem (5.7) with  $w = 0.1$ . This figure shows that the conditioning of the plant can make a substantial difference in the convergence behaviour. Note that by decreasing  $w$  the convergence speed of the positive plant can be increased so that it becomes fast enough for a given application.

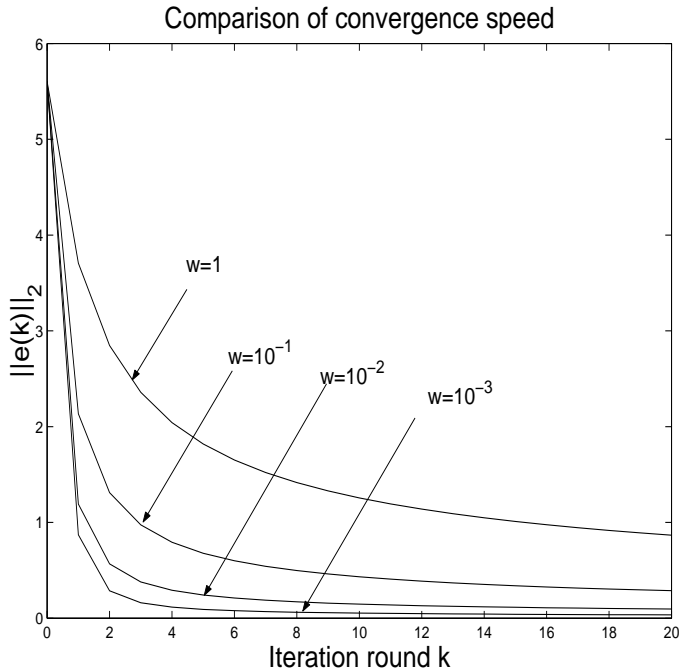
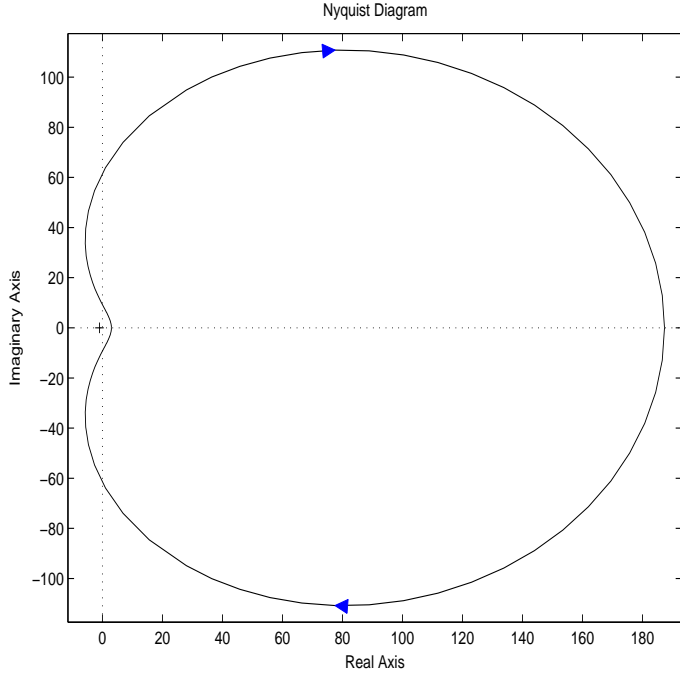


Figure 5.3.  $\|\bar{e}(k)\|$  as a function of iteration round  $k$  with the feedback algorithm.

## 5.5 A time-varying adaptive algorithm

In the previous section it was shown that a reasonably simple feedforward POILC algorithm converges monotonically to zero tracking error if the plant satisfies a suitable positivity condition. However, this positivity condition is restrictive, and therefore it is a valid question whether or not there exists structurally similar algorithms that do not result in any restrictions on the original plant. One possible



**Figure 5.4.** Nyquist-diagram of  $(zG(z))^{-1}$ .

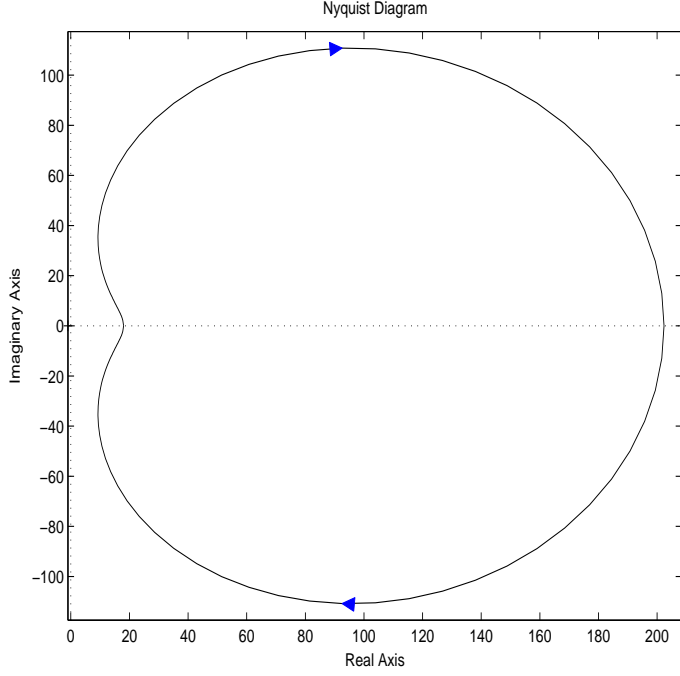
way of achieving this could be to add more flexibility into the algorithm by letting the learning gain be time-varying, which results in the algorithm

$$u_{k+1}(t) = u_k(t) + \gamma_{k+1}(t)e_k(t+1) \quad (5.32)$$

In addition to flexibility, another motivation for letting the learning gain to be time-varying is the observation from Chapter 3 that time-varying algorithms should be used instead of time-invariant ones in order to avoid ‘bad transients’ due to repeated poles along the iteration axis. In order to find suitable values for the learning gains  $\gamma_{k+1}(t)$ , consider the following optimisation problem

$$\begin{aligned} \min_{\vec{\gamma}_{k+1} \in \mathbb{R}^N} J_{k+1}(\vec{\gamma}_{k+1}) \\ J_{k+1}(\vec{\gamma}_{k+1}) = \|\vec{e}_{k+1}\|^2 + w \cdot \vec{\gamma}_{k+1}^T \vec{\gamma}_{k+1} \end{aligned} \quad (5.33)$$

where  $w \in \mathbb{R}$ ,  $w > 0$  and  $\vec{\gamma}_{k+1} := [\gamma_{k+1}(0) \ \gamma_{k+1}(1) \ \dots \ \gamma_{k+1}(N-1)]^T$ . The constraint equation for the optimisation problem is  $y_{k+1} = G_e u_{k+1}$ . Note that in this section the notation  $\vec{v}$  is used to emphasise that  $\vec{v}$  is a vector and not a real number. This is done in order to avoid any confusion between the learning gain  $\gamma \in \mathbb{R}$  from previous section and the vector of learning gains  $\vec{\gamma} \in \mathbb{R}^N$ . In order to



**Figure 5.5.** Nyquist-diagram of  $(\tilde{G}(z))^{-1}$ .

solve the optimisation problem (5.33), define

$$\begin{cases} \Delta x_{k+1}(t) := x_{k+1}(t) - x_k(t) \\ \Delta u_{k+1}(t) := u_{k+1}(t) - u_k(t) \end{cases} \quad (5.34)$$

By using the process model (5.2) the following description for the plant model is obtained:

$$\begin{cases} \Delta x_{k+1}(t+1) = \Phi \Delta x_{k+1}(t) + \Gamma \Delta u_{k+1}(t) \\ e_{k+1}(t) = e_k(t) - C \Delta x_{k+1}(t) \end{cases} \quad (5.35)$$

with  $\Delta x_{k+1}(0) = 0$ . Using the control law  $u_{k+1}(t) = u_k(t) + \gamma_{k+1}(t)e_k(t+1)$  (5.35) becomes

$$\begin{aligned} \Delta x_{k+1}(t+1) &= \Phi \Delta x_{k+1}(t) + \Gamma e_k(t+1) \gamma_{k+1}(t) \\ &= \Phi \Delta x_{k+1}(t) + \Gamma_{k+1}(t) \gamma_{k+1}(t) \end{aligned} \quad (5.36)$$

where  $\Gamma_{k+1}(t) := \Gamma e_k(t+1)$ . Because (5.35) is defined over finite time-interval  $t \in [0, 1, \dots, N]$  (5.35) has an equivalent (lifted) matrix representation in super-vector notation as

$$\vec{e}_{k+1} = \vec{e}_k - \tilde{G}_{e,k+1} \vec{\gamma}_{k+1} \quad (5.37)$$

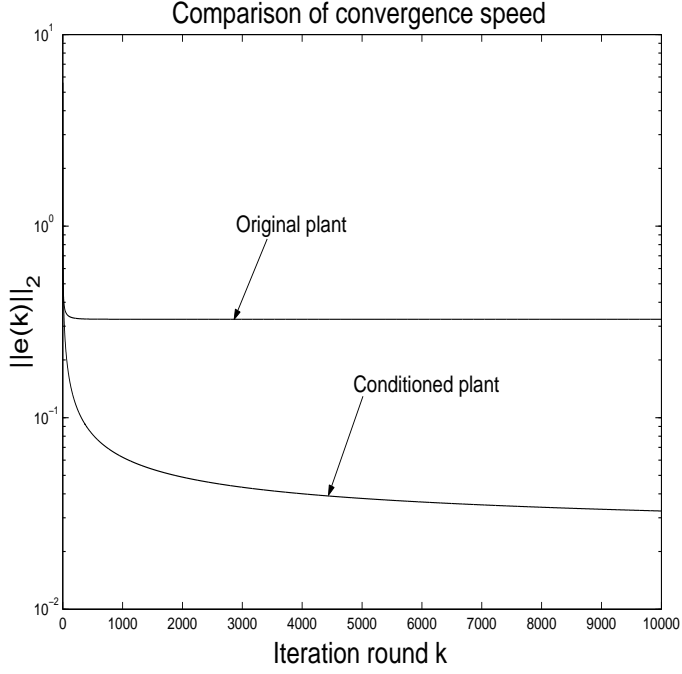


Figure 5.6. Comparison of convergence.

where

$$\tilde{G}_{e,k+1} = \begin{bmatrix} C\Gamma_{k+1}(0) & 0 & \dots & 0 \\ C\Phi\Gamma_{k+1}(0) & C\Gamma_{k+1}(1) & \dots & 0 \\ C\Phi^2\Gamma_{k+1}(0) & C\Phi\Gamma_{k+1}(1) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C\Phi^{N-1}\Gamma_{k+1}(0) & C\Phi^{N-2}\Gamma_{k+1}(1) & \dots & C\Gamma_{k+1}(N-1) \end{bmatrix} \quad (5.38)$$

Using (5.37) in (5.33), and setting the derivative with respect to  $\vec{\gamma}_{k+1}$  to zero, the optimising solution is given by

$$\vec{\gamma}_{k+1} = (wI + \tilde{G}_{e,k+1}^T \tilde{G}_{e,k+1})^{-1} \tilde{G}_{e,k+1}^T \vec{e}_k \quad (5.39)$$

where

$$\tilde{G}_{e,k+1}^T = \begin{bmatrix} \Gamma_{k+1}(0)^T C^T & \Gamma_{k+1}(0)^T \Phi^T C^T & \dots & \Gamma_{k+1}(0)^T (\Phi^{N-1})^T C^T \\ 0 & \Gamma_{k+1}(1)^T C^T & \dots & \Gamma_{k+1}(1)^T (\Phi^{N-2})^T C^T \\ 0 & 0 & \dots & \Gamma_{k+1}(2)^T (\Phi^{N-3})^T C^T \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \Gamma_{k+1}(N-1)^T C^T \end{bmatrix} \quad (5.40)$$

This equation can be used to calculate between the trials the optimal value of  $\vec{\gamma}_{k+1}$  but if the trial length is very large, this is not necessarily computationally efficient. Hence in Section 5.6 it is shown how the update law (5.39) can be implemented as a feedback law from  $\Delta x_{k+1}(t)$  plus a predictive term from  $e_k(t+1)$  and  $r(t+1)$ .

The convergence analysis of this algorithm is structurally very similar to the analysis of the feedforward POILC algorithm:

**Proposition 5.10** *The algorithm results in monotonic convergence, i.e.  $\|\vec{e}_{k+1}\| \leq \|\vec{e}_k\|$ , and  $\|\vec{e}_{k+1}\| = \|\vec{e}_k\|$ , if and only if  $\vec{\gamma}_{k+1}^* = 0$ . In addition, the sequence of optimal learning gains  $\{\vec{\gamma}_k^*\}$  satisfies  $\lim_{k \rightarrow \infty} \vec{\gamma}_k^* = 0$ .*

**Proof.** Let  $\vec{\gamma}_{k+1}^*$  be the optimal vector of learning gains for iteration  $k+1$  for an arbitrary  $k \in \mathbb{N}_+$ . The non-optimal choice  $\vec{\gamma}_{k+1} = 0$  results in the interlacing result

$$\|\vec{e}_{k+1}\| \leq J(\vec{\gamma}_{k+1}^*) = \|\vec{e}_{k+1}\|^2 + w\gamma_{k+1}^{*\text{T}}\gamma_{k+1} \leq J_{k+1}(0) \leq \|\vec{e}_k\|^2 \quad (5.41)$$

which implies that  $\|\vec{e}_{k+1}\| \leq \|\vec{e}_k\|$ . In addition, this result implies that  $\|\vec{e}_{k+1}\| = \|\vec{e}_k\|$  if and only if  $\vec{\gamma}_{k+1}^* = 0$ . The interlacing result also shows that

$$0 \leq \|\vec{e}_{k+1}\|^2 \leq \|\vec{e}_k\|^2 - w\|\vec{\gamma}_{k+1}^*\|^2 \quad (5.42)$$

Applying induction on this inequality gives

$$0 \leq \|\vec{e}_0\|^2 - w \sum_{i=1}^{k+1} \|\vec{\gamma}_i^*\|^2 \quad (5.43)$$

and because  $k$  is arbitrary,  $\lim_{k \rightarrow \infty} \vec{\gamma}_k^* = 0$ . □

In order to show that the algorithm converges to zero tracking error, note that if the equivalence ' $\vec{e}_k = 0 \iff \vec{\gamma}_{k+1} = 0$ ' would hold,  $\lim_{k \rightarrow \infty} \vec{e}_k = 0$  would hold based on Proposition 5.10 (which states that  $\lim_{k \rightarrow \infty} \vec{\gamma}_k^* = 0$ ). In order to show that this equivalence holds, note that (5.39) implies that the statement ' $\vec{e}_k = 0 \Rightarrow \vec{\gamma}_{k+1} = 0$ ' holds trivially. Therefore the convergence proof would be ready if it can be shown that ' $\vec{\gamma}_{k+1} = 0 \Rightarrow \vec{e}_k = 0$ '.

**Proposition 5.11**  *$\vec{e}_k \neq 0$  implies that  $\vec{\gamma}_{k+1} \neq 0$ .*

**Proof.** Assume that  $\vec{e}_k \neq 0$  but  $\vec{\gamma}_{k+1} = 0$  in (5.39). The matrix  $(wI + \tilde{G}_{e,k+1}^{\text{T}} \tilde{G}_{e,k+1})^{-1}$  in (5.39) is positive-definite, and therefore the only possibility for  $\vec{e}_k \neq 0$  but  $\vec{\gamma}_{k+1} = 0$  is that there exists a non-zero vector  $\vec{s}$  so that  $\vec{r} = \tilde{G}_{e,k+1}^{\text{T}} \vec{s} = 0$ . Because  $0 = r(N-1) = B_{k+1}(N-1)^{\text{T}} C^{\text{T}} s(N)$  and it is assumed that  $CB \neq 0$  (implying  $B^{\text{T}} C^{\text{T}} \neq 0$ ) and  $B_{k+1}(N-1) = Bs(N)$  by definition, this shows that  $s(N) = 0$ . Because  $s(N) = 0$ , this gives  $r(N-2) = B_{k+1}(N-1)^{\text{T}} C^{\text{T}} s(N-1)$ , and it can be shown using the same argument as before that  $s(N-1) = 0$ . Continuing inductively produces  $\vec{s} = 0$ , which is a contradiction. Therefore  $\vec{e}_k \neq 0$  implies that  $\vec{\gamma}_{k+1} \neq 0$ . □

Therefore  $\vec{e}_k \neq 0$  but  $\vec{\gamma}_{k+1} = 0$  is impossible, and the algorithm converges monotonically to zero tracking error. In order to have a more detailed description of the convergence behaviour, insert (5.39) into (5.37) which results in

$$\vec{e}_{k+1} = \vec{e}_k - \tilde{G}_{e,k+1}(wI + \tilde{G}_{e,k+1}^T \tilde{G}_{e,k+1})^{-1} \tilde{G}_{e,k+1}^T \vec{e}_k \quad (5.44)$$

A few lines of algebra shows that (5.44) can be written as

$$(I + w^{-1} \tilde{G}_{e,k+1} \tilde{G}_{e,k+1}^T) \vec{e}_{k+1} = \vec{e}_k \quad (5.45)$$

or equivalently

$$\vec{e}_{k+1} = (I + w^{-1} \tilde{G}_{e,k+1} \tilde{G}_{e,k+1}^T)^{-1} \vec{e}_k = L_{k+1} e_k \quad (5.46)$$

where the learning operator  $L_{k+1} := (I + w^{-1} \tilde{G}_{e,k+1} \tilde{G}_{e,k+1}^T)^{-1}$  maps the tracking error from trial  $k$  to trial  $k+1$ . Based on (5.46) the following proposition characterises in more detail the convergence behaviour of the algorithm:

**Proposition 5.12** *Assume that the eigenvalues of  $\tilde{G}_{e,k+1} \tilde{G}_{e,k+1}^T$  are non-zero. Then the sequence of tracking errors  $\{\vec{e}_k\}$  satisfies the following estimate:*

$$\|\vec{e}_{k+1}\| \leq \frac{1}{1 + w^{-1} \sigma_{\min,k+1}} \|\vec{e}_k\| \quad (5.47)$$

where  $\sigma_{\min,k+1}$  is the smallest eigenvalue of  $\tilde{G}_{e,k+1} \tilde{G}_{e,k+1}^T$ . If some of the eigenvalues of  $\tilde{G}_{e,k+1} \tilde{G}_{e,k+1}^T$  are zero, then the following estimate holds for the sequence  $\{\vec{e}_k\}$

$$\|P \vec{e}_{k+1}\| \leq \frac{1}{1 + w^{-1} \tilde{\sigma}_{\min,k+1}} \|P \vec{e}_k\| \quad (5.48)$$

where  $\tilde{\sigma}_{\min,k+1}$  is the smallest non-zero eigenvalue of  $\tilde{G}_{e,k+1} \tilde{G}_{e,k+1}^T$  and  $P : \mathbb{R}^N \rightarrow \mathbb{R}^N$  is a projection operator that projects an arbitrary vector  $v$  on the eigenvectors of  $\tilde{G}_{e,k+1} \tilde{G}_{e,k+1}^T$  that correspond to non-zero eigenvalues of  $\tilde{G}_{e,k+1} \tilde{G}_{e,k+1}^T$ .

**Proof.** Assume that the eigenvalues of  $\tilde{G}_{e,k+1} \tilde{G}_{e,k+1}^T$  are non-zero. Because  $\tilde{G}_{e,k+1} \tilde{G}_{e,k+1}^T$  is a symmetric positive-definite matrix, it can be shown that the quadratic form  $e_{k+1}^T (I + w^{-1} \tilde{G}_{e,k+1} \tilde{G}_{e,k+1}^T) e_{k+1}$  satisfies the inequality (see Gantmacher (1959))

$$e_{k+1}^T (I + w^{-1} \tilde{G}_{e,k+1} \tilde{G}_{e,k+1}^T) e_{k+1} \geq (1 + w^{-1} \tilde{\sigma}_{\min,k+1}) \|e_{k+1}\|^2 \quad (5.49)$$

Applying this result on the error evolution equation  $(I + w^{-1} \tilde{G}_{e,k+1} \tilde{G}_{e,k+1}^T) e_{k+1} = e_k$  results in

$$e_{k+1}^T e_k = e_{k+1}^T (I + w^{-1} \tilde{G}_{e,k+1} \tilde{G}_{e,k+1}^T) e_{k+1} \geq (1 + w^{-1} \tilde{\sigma}_{\min,k+1}) \|e_{k+1}\|^2 \quad (5.50)$$

and applying the Cauchy-Schwarz inequality (i.e.  $|x^T y| \leq \|x\| \|y\|$  for an arbitrary  $x, y \in \mathbb{R}^N$ ) on  $e_{k+1}^T e_k$  in (5.50) gives

$$(1 + w^{-1} \tilde{\sigma}_{\min,k+1}) \|e_{k+1}\|^2 \leq \|e_{k+1}\| \|e_k\| \quad (5.51)$$

Dividing (5.51) by  $\|e_{k+1}\|$  and  $(1 + w^{-1}\tilde{\sigma}_{\min,k+1})$  gives the estimate (5.47). The proof for the case when some of the eigenvalues of  $\tilde{G}_{e,k+1}\tilde{G}_{e,k+1}^T$  are zero can be derived in a similar manner and the proof is omitted due to its simplicity.  $\square$

Note also that the elements of  $G_{e,k+1}$  approach zero as  $\|\vec{e}_{k+1}\|^2$  approaches zero, which qualitatively implies that speed of convergence will decrease as the number of iterations are increased, resulting in a highly cautious algorithm.

## 5.6 Feedback implementation of the algorithm

In order to find a feedback implementation that does not require the calculation of an inverse of the ‘big’ matrix in (5.39), notice that (5.37) can be written as

$$\begin{aligned}\Delta\vec{x}_{k+1} &= G_{e,X,k+1}\vec{\gamma}_{k+1} \\ \vec{e}_{k+1} &= \vec{e}_k - C\Delta\vec{x}_{k+1}\end{aligned}\quad (5.52)$$

where

$$\tilde{G}_{e,X,k+1} = \begin{bmatrix} \Gamma_{k+1}(0) & 0 & \dots & 0 \\ \Phi\Gamma_{k+1}(0) & \Gamma_{k+1}(1) & \dots & 0 \\ \Phi^2\Gamma_{k+1}(0) & \Phi\Gamma_{k+1}(1) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \Phi^{N-1}\Gamma_{k+1}(0) & \Phi^{N-2}\Gamma_{k+1}(1) & \dots & \Gamma_{k+1}(N-1) \end{bmatrix}\quad (5.53)$$

Using this representation the cost function (5.33) can be written as

$$\begin{aligned}J_{k+1}(\Gamma_{k+1}) &= \{\vec{e}_k - C\Delta\vec{x}_{k+1}\}^T Q \{\vec{e}_k - C\Delta\vec{x}_{k+1}\} \\ &+ w \cdot \vec{\gamma}_{k+1}^T \vec{\gamma}_{k+1} + \vec{\lambda}_{k+1}^T \left( \tilde{G}_{e,X,k+1} \vec{\gamma}_{k+1} - \Delta\vec{x}_{k+1} \right)\end{aligned}\quad (5.54)$$

where the process model has been adjoined into the cost function and  $Q$  is a block diagonal semi-definite weighing matrix. Taking the partial derivatives with respect to  $\Delta\vec{x}_{k+1}$ ,  $\vec{\lambda}_{k+1}$  and  $\vec{\gamma}_{k+1}$  and setting them to zero results in the coupled equations

$$\begin{aligned}\vec{\lambda}_{k+1} &= -C^T Q \vec{e}_k + C^T Q C \Delta\vec{x}_{k+1} \\ \Delta\vec{x}_{k+1} &= \tilde{G}_{e,X,k+1} \vec{\gamma}_{k+1} \\ \vec{\gamma}_{k+1} &= -w^{-1} \tilde{G}_{e,X,k+1}^T \vec{\lambda}_{k+1}\end{aligned}\quad (5.55)$$

By inspection of  $\vec{\gamma}_{k+1} = -w^{-1} \tilde{G}_{e,X,k+1}^T \vec{\lambda}_{k+1}$  it is rather straightforward to see that an equivalent representation of this equation is given by

$$\gamma_{k+1}(t) = w^{-1} \Gamma_{k+1}^T p_{k+1}(t)\quad (5.56)$$

where  $p_{k+1}(t)$  is given by the following recursive equation

$$p_{k+1}(t) = \Phi^T p_{k+1}(t+1) - C^T Q e_k(t+1) + C^T Q C \Delta x_{k+1}(t+1)\quad (5.57)$$

with the boundary condition  $p_{k+1}(N) = 0$ . This is a non-causal implementation of the algorithm because at time point  $(k+1, t)$  the values of the state  $x_{k+1}(s)$  should be known for  $s > t$ . In order to find a causal solution it is postulated that  $p_{k+1}(t)$  can be written as

$$p_{k+1}(t) = -K_{k+1}(t)(I + \Gamma_{k+1}(t)^T w^{-1} \Gamma_{k+1}(t) K_{k+1}(t))^{-1} \Phi \Delta x_{k+1}(t) + \rho_{k+1}(t) \quad (5.58)$$

The idea now is to make (5.57) and (5.58) equal and collect all terms depending on  $\Delta x_{k+1}(t+1)$  on one side and the rest of the terms on the other side and set both sides to zero. This results in the following recursive equations for  $K_{k+1}(t)$  and  $\rho_{k+1}(t)$

$$\begin{aligned} K_{k+1}(t) &= \Phi^T M_{k+1}^{-1}(t+1) K_{k+1}(t+1) \Phi + C^T Q C \\ \rho_{k+1}(t) &= M_{k+1}^{-1}(t) \{ \Phi^T \rho_{k+1}(t+1) + C^T Q e_k(t+1) \} \end{aligned} \quad (5.59)$$

where  $M_{k+1}(t) = I + K_{k+1}(t) \Gamma_{k+1}(t) w^{-1} \Gamma_{k+1}^T(t)$  and the boundary conditions become  $K_{k+1}(N) = 0$  and  $\rho_{k+1}(N) = 0$ . These two equations are solved between trials  $k$  and  $k+1$  and the optimal value of  $\gamma_{k+1}(t)$  is given by

$$\gamma_{k+1}(t) = -w^{-1} \Gamma_{k+1}^T(t) M_{k+1}^{-1}(t) K_{k+1}(t) \Phi \Delta x_{k+1}(t) + w^{-1} \Gamma_{k+1}^T(t) \rho_{k+1}(t) \quad (5.60)$$

which is inserted into the control law  $u_{k+1}(t) = u_k(t) + \gamma_{k+1}(t) e_k(t+1)$  in ‘real-time’. Consequently the control law requires that the states of the system can be measured, but if this information is not available, an optimal state-observer can always be designed to estimate the states of the system. Furthermore, between trials both the predictive term and Riccati-feedback term have to be solved by using numerical simulation, which are computationally intensive tasks. However, when compared to the implementation (5.39), the implementation (5.60) avoids the need for calculating an inverse of a large matrix. The algorithm contains a state-feedback term, which possibly enhances the robustness properties of the algorithm when compared to the feedforward algorithm (5.39).

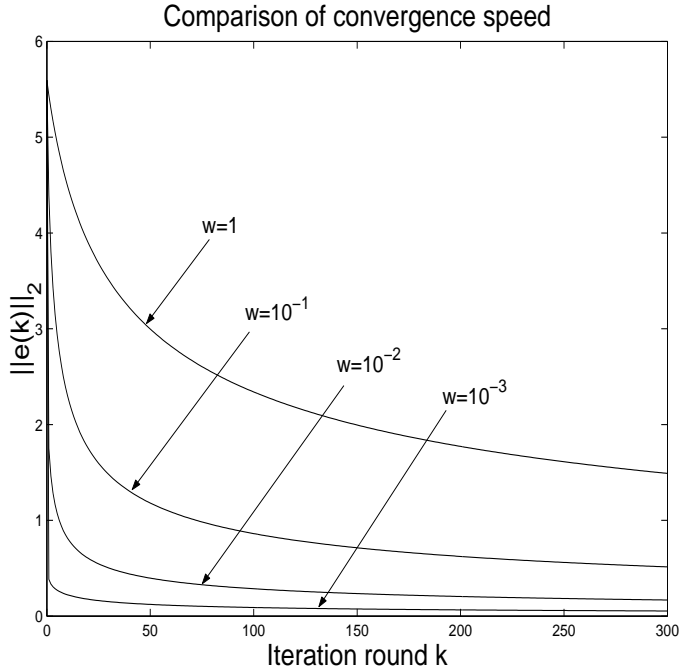
## 5.7 Simulation examples

As a first example we consider a process model

$$(p^2 + 5p + 6)y(t) = (p + 1)u(t) \quad (5.61)$$

where the plant is sampled with zero-order hold using sampling time  $T_s = 0.1$  seconds and the length of the trial is 6 seconds. The reference signal is  $r(t) = \sin(t)$ . Fig. 5.7 shows the convergence behaviour for different values of  $w$ . The graph shows that during earlier rounds the convergence speed is high, but the terminal convergence speed is slow, which is the expected result from the theoretical analysis.

In order to test that the idea that the algorithm is very cautious and hence possibly robust, assume that the true plant model is given by the transfer function



**Figure 5.7.**  $\|\tilde{e}(k)\|$  as a function of iteration round  $k$ .

$$(p + 1/2)(p + 2)(p + 10)y(t) = (p + 1)u(t) \quad (5.62)$$

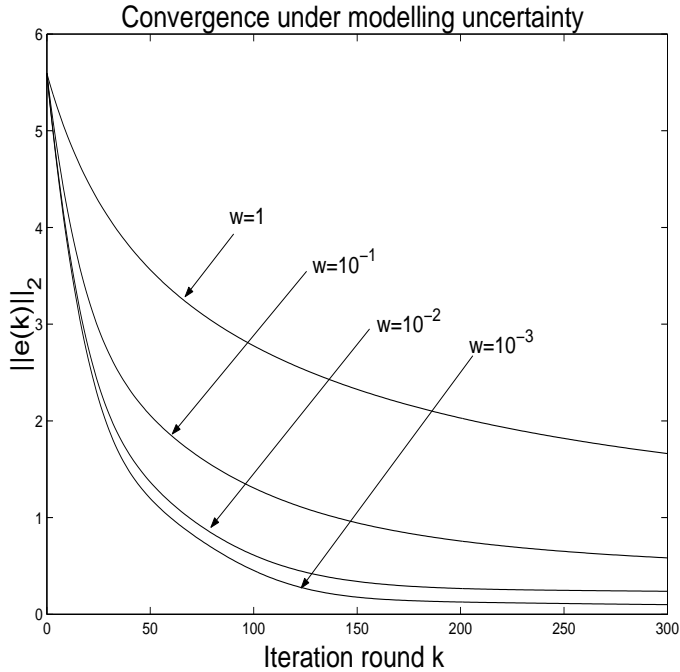
but for design the following nominal plant is used

$$py(t) = u(t) \quad (5.63)$$

which is a very rough approximation of the dynamics of the true plant. Fig. 5.8 shows the convergence behaviour for different values of  $w$ . This figure shows that even under the severe model uncertainty the algorithm results in monotonic convergence.

## 5.8 Summary

In this chapter a new paradigm termed Parameter Optimal Iterative Learning Control was proposed. The 'philosophy' behind POILC framework is to use a fixed algorithm structure where the algorithm contains an internal model of the reference signal. Furthermore, a suitable learning gain of the algorithm is made



**Figure 5.8.**  $\|\tilde{e}(k)\|_2$  as a function of iteration round  $k$  with model uncertainty.

iteration varying, and this learning gain is taken to be the solution of a suitable optimisation problem. As a first step the possibility of using a structurally very simple feedforward and predictive feedback algorithm was investigated. As a main result it was shown that the feedforward and predictive algorithm will converge monotonically to zero tracking error if the plant satisfies a positivity condition, otherwise convergence to a non-zero tracking error is possible. Furthermore, a predictive feedback controller was proposed for a class of LTI discrete-time systems. This controller conditions a non-positive plant so that the closed-loop system becomes positive, and after this conditioning step both the feedforward and feedback POILC algorithm can be used to achieve monotonic convergence to zero tracking error.

These results point out that by combining Predictive Control and ILC improved results can be achieved in terms of convergence behaviour. In fact in Townley (2002) Predictive Control and NOILC have been combined successfully, but more work on the connections between ILC and Predictive Control is needed.

In order to relax the assumption on positivity, a time-varying POILC algorithm was introduced. This modification was motivated by the algebraic analysis in Chapter 3, which implies that time-invariant algorithms can result in bad transient behaviour due to repeated poles along the iteration axis. Subsequent analysis

showed this time-varying algorithm results in monotonic convergence to zero tracking error for an arbitrary discrete-time LTI plant. This is a very strong property for any ILC algorithm. It was also shown that the time-varying algorithm has a highly nonlinear implementation where the learning gain is a function of the current state. Therefore a very general conclusion from the theoretical results presented in this chapter could be the following (rather bold) proposition:

In order to achieve high-quality performance, ILC based on LTI (possibly noncausal) filters should be abandoned all together. Instead, the focus of research should be shifted towards time-varying, adaptive and possibly nonlinear ILC algorithms, even if the original plant is a LTI system.

The drawback in the nonlinear implementation of the time-varying POILC algorithm is that it requires that the states of the system are measured or estimated. Furthermore, when compared to non-predictive NOILC, the implementation of the time-varying POILC algorithm is more complicated - in NOILC the Riccati equation for the state feedback term has to be solved only once, whereas in the time-varying POILC algorithm this equation has to be solved individually for each iteration. Therefore the search for simple algorithms with good convergence properties has to continue. In the following chapter it is investigated if simplicity and good convergence properties can be attained with a *high-order* time-invariant feed-forward POILC algorithm, where the input for iteration  $k + 1$  is calculated from

$$u_{k+1}(t) = u_k(t) + \delta_{k+1}(t) \quad (5.64)$$

$$\delta_{k+1}(t) := \sum_{i=1}^{M_1} \alpha_{k+1}(i) u_{k-1}(t) + \sum_{i=1}^{M_2} \beta_{k+1}(i) e_{k-i+1}(t+1)$$

In addition, the learning gains  $\{\alpha_{k+1}(i), \beta_{k+1}(j)\}$  are selected to be a solution of a suitable optimisation problem. Intuition says that the added flexibility should give better convergence properties when compared to the first-order time-invariant POILC algorithm. However, as the next chapter shows, intuition cannot be always trusted!

## 6 Basis functions and HOILC

As was shown in the previous chapter, the feedforward POILC algorithm

$$u_{k+1}(t) = u_k(t) + \gamma_{k+1}e_k(t+1) \quad (6.1)$$

will result in monotonic convergence to zero tracking error if the matrix description  $G_e$  of the plant model

$$\begin{cases} x_{k+1}(t+1) = \Phi x_{k+1}(t) + \Gamma u_{k+1}(t), & x_{k+1}(0) = 0 \\ y_{k+1} = C x_{k+1}(t) \end{cases} \quad (6.2)$$

satisfies a positivity condition  $G_e + G_e^T > 0$ . Furthermore, a sufficient condition for positivity is that  $zG(z)$  is a positive-real system. This is, however, a rather restrictive condition, and it would be important to find ways of relaxing this condition. In Owens & Feng (2002) and Owens & Feng (2003) it was suggested that one way to achieve this would be to use tracking errors from the last  $M$  repetitions resulting in the algorithm

$$u_{k+1}(t) = u_k(t) + \sum_{i=1}^M \beta_{k+1}(i) e_{k-i+1}(t+1) \quad (6.3)$$

The vector of learning gains  $\beta_{k+1} := [\beta_{k+1}(1) \ \beta_{k+1}(2) \ \dots \ \beta_{k+1}(M)]^T$  is selected to be the solution of the optimisation problem

$$\begin{aligned} \min_{\beta_{k+1} \in \mathbb{R}^N} J(\beta_{k+1}) \\ J(\beta_{k+1}) = \|e_{k+1}\|^2 + \beta_{k+1}^T W \beta_{k+1} \end{aligned} \quad (6.4)$$

where  $W$  is a diagonal positive-definite weighting matrix. This algorithm has added flexibility when compared to the first-order POILC algorithm because it has  $M$  decision variables instead of one. The optimisation problem (6.4) has been solved in Owens & Feng (2002) and Owens & Feng (2003), and the subsequent convergence analysis shows that the resulting algorithm has the following main properties:

- 1) The algorithm results in monotonic convergence, i.e.  $\|e_{k+1}\| \leq \|e_k\|$ .

- 2) If  $G_e + G_e^T$  is a positive-definite matrix then  $\lim_{k \rightarrow \infty} e_k = 0$
- 3) If  $G_e + G_e^T$  is semi-definite or non-definite, it is possible for the algorithm to converge to a non-zero tracking error.

Therefore in terms of learning (whether or not an algorithm converges to zero tracking error) the high-order algorithm behaves exactly as the first-order algorithm. In order to further explore the idea (giving up is not an option here) that high-order ILC might result in better convergence properties than the first-order algorithm, also previous inputs and a fixed set of basis functions are added into the algorithm (6.3) in the next section. It turns out that the basis functions play a crucial role in determining whether or not the algorithm converges to zero tracking error for a non-positive plant.

## 6.1 Motivation and derivation of the optimal high-order algorithm

In this section the following high-order algorithm

$$u_{k+1}(t) = u_k(t) + \delta_{k+1}(t) \quad (6.5)$$

$$\delta_{k+1}(t) := \sum_{i=1}^{M_1} \alpha_{k+1}(i) u_{k-1}(t) + \sum_{i=1}^{M_2} \beta_{k+1}(i) e_{k-i+1}(t+1) + \sum_{i=1}^{M_3} \gamma_{k+1}(i) f_i(t)$$

is considered. This algorithm is different from algorithm (6.3) because during iteration  $k+1$  the algorithm also uses information from previous input vectors  $u_j$  for  $j = k-1, k-2, \dots, k-M_1$  and more flexibility is allowed by adding iteration independent ‘basis functions’  $f_i$  into the input function. In this section a set of vectors  $\{f_i\}$  is called basis functions if an arbitrary vector  $v \in \mathbb{R}^N$  can be written as

$$v = \sum_{i=1}^N \alpha_i f_i \quad (6.6)$$

for some  $\alpha_i \in \mathbb{R}$  and  $\{f_i\}$  satisfies

$$f_i^T f_j = 0 \quad (6.7)$$

when  $i \neq j$ . The motivation for the use of basis functions is that without them the input  $u_{k+1}$  must lie in the span of  $[u_k, e_k, e_{k-1}, \dots, e_{k-M}]$ . If  $u_k$  converges (as is desired) and  $e_k \rightarrow 0$  (which is the objective), the vectors generating the span become highly collinear or infinitesimally small. The possibility of numerical ill-conditioning in the calculation of the coefficients  $\alpha_i, \beta_i$  is therefore potentially very high. The addition of the basis functions  $\{f_i\}$  extends the dimension of the search space and consequently is expected to result in faster learning convergence and increased accuracy. In the later parts of this chapter it is proved that a careful

choice of the basis functions leads to convergence to zero tracking error even for non-positive systems.

Define now the following parameter vectors

$$\alpha_{k+1} := \begin{bmatrix} \alpha_{k+1}(1) \\ \alpha_{k+1}(2) \\ \vdots \\ \alpha_{k+1}(M_1) \end{bmatrix}, \beta_{k+1} := \begin{bmatrix} \beta_{k+1}(1) \\ \beta_{k+1}(2) \\ \vdots \\ \beta_{k+1}(M_2) \end{bmatrix}, \gamma_{k+1} := \begin{bmatrix} \gamma_{k+1}(1) \\ \gamma_{k+1}(2) \\ \vdots \\ \gamma_{k+1}(M_3) \end{bmatrix} \quad (6.8)$$

With this notation the algorithm (6.5) can be written in more compact form as

$$u_{k+1} = u_k + U_k \alpha_{k+1} + E_k \beta_{k+1} + F \gamma_{k+1} \quad (6.9)$$

where

$$\begin{aligned} U_k &:= [u_{k-1} \ u_{k-2} \ \dots \ u_{k-M_1}] \\ E_k &:= [e_k \ e_{k-1} \ \dots \ e_{k+1-M_2}] \\ F &:= [f_1 \ f_2 \ \dots \ f_{M_3}] \end{aligned} \quad (6.10)$$

In order to select the free parameter vectors  $\alpha_{k+1}$ ,  $\beta_{k+1}$  and  $\gamma_{k+1}$  in the algorithm (6.9) it is proposed that, during iteration  $k+1$ , the following optimisation problem is to be solved

$$\begin{aligned} [\alpha_{k+1}^T, \beta_{k+1}^T, \gamma_{k+1}^T]^T &= \operatorname{argmin}\{J_{k+1}(\alpha_{k+1}, \beta_{k+1}, \gamma_{k+1}); y_{k+1} = G u_{k+1}\} \\ J_{k+1}(\alpha_{k+1}, \beta_{k+1}, \gamma_{k+1}) &:= \|e_{k+1}\|^2 + \alpha_{k+1}^T W_1 \alpha_{k+1} + \beta_{k+1}^T W_2 \beta_{k+1} + \gamma_{k+1}^T W_3 \gamma_{k+1} \end{aligned} \quad (6.11)$$

where  $W_1, W_2$  and  $W_3$  are symmetric positive-definite matrices. The stationary point of the cost-function (which is also the unique minimising solution of the minimisation problem due to convexity) is characterised by the equations

$$\frac{\partial J_{k+1}}{\partial \alpha_{k+1}} = 0 \quad \frac{\partial J_{k+1}}{\partial \beta_{k+1}} = 0 \quad \frac{\partial J_{k+1}}{\partial \gamma_{k+1}} = 0 \quad (6.12)$$

resulting in the following set of equations

$$A_k [\alpha_{k+1}^T \ \beta_{k+1}^T \ \gamma_{k+1}^T]^T = B_k \quad (6.13)$$

where

$$A_k = \begin{bmatrix} U_k^T G_e^T G_e U_k + W_1 & U_k^T G_e^T G_e E_k & U_k^T G_e^T G_e F \\ E_k^T G_e^T G_e U_k & E_k^T G_e^T G_e E_k + W_2 & E_k^T G_e^T G_e F \\ F^T G_e^T G_e U_k & F^T G_e^T G_e E_k & F^T G_e^T G_e F + W_3 \end{bmatrix} \quad (6.14)$$

and

$$B_k = \begin{bmatrix} U_k^T G_e^T e_k \\ E_k^T G_e^T e_k \\ F^T G_e^T e_k \end{bmatrix} \quad (6.15)$$

The matrix  $A_k$  can be rewritten as

$$A_k = \begin{bmatrix} W_1 & 0 & 0 \\ 0 & W_2 & 0 \\ 0 & 0 & W_3 \end{bmatrix} + [(G_e U_k)^T \ (G_e E_k)^T \ (G_e F)^T]^T [G_e U_k \ G_e E_k \ G_e F] \quad (6.16)$$

Furthermore, because the  $W_i$ s were assumed to be positive-definite,  $A_k$  is also positive-definite, and hence its inverse exists and the optimal values for the parameter vector for iteration  $k + 1$  can be obtained from the equation

$$[\alpha_{k+1}^T \beta_{k+1}^T \gamma_{k+1}^T]^T = A_k^{-1} B_k \quad (6.17)$$

The next step is to analyse whether or not the addition of basis functions relaxes the assumption on the positivity of the plant model. This analysis is carried out in the following section, and the analysis shows that if the basis functions are selected in a particular way, the relaxation is possible.

## 6.2 Convergence analysis

In order to analyse the convergence properties of the algorithm (6.5) with the parameter vector defined in (6.11) the following preliminary result is needed:

### Proposition 6.1

- a) Let  $(\alpha_{k+1}^*, \beta_{k+1}^*, \gamma_{k+1}^*)$  be the optimal parameter vector in (6.11). The performance index in (6.11) satisfies the following interlacing property  $\|e_{k+1}\| \leq J_{k+1}(\alpha_{k+1}^*, \beta_{k+1}^*, \gamma_{k+1}^*) \leq \|e_k\|$  with equality holding, if and only if,  $\alpha_{k+1}^* = 0$ ,  $\beta_{k+1}^* = 0$ ,  $\gamma_{k+1}^* = 0$ .
- b) The optimal parameter sequence satisfies the condition

$$\lim_{n \rightarrow \infty} \left[ \sum_{i=0}^n \alpha_{i+1}^{*T} W_1 \alpha_{i+1}^* + \beta_{i+1}^{*T} W_2 \beta_{i+1}^* + \gamma_{i+1}^{*T} W_3 \gamma_{i+1}^* \right] < \infty \quad (6.18)$$

which implies that

$$\lim_{k \rightarrow \infty} \alpha_{k+1}^* = 0, \quad \lim_{k \rightarrow \infty} \beta_{k+1}^* = 0 \quad \text{and} \quad \lim_{k \rightarrow \infty} \gamma_{k+1}^* = 0 \quad (6.19)$$

**Proof.** The non-optimal choice  $\alpha_{k+1} = \beta_{k+1} = \gamma_{k+1} = 0$  and an optimality argument results in the interlacing result

$$\begin{aligned} J_{k+1}(0, 0, 0) &= \|e_k\|^2 \geq J_{k+1}(\alpha_{k+1}^*, \beta_{k+1}^*, \gamma_{k+1}^*) \\ &= \|e_{k+1}\|^2 + \alpha_{k+1}^{*T} W_1 \alpha_{k+1}^* + \beta_{k+1}^{*T} W_2 \beta_{k+1}^* + \gamma_{k+1}^{*T} W_3 \gamma_{k+1}^* \end{aligned} \quad (6.20)$$

and consequently

$$\|e_{k+1}\|^2 \leq J_{k+1}(\alpha_{k+1}^*, \beta_{k+1}^*, \gamma_{k+1}^*) \leq \|e_k\|^2 \quad (6.21)$$

Furthermore, equality holds, if and only if,  $\alpha_{k+1}^* = \beta_{k+1}^* = \gamma_{k+1}^* = 0$ , which proves a). For the second part of Proposition 6.1, note that (6.21) implies that

$$0 \leq \|e_{k+1}\|^2 \leq \|e_k\|^2 - \alpha_{k+1}^{*T} W_1 \alpha_{k+1}^* - \beta_{k+1}^{*T} W_2 \beta_{k+1}^* - \gamma_{k+1}^{*T} W_3 \gamma_{k+1}^* \quad (6.22)$$

Applying induction gives the estimate

$$0 \leq \|e_{k+1}\|^2 \leq \|e_0\|^2 - \left( \sum_{i=1}^{k+1} \alpha_i^{*\text{T}} W_1 \alpha_i^* + \beta_i^{*\text{T}} W_2 \beta_i^* + \gamma_i^{*\text{T}} W_3 \gamma_i^* \right) \quad (6.23)$$

which holds for an arbitrary large  $k$ , concluding the proof for b).  $\square$

Proposition 6.1 shows that the parameter-optimal high order ILC algorithm is a descent algorithm as the norm of the tracking error is monotonically non-increasing in  $k$ , and the sequences of learning gains  $\alpha_{k+1}^*$ ,  $\beta_{k+1}^*$  and  $\gamma_{k+1}^*$  belong to  $l_{2,W_i}$  respectively where the norm is defined as  $\|v\|_{2,W_i}^2 := \sum_{j=1}^{\infty} v_j^{\text{T}} W_i v_j$  for an arbitrary vector  $v \in l_{2,W_i}$ . In the next theorem it is shown that for a special class of dynamical systems the tracking error sequence  $e_k$  converges to zero.

**Proposition 6.2** *If  $G_e$  is positive in the sense that  $G_e^{\text{T}} + G_e > 0$ , then the high-order algorithm (6.5) results in convergent learning, i.e.*

$$\lim_{k \rightarrow \infty} \|e_k\| = 0 \quad (6.24)$$

(Note that a similar result holds if the each of the eigenvalues of  $G_e^{\text{T}} + G_e$  are negative and this negative system can also transformed to a positive system by multiplying the plant with a negative unity gain).

**Proof.** The assumption that  $G_e + G_e^{\text{T}} > 0$  implies that

- a)  $G_e$  is non-singular
- b) There exists  $\sigma > 0$  so that  $G_e + G_e^{\text{T}} > \sigma I$

Let  $r = G_e u^*$  which results in  $e_k = G_e(u^* - u_k)$ . Because  $\{e_k\}$  is bounded according to Proposition 6.1 and  $G_e$  was assumed to be non-singular, the sequence  $\{u_k\}$  is bounded. As a consequence the sequences  $A_k$  and  $\{B_k\}$  in (6.13) are bounded. Consequently (6.13) and Proposition 6.1 imply that  $\lim_{k \rightarrow \infty} B_k = 0$ , which is equivalent to

$$\begin{cases} \lim_{k \rightarrow \infty} U_k^{\text{T}} G^{\text{T}} e_k = 0 \\ \lim_{k \rightarrow \infty} E_k^{\text{T}} G^{\text{T}} e_k = 0 \\ \lim_{k \rightarrow \infty} F^{\text{T}} G^{\text{T}} e_k = 0 \end{cases} \quad (6.25)$$

and in particular  $\lim_{k \rightarrow \infty} e_k^{\text{T}} G_e e_k = 0$ . Using the positivity assumption b) in Proposition 6.2,  $\lim_{k \rightarrow \infty} e_k^{\text{T}} G_e e_k = 0$ , which implies that  $\lim_{k \rightarrow \infty} \|e_k\| = 0$ .  $\square$

In summary, the high-order update law plus the positivity condition on  $G_e$  implies that the tracking error sequence  $\{e_k\}$  converges in norm to zero, and the convergence is monotonic.

The result above was derived using only the second limit in (6.25). Intuitively, the three limits together should permit a relaxation of the positivity condition. Hence the next step is to analyse the effect of using previous input functions and basis functions on the convergence behaviour because these are the two new elements.

Suppose now that the original plant is not positive, and using a compactness argument, let  $e_\infty$  to be a cluster point of the algorithm and  $u_\infty$  the corresponding input, which are connected by the equation  $e_\infty = r - G_e u_\infty$ . Furthermore, continuity indicates that  $u_\infty$  and  $e_\infty$  satisfy the equations

- a)  $u_\infty^T G_e^T e_\infty = 0$  (from the first equation in (6.25))
- b)  $e_\infty^T G_e^T e_\infty = 0$  (from the second equation in (6.25))
- c)  $F^T G_e^T e_\infty = 0$  (from the third equation of (6.25))
- d)  $\|e_\infty\| \leq \|e_0\|$  (from Proposition 6.1)

Note that the condition  $u_\infty^T G_e^T e_\infty = 0$  is equivalent to

$$(r - e_\infty)^T e_\infty = 0 \quad (6.26)$$

The equations above define all possible cluster points of the algorithm. It is of interest to note that  $M_1 + M_2 + M_3$  equations define the optimal parameters but only  $2 + M_3$  equations define the cluster points, showing that the flexibility in  $\alpha_{k+1}$  and  $\beta_{k+1}$  has no effect on the potential limit set.

The first implication from these equations is that the set of cluster points is bounded based on d). The condition b) shows that the previously established result on positivity (see Owens & Feng (2002) and Owens & Feng (2003)), i.e. if the plant  $G_e$  is positive, the tracking error will converge, but if the plant is not positive, the norm of tracking error may converge to a non-zero value. Furthermore, (6.26) shows that the cluster points lie on the boundary of a high-dimensional ball (the zero vector being a member of this set). Finally, the equation  $F^T G_e^T e_\infty = 0$  shows that  $e_\infty$  is orthogonal to the vectors  $\{G_e f_i\}$  for  $i \in \{1, 2, \dots, M_3\}$ , further restricting the set of possible cluster points. In summary the cluster points have to lie in the intersection of the sets generated by a), b), c) and d), resulting in a rather complex geometrical characterisation of the cluster points. Note also that the limit set can be made arbitrarily small due to inequality d), i.e. if  $e_0 \rightarrow 0$ , then an arbitrary cluster point  $e_\infty \rightarrow 0$ .

Even though the geometrical characterisation of the cluster points is rather complex, there exists the following straightforward geometrical result on how the basis functions should be chosen so that the algorithm converges to zero tracking error in the limit:

**Proposition 6.3** *Suppose that  $G_e^T + G_e$  has both strictly positive and non-positive eigenvalues, and let  $S_1$  to be the span of eigenvectors of  $G_e^T + G_e$  that correspond to the non-positive eigenvalues of  $G_e^T + G_e$ . Assume that  $\{f_i\}$  is selected so that  $S_2 := \text{Span}\{G_e f_i\}_{j \in \{1, 2, \dots, M_3\}} \supset S_1$ . Then  $\lim_{k \rightarrow \infty} e_k = 0$  in norm.*

**Proof.** The relation  $(G_e F)^T e_\infty = 0$  shows that  $e_\infty$  is orthogonal to  $S_2$ . This implies that  $e_\infty$  belongs to the orthogonal complement of  $S_2$ . Because  $S_1 \subset S_2$ ,  $e_\infty$  belongs also to  $S_1^\perp$ , i.e.  $e_\infty$  belongs to the span of eigenvectors of  $G_e^T + G_e$  which correspond to the positive eigenvalues of  $G_e^T + G_e$ . The relation  $e_\infty^T (G_e + G_e^T) e_\infty = 0$  then implies that  $e_\infty = 0$ .  $\square$

In summary, this result shows that if the basis functions  $f_i$  are selected according to Proposition 6.3, the high-order algorithm will converge to zero, even if the original

plant is not positive. Basis functions hence provide clear benefits for convergence behaviour. In addition, as was explained in Section 6.1, the basis functions increase the size of the subspace where the input vector  $u_{k+1}$  can lie, and hence may increase the convergence speed even if the limit error is non-zero. Note also that if  $\{f_i\}$  form an orthogonal basis in  $\mathbb{R}^N$  where  $N$  is the number of sampling points in a trial, then by adding more basis functions the span can be made arbitrary large, and in the limit when all the basis functions are used, the span is whole  $\mathbb{R}^N$  and the convergence condition in Proposition 6.3 is automatically met.

However, if the condition above is not met, and there exists a non-zero  $e_\infty$  in the limit set, according to the next proposition the algorithm can converge to a non-zero solution:

**Proposition 6.4** *Assume that there exists a non-zero  $e_\infty$  in the limit set defined in (6.25). There then exists an initial condition  $u_0$  so that the algorithm converges to this non-zero  $e_\infty$ .*

**Proof.** Because  $G_e$  is assumed to be non-singular, there exists a unique  $u_\infty$  so that  $e_\infty = r - G_e u_\infty$ . Select the initial condition  $u_0$  to be equal to  $u_\infty$  and consequently  $e_0 = e_\infty$ . Furthermore, (6.14) shows that  $[\alpha_1^T, \beta_1^T, \gamma_1^T]^T = 0$ , and consequently  $u_1 = u_\infty$ . Using induction,  $u_k = u_\infty$  for  $k = 2, 3, \dots$  and the algorithm converges to the non-zero  $e_\infty$ .  $\square$

Numerical work in the next section will demonstrate the validity and importance of the theoretical results in this section.

## 6.3 Simulations

### 6.3.1 A positive system

As a first simulation example consider the continuous-time system

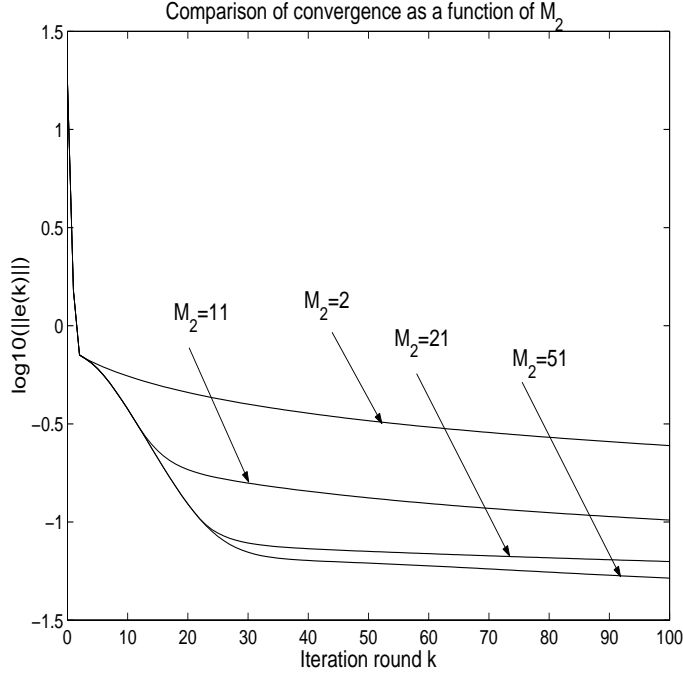
$$(p^2 + 5p + 6)y(t) = (p + 1)u(t) \quad (6.27)$$

where the trial length is 20 seconds, the sampling time  $T_s=0.1$  seconds, and the reference signal is  $r(t) = e^{t/20} \sin(t)$ . If this system is sampled with zero-order hold, it can be numerically checked that the corresponding system matrix  $G_e$  is positive-definite.

#### A positive system and the effect of previous tracking errors

In order to analyse the effect of previous tracking errors on the convergence behaviour, select  $M_1 = 0$  and  $M_3 = 0$ . Consider now the following four different cases:  $M_2 = 2$  and  $W_2 = I_{2 \times 2}$ ,  $M_2 = 11$  and  $W_2 = I_{11 \times 11}$ ,  $M_2 = 21$  and  $W_2 = I_{21 \times 21}$ , and finally,  $M_2 = 51$  and  $W_2 = I_{51 \times 51}$  where  $I_{n \times n}$  is the  $n \times n$ -dimensional identity matrix. Fig. 6.1 shows, on a logarithmic scale, that by increasing  $M_2$  (i.e. by increasing the number of previous tracking errors in the update

law) the convergence becomes faster and more accurate. In order to demonstrate how the logarithmic norm correlates with tracking accuracy, Fig. 6.2 shows the tracking error after 200 iterations.



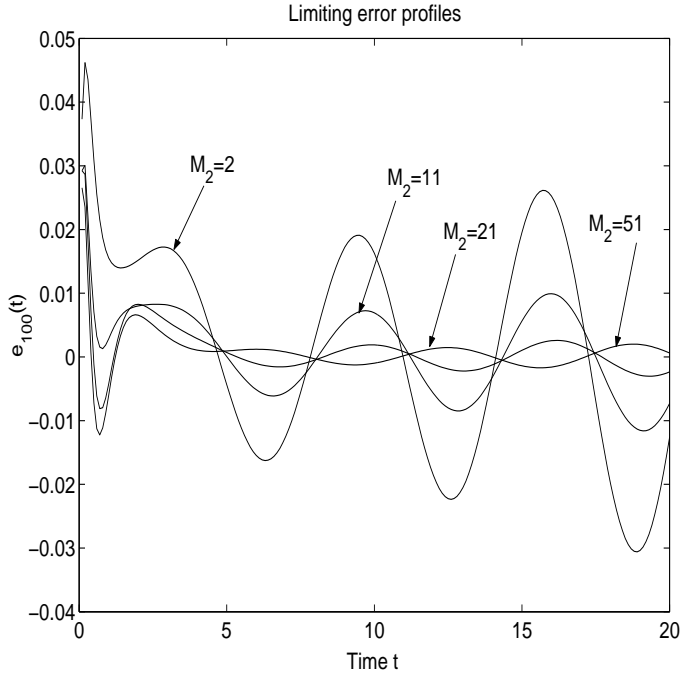
**Figure 6.1.** Convergence behaviour in terms of  $M_2$  for the positive system (6.27).

### A positive system and the effect of previous input functions

In order to test the effect of  $M_1$  on convergence behaviour take  $M_2 = 1$ ,  $W_2 = 1$ ,  $M_3 = 0$  and consider the following four cases:  $M_1 = 5$  and  $W_1 = I_{5 \times 5}$ ,  $M_1 = 11$  and  $W_1 = I_{11 \times 11}$ ,  $M_1 = 15$  and  $W_1 = I_{15 \times 15}$ , and finally,  $M_1 = 51$  and  $W_1 = I_{51 \times 51}$ . Fig. 6.3 shows the convergence behaviour which indicates that by increasing  $M_1$  faster initial convergence speed can be achieved but terminal convergence is not affected considerably by  $M_1$ .

### A positive system and the effect of weighting matrices

Finally, in order to test the effect of the weighting matrices on the convergence speed, take  $M_1 = M_2 = 2$  and  $M_3 = 0$ . Furthermore, let  $W_1 = W_2$  to be



**Figure 6.2.** Tracking error after 200 iterations in terms of  $M_2$  for the positive system (6.27).

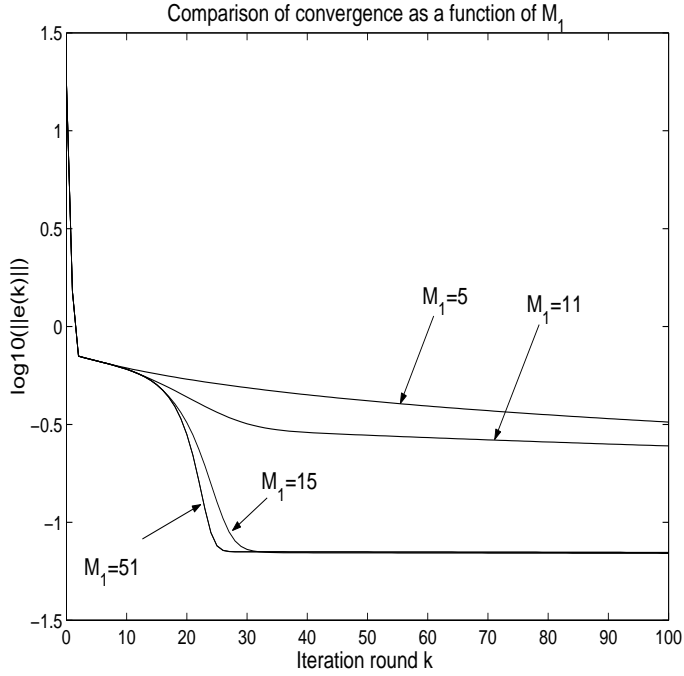
$W_1 = W_2 = 0.1I_{2 \times 2}$ ,  $W_1 = W_2 = I_{2 \times 2}$  and  $W_1 = W_2 = 10I_{2 \times 2}$ . From Fig. 6.4 it is clear that by decreasing  $W_1$  and  $W_2$  a faster convergence rate can be obtained. However, extremely small values of  $W_1$  or  $W_2$  can result in large values for the learning gains  $\alpha_{k+1}(i)$  and  $\beta_{k+1}(i)$ , possibly resulting in a non-robust algorithm.

### 6.3.2 A non-positive system

Consider now the linear system

$$(p+1)^2 y(t) = u(t) \quad (6.28)$$

sampled with zero-order hold and sampling time  $T_s = 0.1$  seconds. The length of the trial is 20 seconds and the reference signal is again  $r(t) = e^{t/20} \sin(t)$ . In this case the system matrix  $G_e$  is not positive-definite, so it is possible for the algorithm to converge to a non-zero tracking error if the basis functions are not selected properly.



**Figure 6.3.** Convergence behaviour in terms of  $M_1$  for the positive system (6.27).

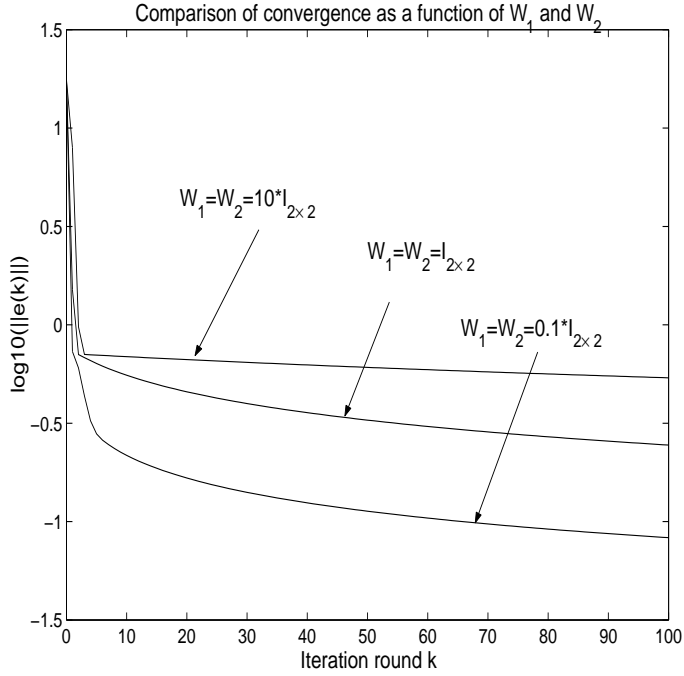
### A non-positive system and the effect of previous input functions

In order to analyse the effect of  $M_1$  on convergence speed without basis functions when the system is not positive take  $M_3 = 0$  and let  $M_1 = 2$  and  $W_2 = I_{2 \times 2}$ . Consider the four cases  $M_1 = 2$  and  $W_1 = I_{2 \times 2}$ ,  $M_1 = 11$  and  $W_1 = I_{11 \times 11}$ ,  $M_1 = 21$  and  $W_1 = I_{21 \times 21}$ , and  $M_1 = 51$  and  $W_1 = I_{51 \times 51}$ . The resulting convergence behaviour is shown in Fig. 6.5. From this figure it can be seen that by adding more previous inputs into the update law, the tracking accuracy increases but the algorithm converges to a non-zero tracking error.

### A non-positive system and Fourier basis

Add now the Fourier basis functions

$$f_n(t) = \sin(2n\pi t/T_l) \quad (6.29)$$



**Figure 6.4.** Convergence behaviour in terms of  $W_1$  and  $W_2$  for the positive system (6.27).

for  $n = 1, 2, \dots, (M_3 - 1)/2$  and

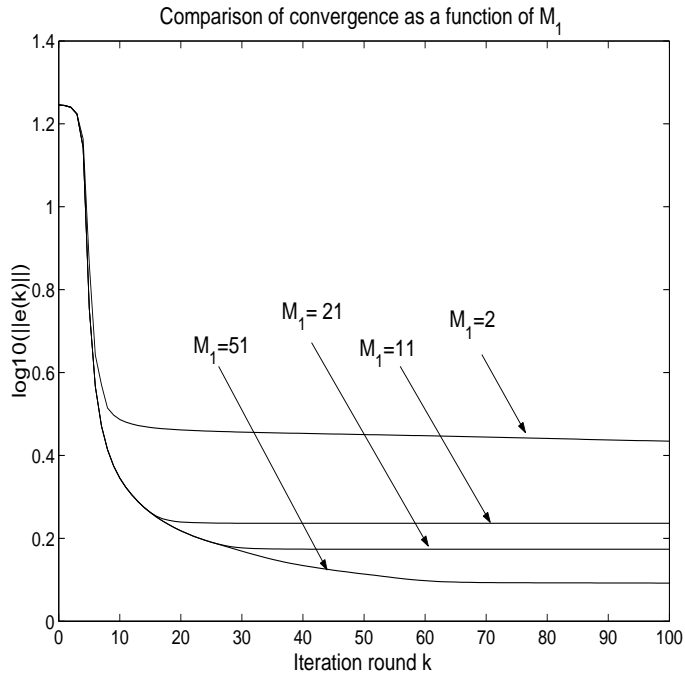
$$f_n(t) = \cos(2n\pi t/T_l) \quad (6.30)$$

$n = ((M_3 - 1)/2) + 1, \dots, M_3 - 1$  and  $f_{M_3} = 1$  into the basis function algorithm where  $M_3$  is assumed to be an odd number. Select  $M_1 = M_2 = 2$  and  $W_1 = W_2 = I_{2 \times 2}$ . Furthermore, let  $M_3 = 5$  and  $W_3 = I_{5 \times 5}$ ,  $M_3 = 11$  and  $W_3 = I_{11 \times 11}$ ,  $M_3 = 21$  and  $W_3 = I_{21 \times 21}$ , and finally,  $M_3 = 51$  and  $W_3 = I_{51 \times 51}$ . Fig. 6.6 shows the convergence behaviour for the Fourier basis. From this figure it is clear that by adding a sufficient number of basis functions, the convergence behaviour can be improved considerably.

### A non-positive system with Laguerre basis

As an alternative to the Fourier basis functions consider the Laguerre basis functions defined by the recursive equation

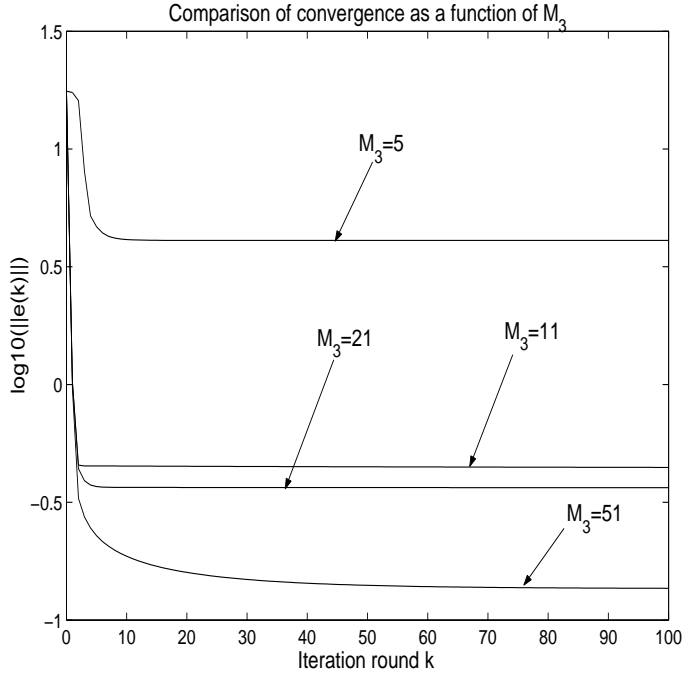
$$(i + 1)f_{n+1}(t) = (2n + 1 - t)f_n(t) - nf_{n-1}(t) \quad (6.31)$$



**Figure 6.5.** Convergence behaviour in terms of  $M_1$  for the non-positive system (6.28).

where the initial conditions for the recursion are  $f_1(t) = 1$  and  $f_t = -t+1$ . Fig. 6.7 shows the simulation results with exactly the same parameters as in the previous example, the only difference is that the Fourier basis functions are now replaced by the Laguerre basis functions.

Comparing Fig. 6.6 and Fig. 6.7 it is clear that at least with this particular simulation example the Laguerre basis results in faster convergence and more accurate tracking after 100 iterations than the Fourier basis functions. Note, that in both basis function cases, there is a sharp reduction in the norm of the tracking error during the first few iterations and after that the convergence speed slows down considerably. An intuitive explanation for this phenomenon is that the basis functions are linearly independent. Hence it is ‘easy’ for the algorithm to find the projection of the input function that gives perfect tracking on the span of the basis functions, resulting in the sharp reduction in the norm of the tracking error during the initial iterations.



**Figure 6.6.** Convergence behaviour in terms of  $M_3$  for the non-positive system (6.28) with Fourier basis.

## 6.4 Summary

This chapter investigated whether or not high-order POILC algorithms would result in better convergence properties when compared to the first-order feedforward POILC algorithm proposed in Section 5. Intuition would suggest that this should be the case, because for the first-order case

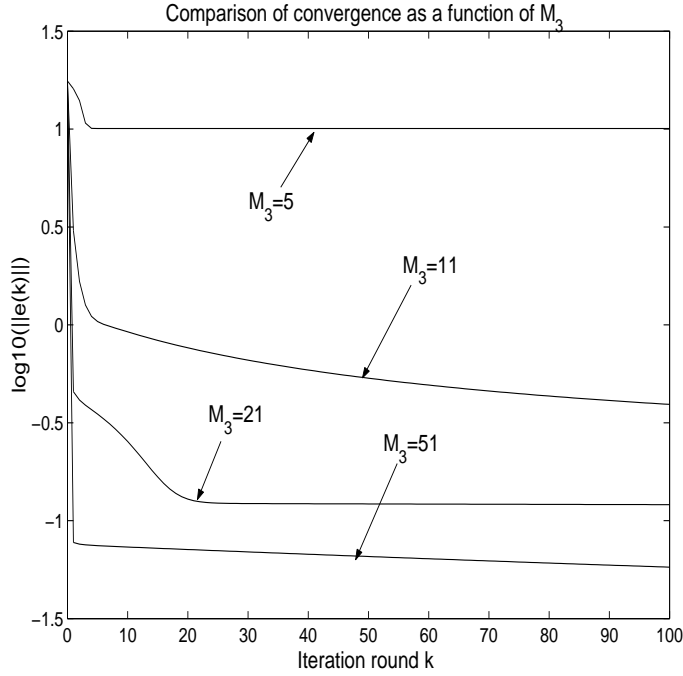
$$u_{k+1} \in \text{Span}\{u_k, e_k\} \quad (6.32)$$

whereas the in the high-order case

$$u_{k+1} \in \text{Span}\{u_k, e_k, e_{k-1}, \dots, e_{k-M}\} \quad (6.33)$$

and therefore the size of the subspace where  $u_{k+1}$  has to lie is ‘bigger’ with high-order POILC than with first-order POILC. Quite surprisingly, though, both the first-order and high-order algorithm have to meet a positivity condition on the plant model in order to guarantee zero tracking error in the limit.

The underlying reason for this phenomenon is still unclear. A possible explanation is that, as the tracking error approaches to zero, the previous input and



**Figure 6.7.** Convergence behaviour in terms of  $M_3$  for the non-positive system (6.28) with Laguerre basis.

tracking error vectors become highly collinear and the added flexibility in terms of span is lost. Therefore in this chapter it was suggested that by adding suitable basis functions  $\{f_i\}$  into the algorithm the problem with collinearity can be avoided. The analysis shows that, if the span of basis functions contains the subspace spanned by the eigenvectors of  $G_e + G_e^T$  that correspond to the non-positive eigenvalues of  $G_e + G_e^T$ , the algorithm will converge monotonically to zero tracking error. Whether or not this addition of basis functions has any value in terms of practical applications has not yet been fully evaluated.

If the collinearity conjecture turns out to be true, this also implies that the high-order algorithm does not necessarily result in faster terminal convergence rate, because the subspaces (6.32) and (6.33) are almost equal during as  $k \rightarrow \infty$ . This observation was also supported by simulation studies, where an increase in  $M_1$  or  $M_2$  gave faster convergence rate during earlier iteration rounds, but the terminal convergence rate was almost equal for different values of  $M_1$  or  $M_2$ .

In summary there are two main issues with the POILC approach: the high-order algorithm, without the basis functions, does not relax the positivity assumption on the original plant. Secondly, improvement in terms of convergence speed can be seen only during earlier iterations rounds. Furthermore, both issues are related

to the fact that, during terminal convergence, previous tracking errors and inputs do not generate a span that would be rich enough to increase convergence speed or relax the positivity assumption. Therefore in the next chapter the more complex steepest-descent ('adjoint') algorithm  $u_{k+1} = u_k + \gamma_{k+1} G_e^T e_k$  is being considered, and the learning gain  $\gamma_{k+1}$  is taken to be solution to a suitable optimisation problem. Note that in this algorithm  $u_{k+1} \in \text{Span}\{u_k, g_i\}$  where  $g_i$  are the columns of the matrix  $G_e^T$ . Furthermore, because  $G_e$  is assumed invertible, the span is whole  $\mathbb{R}^N$ , and the coefficients in the linear combination are defined by  $e_k$ . This seems to result in geometrically richer structure when compared to POILC case. Therefore it is not a surprise that subsequent theoretical analysis shows that this new adjoint algorithm has several desirable properties in terms of convergence behaviour and modelling uncertainty.

## 7 A new robust steepest-descent algorithm

In this chapter the concept of parameter optimisation in ILC is combined with a steepest descent algorithm. The steepest descent algorithm in the context of continuous-time ILC was first proposed in Furuta & Yamakita (1987). In this paper the authors analyse the convergence properties of the algorithm when the plant model contains multiplicative uncertainty. The analysis is done under the assumption that the trial length is infinite. Furthermore, because the steepest-descent algorithm contains a non-causal operator, the time-axis is taken to be  $t \in (-\infty, \infty)$ . This results in complicated analysis and gives only sufficient conditions for convergence under model uncertainty. In this chapter a similar analysis is carried out in the discrete-time case assuming finite trial length, resulting in a sufficient and necessary condition for monotonic convergence. Furthermore, a modified steepest-descent algorithm is proposed to provide a transparent mechanism for balancing the dual aims of convergence speed and robustness. Finally, an interesting connection is established between the steepest-descent algorithm and NOILC algorithm, which may be a useful starting point for analysing the robustness properties of the NOILC algorithm.

In the standard steepest-descent method the following performance index is considered

$$J(u_{k+1}) = \|e_{k+1}\|^2, \quad e_{k+1} = r - G_e u_{k+1} \quad (7.1)$$

which reflects the objective of having a small tracking error during each iteration. In this chapter the analysis is done in the discrete-time domain, and therefore the plant model  $G_e$  in the cost function (7.1) is generated by the SISO state-space description

$$\begin{cases} x_{k+1}(t+1) = \Phi x_{k+1}(t) + \Gamma u_{k+1}(t), & x_{k+1}(0) = 0 \\ y_{k+1}(t) = C x_{k+1}(t) \end{cases} \quad (7.2)$$

Assume now that the input  $u_k$  is modified so that during iteration  $k+1$  the input function  $u_{k+1} = u_k + \gamma_{k+1} \delta_{k+1}$  is used where  $\gamma_{k+1}$  is the step size and  $\delta_{k+1}$  is the vector that determines the direction of the update vector. The tracking error for iteration  $k+1$  becomes

$$\begin{aligned} J(u_{k+1}) &= J(u_k + \gamma_{k+1} \delta_{k+1}) = \|e_{k+1}\|^2 \\ &= \|e_k\|^2 - 2\gamma_{k+1} \delta_{k+1}^T G_e^T e_k + \gamma_{k+1}^2 \delta_{k+1}^T G_e^T G_e \delta_{k+1} \end{aligned} \quad (7.3)$$

which results in

$$\begin{aligned} & \|e_{k+1}\|^2 - \|e_k\|^2 = \\ & -2\gamma_{k+1}\delta_{k+1}^T G_e^T e_k + \gamma_{k+1}^2 \delta_{k+1}^T G_e^T G_e \delta_{k+1} \end{aligned} \quad (7.4)$$

In order to achieve monotonic convergence, the right-hand side in (7.4) has to be made negative. One potential candidate is  $\delta_{k+1} = G_e^T e_k$ , resulting in the control law

$$u_{k+1} = u_k + \gamma_{k+1} G_e^T e_k \quad (7.5)$$

and the difference  $\|e_{k+1}\|^2 - \|e_k\|^2$  becomes

$$\begin{aligned} & \|e_{k+1}\|^2 - \|e_k\|^2 = \\ & -2\gamma_{k+1}\|G_e^T e_k\|^2 + \gamma_{k+1}^2 \|G_e G_e^T e_k\|^2 \end{aligned} \quad (7.6)$$

Because the negative term  $-2\gamma_{k+1}\|G_e^T e_k\|^2$  is of  $O(\gamma)$  and the positive term  $\gamma_{k+1}^2 \|G_e G_e^T e_k\|^2$  is of  $O(\gamma^2)$  in (7.6), by using a sufficiently small positive  $\gamma_{k+1}$  the right-hand side of (7.6) can be always made negative (note that  $G_e$  is assumed to be invertible). In order to automate the selection process for  $\gamma_{k+1}$ , in Furuta & Yamakita (1987) it was suggested that an ‘optimal’  $\gamma_{k+1}^*$  should be used for each iteration where  $\gamma_{k+1}^*$  is obtained from the equation

$$\gamma_{k+1}^* = \arg \min_{\gamma_{k+1} \in \mathbb{R}} J(u_k + \gamma_{k+1} G_e^T e_k) \quad (7.7)$$

It is straightforward to show that the optimal  $\gamma_{k+1}^*$  is given by

$$\gamma_{k+1}^* = \frac{\|G_e^T e_k\|^2}{\|G_e G_e^T e_k\|^2} \quad (7.8)$$

which results in

$$\|e_{k+1}\|^2 - \|e_k\|^2 = -\frac{\|G_e^T e_k\|^4}{\|G_e G_e^T e_k\|^2} \quad (7.9)$$

where the right-hand side is obviously negative, implying monotonic convergence, i.e.  $\|e_{k+1}\| < \|e_k\|$  if  $e_k \neq 0$ . It can be also shown by using a similar argument as in Proposition 7.4 that the algorithm will converge monotonically to zero tracking error. Assume now that the true plant includes uncertainty which is modelled with the equation

$$G_e = G_o U \quad (7.10)$$

where  $G_o$  is the nominal model (i.e. an estimate of the true plant), and  $U$  reflects the multiplicative uncertainty (i.e. modelling errors). Furthermore,  $G_o$  is used instead of  $G_e$  in the update law, i.e.  $u_{k+1}$  is calculated from the equation  $u_{k+1} = u_k + \gamma_{k+1} G_o^T e_k$ . For  $\|e_{k+1}\|^2 - \|e_k\|^2$  this results in

$$\begin{aligned} & \|e_{k+1}\|^2 - \|e_k\|^2 = -2\gamma_{k+1} e_k^T G_o U^T G_o^T e_k \\ & + \gamma_{k+1}^2 e_k^T G_o G_o^T G_e G_o^T e_k \end{aligned} \quad (7.11)$$

This equation can be used to characterise the effect of uncertainty on the convergence behaviour as will be shown in the following

**Proposition 7.1** *Suppose that  $U + U^T$  is a positive-definite matrix. If  $\|e_k\| \neq 0$  there exists a  $\gamma_{k+1} > 0$  so that  $\|e_{k+1}\|^2 - \|e_k\|^2 < 0$ .*

**Proof.** Note that because  $U$  is assumed to be positive, the term  $-\gamma_{k+1}e_k G_o U^T G_o^T e_k$  in (7.11) is strictly negative for an arbitrary non-zero  $e_k$  and  $\gamma_{k+1} > 0$  and the second term  $\gamma_{k+1}^2 e_k^T G_o G_e^T G_e G_o^T e_k$  in (7.11) is strictly positive for an arbitrary non-zero  $e_k$ . Furthermore, the first term is of  $O(\gamma_{k+1})$  whereas the second term is of  $O(\gamma_{k+1}^2)$ , showing that a sufficiently small non-zero  $\gamma_{k+1}$  will result in monotonic convergence.  $\square$

In the standard steepest-descent algorithm  $\gamma_{k+1}$  is given by

$$\gamma_{k+1} = \frac{\|G_o^T e_k\|^2}{\|G_o G_o^T e_k\|^2} \quad (7.12)$$

and hence there is no clear mechanism to modify  $\gamma_{k+1}$  so that it would be sufficiently small in terms of Proposition 7.1. Consequently in the next section a new modified steepest-descent algorithm is presented that will result in monotonic convergence for plants with multiplicative uncertainty  $U$  when  $U + U^T$  can be assumed to be a positive-definite matrix.

## 7.1 A modified steepest-descent algorithm

In order to enhance the robustness properties of the standard steepest-descent algorithm consider again the algorithm

$$u_{k+1} = u_k + \gamma_{k+1} G_e^T e_k \quad (7.13)$$

where  $\gamma_{k+1}$  is selected to be the solution of the modified optimisation problem

$$\min_{\gamma_{k+1} \in \mathbb{R}} J(\gamma_{k+1}) \quad (7.14)$$

$$J(\gamma_{k+1}) := \|e_{k+1}\|^2 + w\gamma_{k+1}^2$$

where  $w \in \mathbb{R}$ ,  $w > 0$ . The cost function  $J(\gamma_{k+1})$  in (7.14) reflects two design objectives:

- 1) the first term in  $J(\gamma_{k+1})$  reflects the objective that the tracking error should be small during each iteration
- 2) the second term on the other hand attempts to minimise the magnitude of  $\gamma_{k+1}$ , possibly resulting in a more cautious and robust algorithm when compared to the standard steepest-descent algorithm.

The optimisation problem (7.14) can be solved in a straightforward manner and the optimal solution is

$$\gamma_{k+1} = \frac{\|G_e^T e_k\|^2}{w + \|G_e G_e^T e_k\|^2} \quad (7.15)$$

Convergence analysis for this algorithm is given in the following

**Proposition 7.2** *The algorithm (7.13) where  $\gamma_{k+1}$  is obtained from (7.15) results in  $\|e_{k+1}\| \leq \|e_k\|$ , where equality holds if and only if  $\gamma_{k+1} = 0$ . Furthermore,*

$$\lim_{k \rightarrow \infty} \|e_k\| = 0 \quad \text{and} \quad \lim_{k \rightarrow \infty} \gamma_{k+1} = 0 \quad (7.16)$$

*demonstrating monotonic convergence to zero tracking error.*

**Proof.** Let  $k \in \mathbb{N}_+$  be arbitrary. Selecting a sub-optimal choice  $\gamma_{k+1} = 0$  in the cost function (7.14) yields  $J(0) = \|e_k\|^2$ . Since this choice is sub-optimal it follows that

$$\|e_k\|^2 \geq \|e_{k+1}\|^2 + w\gamma_{k+1}^2 \geq \|e_{k+1}\|^2 \quad (7.17)$$

which shows monotonic convergence. This inequality also shows that  $\|e_{k+1}\| = \|e_k\|$  if and only if  $\gamma_{k+1} = 0$ . Based on (7.17)

$$0 \leq \|e_{k+1}\|^2 \leq \|e_k\|^2 - w\gamma_{k+1}^2 \quad (7.18)$$

Applying induction on this inequality gives

$$0 \leq \|e_0\|^2 - w \sum_{i=1}^{k+1} \gamma_i^2 \quad (7.19)$$

and because  $k$  is arbitrary,  $\sum_{i=1}^{\infty} \gamma_i^2 < \infty$ , and therefore  $\lim_{k \rightarrow \infty} \gamma_k = 0$ . This results in

$$0 = \lim_{k \rightarrow \infty} \gamma_{k+1} = \lim_{k \rightarrow \infty} \frac{\|G_e^T e_k\|^2}{w + \|G_e G_e^T e_k\|^2} \quad (7.20)$$

Because  $G_e$  is assumed to be invertible, this is only possible if  $\lim_{k \rightarrow \infty} e_k = 0$ , which concludes the proof.  $\square$

## 7.2 Robustness analysis

Consider again the case where the true plant includes multiplicative uncertainty, i.e.  $G_e = G_o U$  where  $U$  is the uncertainty and  $G_o$  is the nominal plant model which is used in the update law (7.5) so that

$$u_{k+1} = u_k + \gamma_{k+1} G_o^T e_k \quad (7.21)$$

and  $\gamma_{k+1}$  is computed from  $e_k$  and the nominal plant  $G_o$

$$\gamma_{k+1} = \frac{\|G_o^T e_k\|^2}{w + \|G_o G_o^T e_k\|^2} \quad (7.22)$$

Inserting the optimal  $\gamma_{k+1}$  into equation (7.11), which defines  $\|e_{k+1}\|^2 - \|e_k\|^2$  when  $G_e$  contains multiplicative uncertainty, results in

$$\begin{aligned} \|e_{k+1}\|^2 - \|e_k\|^2 &= \\ &- 2 \frac{\|G_o^T e_k\|^2}{w + \|G_o G_o^T e_k\|^2} e_k^T G_o U G_o^T e_k \\ &+ \frac{\|G_o^T e_k\|^4}{(w + \|G_o G_o^T e_k\|^2)^2} \|G_o G_o^T e_k\|^2 \end{aligned} \quad (7.23)$$

and for strictly monotonic convergence the inequality  $\|e_{k+1}\|^2 - \|e_k\|^2 < 0$  has to be satisfied for a non-zero  $e_k$ . The next proposition shows how this can be achieved by taking  $w$  to be a sufficiently large positive number:

**Proposition 7.3** *Consider the algorithm (7.13) where  $\gamma_{k+1}$  is obtained from (7.15). Assume that  $w$  satisfies the following inequality*

$$w > \frac{1}{2} \frac{\|G_o^T\|^2 \|G_o G_e^T\|^2 \|e_0\|^2}{\sigma_{\min}(G_o U G_o^T)} \quad (7.24)$$

where  $\sigma_{\min}(G_o U G_o^T)$  is the smallest singular value of the positive-definite matrix  $G_o U G_o^T$ . This implies that  $\|e_{k+1}\| < \|e_k\|$  if  $e_k \neq 0$ .

**Proof.** A few lines of algebraic manipulations on (7.23) show that a necessary and sufficient condition for monotonic convergence is that the inequality

$$w + \|G_o G_o^T e_k\|^2 > \frac{1}{2} \frac{\|G_o^T e_k\|^2 \|G_o G_e^T e_k\|^2}{e_k^T G_o U G_o^T e_k} \quad (7.25)$$

holds. From this equation, however, it is not clear how to select  $w$  so that the inequality would hold. In order to achieve this, the term on the right-hand side can be estimated as

$$\frac{\|G_o^T e_k\|^2 \|G_o G_e^T e_k\|^2}{e_k^T G_o U G_o^T e_k} \leq \frac{\|G_o^T\|^2 \|G_o G_e^T\|^2 \|e_k\|^2}{\sigma_{\min}(G_o U G_o^T)} \quad (7.26)$$

and a sufficient condition for convergence becomes

$$w > \frac{1}{2} \frac{\|G_o^T\|^2 \|G_o G_e^T\|^2 \|e_k\|^2}{\sigma_{\min}(G_o U G_o^T)} \quad (7.27)$$

Furthermore, because the initial guess  $u_0$  results in a bounded tracking error  $e_0$ ,  $w$  can be selected so that the inequality (7.27) holds for  $\|e_0\|^2$ . This results in  $\|e_1\|^2 \leq \|e_0\|^2$ , and consequently the inequality (7.27) holds with this particular  $w$  also for  $\|e_1\|^2$ . Using an induction argument, the inequality holds for an arbitrary iteration round  $k$ , implying monotonic convergence.  $\square$

**Remark 7.1** *The estimate for  $w$  can be very conservative because the term  $e_k^T G_o U G_o^T e_k$  is estimated in terms of the smallest singular value of  $G_o U G_o^T$ . Furthermore, excessively large magnitudes of  $w$  can have a negative effect on the convergence speed. This is because a large  $w$  will result in a small  $\gamma_{k+1}$ , implying that  $u_{k+1} \approx u_k$  in such a case. Consequently this proposition should be understood as a formal convergence result, and in practice  $w$  can be selected by resorting to a trial and error approach.*

**Remark 7.2** *In (7.27) the sufficient value of  $w$  decreases as  $\|e_k\|^2$  decreases. This opens up the potential to reduce  $w$  with each successive iteration  $k$  and a reduced  $w$  will possibly result in an increase in the convergence speed.*

**Proposition 7.4** *Under the assumptions of Proposition 7.3 the algorithm converges monotonically to zero tracking error.*

**Proof.** Because  $\|e_k\| \geq \|e_{k+1}\|$ ,  $\lim_{k \rightarrow \infty} \|e_k\|$  exists, i.e.  $\lim_{k \rightarrow \infty} \|e_k\| = E$ ,  $E \geq 0$ . Assume now that  $E > 0$ . There then exists a subsequence  $\{e_{k_j}\}$  of  $\{e_k\}$  so that  $\lim_{k_j \rightarrow \infty} e_{k_j} = e_\infty$ , where  $e_\infty$  satisfies  $\|e_\infty\| = E$ . Consider the sequence  $\{e_{k_j+1}\}$  generated by

$$e_{k_j+1} = (I - \gamma_{k_j+1} G_e G_o^T) e_{k_j} \quad (7.28)$$

Based on Proposition 7.3, the sequence  $\{e_{k_j+1}\}$  satisfies the inequality

$$\lim_{k_j \rightarrow \infty} \|e_{k_j+1}\| < \lim_{k_j \rightarrow \infty} \|e_{k_j}\| = E \quad (7.29)$$

However, the sequence  $\{e_{k_j+1}\}$  is also a subsequence of  $\{e_k\}$ , and therefore it must hold that  $\lim_{k_j \rightarrow \infty} \|e_{k_j}\| \geq E$ . This is a contradiction with (7.29), which concludes the proof.  $\square$

In summary, if  $w$  is selected to be sufficiently large, the algorithm will converge *monotonically* to zero tracking error even if the plant model contains multiplicative uncertainty  $U$ , under the assumption that  $U^T + U$  is a positive definite matrix.

**Remark 7.3** Assume now that  $G_e$  is generated by the transfer function  $G_e(z)$  and  $G_o$  by  $G_o(z)$ . In this case  $U$  is generated by the transfer function  $U(z) = G_o^{-1} G_e(z)$ , and based on Section 5.3 a sufficient condition for  $U + U^T$  being a positive-definite matrix is that  $U(z)$  is a positive-real system. Therefore a sufficient condition for convergence is that  $\arg\{U(e^{j\omega T_s})\}$  lies in the open interval  $(-\pi/2, \pi/2)$ , demonstrating a phase margin of  $90^\circ$ . In summary, with the modified algorithm the weighting factor  $w$  can be always used to compensate for uncertainty in the gain of the plant, but excessive uncertainty in phase can result in divergence.

### 7.3 A connection between NOILC and adjoint algorithm

A connection between NOILC and the adjoint algorithm is established in this section in the nominal case  $G_o = G_e$ . As was shown in Chapter 4, in the non-predictive case the non-causal implementation of the NOILC algorithm can be written as  $u_{k+1} = u_k + G^* e_{k+1}$ , and this implementation is equivalent to

$$\begin{aligned} p_{k+1}(t) &= -K(t)(I + \Gamma\Gamma^T K(t))^{-1} \Phi \Gamma x_{k+1}(t) + \psi_{k+1}(t) \\ u_{k+1}(t) &= u_k(t) + \Gamma^T p_{k+1}(t) \end{aligned} \quad (7.30)$$

where  $K(t)$  and  $\psi_{k+1}(t)$  are generated by the difference equations

$$\begin{aligned} K(t) &= \Phi^T K(t+1)(I + \Gamma\Gamma^T K(t+1))^{-1} \Phi + C^T C \\ \psi_{k+1}(t) &= (I + K(t)\Gamma\Gamma^T)^{-1} (\Phi^T \psi_{k+1}(t+1) + C^T e_k(t+1)) \end{aligned} \quad (7.31)$$

with the boundary conditions  $p(N) = 0$  and  $K(N) = 0$ . Define now  $\Phi_{cl} := \Phi - \Gamma M(t)$ , and  $M(t) := \Gamma^T K(t)(I + \Gamma\Gamma^T K(t))^{-1} \Gamma$ . Writing the process model

$$\begin{cases} x_{k+1}(t+1) = \Phi x_{k+1}(t) + \Gamma u_{k+1}(t), & x_{k+1}(0) = 0 \\ y_{k+1} = C x_{k+1}(t) \end{cases} \quad (7.32)$$

in ‘ $\Delta$ -variables’ gives

$$\begin{cases} \Delta x_{k+1}(t+1) = \Phi \Gamma x_{k+1}(t) + \Gamma \Delta u_{k+1}(t) \\ e_{k+1}(t) = e_k(t) - C \Delta x_{k+1}(t) \end{cases} \quad (7.33)$$

With this notation it can be shown that the closed-loop system satisfies the following equation (this calculation is lengthy and tedious, and therefore it is omitted)

$$\begin{cases} \begin{bmatrix} \Delta x_{k+1}(t+1) \\ p_{k+1}(t+1) \end{bmatrix} = \begin{bmatrix} \Phi_{11,e} & \Phi_{12,e} \\ \Phi_{21,e} & \Phi_{22,e} \end{bmatrix} \begin{bmatrix} \Delta x_{k+1}(t) \\ p_{k+1}(t) \end{bmatrix} + \begin{bmatrix} \Gamma_{11,e} \\ \Gamma_{21,e} \end{bmatrix} e_k(t+1) \\ e_{k+1}(t) = e_k(t) - C \Delta x_{k+1}(t) \end{cases} \quad (7.34)$$

where it is assumed that  $|\Phi_{cl}| \neq 0$  and

$$\Phi_e = \begin{bmatrix} \Phi_{cl} & \Gamma \Gamma^T (I + K(t) \Gamma \Gamma^T)^{-1} \\ 0 & \Phi_{cl}^{-T} \end{bmatrix}, \Gamma_e = \begin{bmatrix} 0 \\ -\Phi_{cl}^{-T} C^T \end{bmatrix} \quad (7.35)$$

In order to compare this algorithm with a steepest descent algorithm

$$u_{k+1} = u_k + \gamma_{k+1} G^* e_k \quad (7.36)$$

write the steepest descent algorithm using the following equivalent form

$$\begin{cases} u_{k+1}(t) = u_k(t) + \gamma_{k+1} \Gamma^T p_{k+1}(t) \\ p_{k+1}(t) = \Phi^T p_{k+1}(t+1) + C^T e_k(t+1) \end{cases} \quad (7.37)$$

Combining (7.37) and (7.33) gives the closed-loop system

$$\begin{cases} \begin{bmatrix} \Delta x_{k+1}(t+1) \\ p_{k+1}(t+1) \end{bmatrix} = \begin{bmatrix} \Phi & \gamma_{k+1} \Gamma \Gamma^T \\ 0 & \Phi^{-T} \end{bmatrix} \begin{bmatrix} \Delta x_{k+1}(t) \\ p_{k+1}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ -\Phi^{-T} C^T \end{bmatrix} e_k(t+1) \\ e_{k+1}(t) = e_k(t) - C \Delta x_{k+1}(t) \end{cases} \quad (7.38)$$

where it is assumed that  $|\Phi| \neq 0$ . Comparing (7.34) and (7.38) it is clear that the Amann algorithm in fact applies the steepest descent algorithm to the closed loop system  $(\Phi_{cl}, \Gamma(I + \Gamma^T K(t) \Gamma)^{-\frac{1}{2}}, C)$ . In other words the feedback term in the control law gives a closed-loop system  $G_{cl}$  and the predictive term is the steepest descent algorithm with fixed step-size

$$u_{k+1} = u_k + G_{cl}^* e_k \quad (7.39)$$

The resulting evolution equation is given by

$$e_{k+1} = (I - G_{cl} G_{cl}^*) e_k \quad (7.40)$$

and it can be seen that the equation will converge geometrically to zero if the singular values of  $G_{cl}$  lie in the open interval  $(0, 2)$ . To see that the state-feedback law achieves this, note that (7.40) has to be equal to the error evolution equation

$$e_{k+1} = (I + G G^*)^{-1} e_k \quad (7.41)$$

Consequently

$$I - G_{cl}G_{cl}^* = (I + GG^*)^{-1} \quad (7.42)$$

and solving for  $G_{cl}G_{cl}^*$  gives  $G_{cl}G_{cl}^* = (I + GG^*)^{-1}GG^*$ , and hence

$$G_{cl} = G(I + G^*G)^{-\frac{1}{2}}U \quad (7.43)$$

where  $U$  is a unitary matrix, i.e.  $U^T U = I$ . From this equation it is clear that the singular values of  $G_{cl}$  lie in the open interval  $(0,2)$ , and the algorithm will converge to zero tracking error geometrically.

**Remark 7.4** *In conclusion, it can be said that the purpose of the feedback term in the control law (7.30) is to scale the plant so that the singular values of the closed-loop system  $G_{cl}$  lie in the open interval  $(0,2)$ . Consequently, the feedforward algorithm  $u_{k+1} = u_k + G_e^* e_k$  (which is the standard steepest descent algorithm) is applied, and due to the scaling the algorithm will converge. However, any other feedback control design methods could be used to achieve this scaling,  $H_\infty$ -control being (perhaps) one of the most natural choices. On the other hand, the modified steepest descent algorithm  $u_{k+1} = u_k + \gamma_{k+1} G_e^T e_k$  results in the error evolution equation  $e_{k+1} = (I - \gamma_{k+1} G_e G_e^T) e_k$ , and consequently the parameter  $\gamma_{k+1}$  scales the singular values of the matrix  $G_e G_e^T$  to be less than unity in magnitude.*

## 7.4 Simulation examples

### 7.4.1 The effect of $w$ on convergence speed

As a first simulation example consider the continuous-time model

$$(p^2 + 2p + 1)y(t) = 0.1(p + 1)u(t) \quad (7.44)$$

which is sampled at  $T_s = 0.1$  second intervals using zero-order hold. The trial length is 6 seconds and the reference signal is selected to be  $r(t) = e^{t/6} \sin(t)$ . Fig. 7.1 shows the  $l_2$ -norm of the tracking error in logarithmic scale as a function of the iteration round. In the simulation it is assumed that there is no uncertainty in the plant model. This figure shows the expected result: namely that an increase in  $w$  will decrease the convergence speed. Note that as  $w \rightarrow 0$ , the standard steepest-descent algorithm is obtained.

### 7.4.2 An example of increased robustness

As an example of the enhanced robustness properties of the modified steepest-descent algorithm, consider the nominal plant model (note that in this example transfer functions are used in order to illustrate the nature of the multiplicative

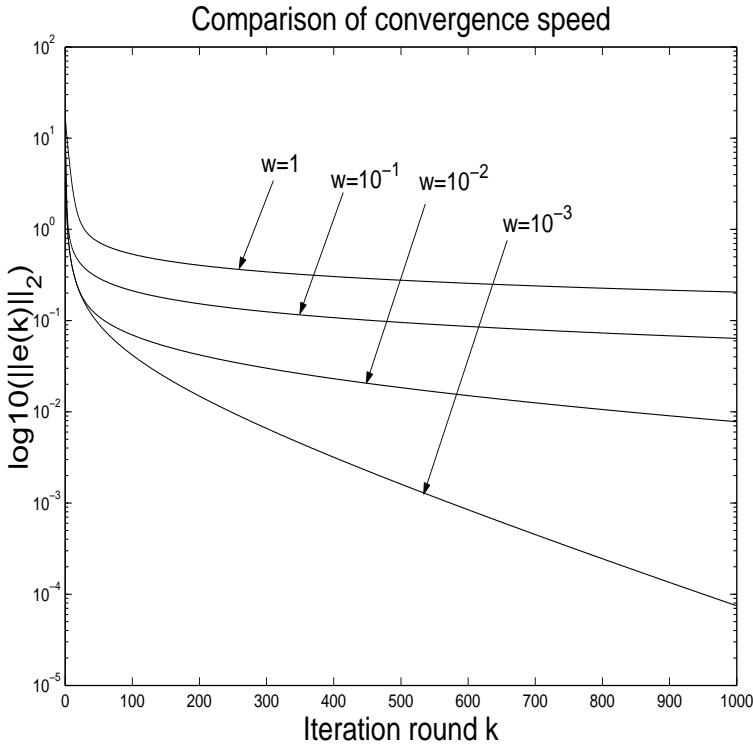


Figure 7.1. Comparison of convergence speed as a function of  $w$  with the plant model (7.44).

uncertainty)

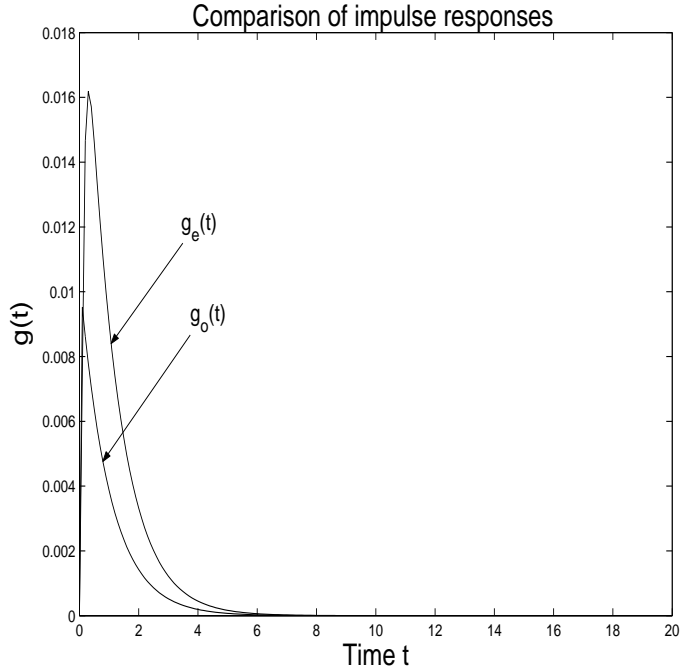
$$G_o(s) = \frac{0.1(s+1)}{s^2 + 2s + 1} \quad (7.45)$$

where the true plant model is given by

$$G_e(s) = \frac{0.1(s+1)}{s^2 + 2s + 1} \cdot \frac{21}{s+10} \quad (7.46)$$

The sampling settings and the reference signal are the same as in the previous example. The additional term in (7.46) describes the unmodelled high-order dynamics. This kind of uncertainty is common in practical applications because either safety and/or economical reasons frequently prevent the running of identification experiments which excite the ‘high-frequency poles’. Fig. 7.2 shows the impulse response of the nominal model and the impulse response of the true plant. From this figure it can be seen that there is no substantial difference between the two impulse responses. It can be shown numerically that the multiplicative uncer-

tainty in the sampled model is positive and hence the theory developed in previous sections is applicable.

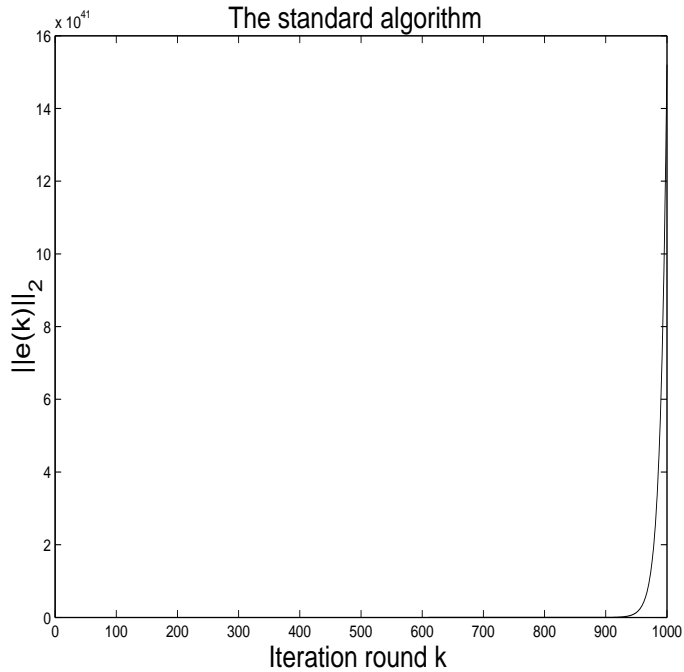


**Figure 7.2.** Comparison of impulse responses.

Fig. 7.3 shows the convergence behaviour if the standard steepest-descent algorithm is used, i.e.  $w = 0$ . From this figure it is clear that the standard algorithm diverges. Fig. 7.4 shows how the modified algorithm performs with different values of  $w$  on a logarithmic scale (the y-axis). This figure shows the expected result: the algorithm converges monotonically to zero tracking error.

## 7.5 Summary

In this chapter the algorithm  $u_{k+1} = u_k + \gamma_{k+1} G_e^T e_k$  was considered, where the learning gain  $\gamma_{k+1}$  is chosen by using parameter optimisation. In the nominal case, when the plant model does not have any uncertainty, it was shown that the algorithm converges monotonically to zero tracking error for an arbitrary discrete-time LTI plant. Assume now the plant model contains multiplicative uncertainty, i.e.

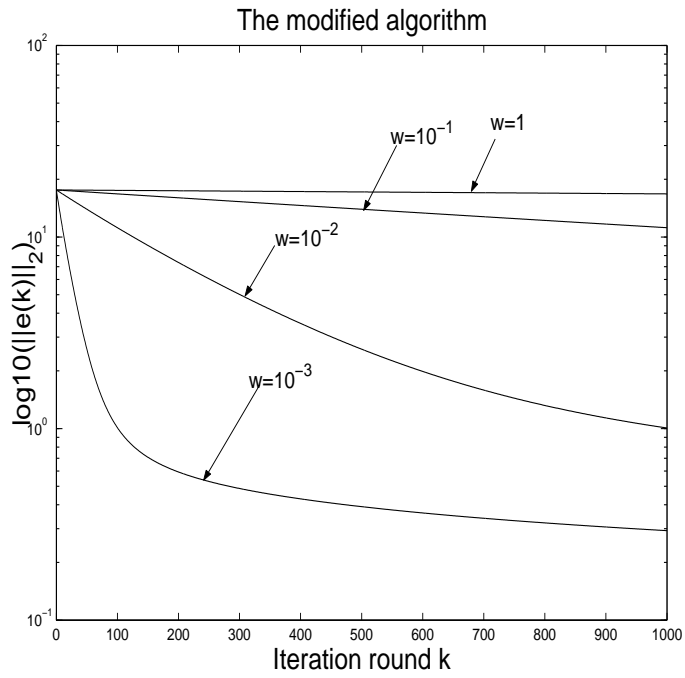


**Figure 7.3.** The convergence behaviour with the standard algorithm.

$G_e = G_o U$  where  $G_e$  is the true plant,  $G_o$  is the nominal plant, and  $U$  is multiplicative uncertainty. In this case the algorithm will still converge *monotonically* to zero tracking error if  $U^T + U$  is a positive-definite matrix and the learning gain is made sufficiently small. In the standard steepest-descent algorithm there is no clear method to guarantee that the learning gain is sufficiently small, whereas in the 'robust' modification the tuning parameter  $w$  gives a straightforward mechanism for finding a balance between convergence speed and robustness.

A connection between the steepest-descent algorithm and NOILC was established (the same connection for the continuous-time case can be found from Amann (1996)): in the NOILC case the feedback component of the controller scales or 'conditions' the plant so that the singular values of  $G_e G_e^T$  are less than unity in magnitude, and the feedforward part applies the standard steepest-descent algorithm with a *fixed* step length on the conditioned plant. The modified algorithm, by contrast, uses the step length (or learning gain)  $\gamma_{k+1}$  to scale adaptively the singular values of  $G_e G_e^T$  to that they become less than unity in magnitude. Which scaling method is more applicable in practical applications is still an open question.

Note that in publication P6 (see Section 1.4) the modified steepest-descent algorithm is applied on an industrial-scale gantry robot, and the algorithm converges to near perfect tracking in 100 iterations. These results are very promising because



**Figure 7.4.** The convergence behaviour with the modified algorithm.

the gantry robot has several nonlinear elements that have not been included in the linear nominal model  $G_o$ .

## 8 Repetitive Control

### 8.1 Problem definition and earlier work

As was explained in Chapter 1, as a starting point in continuous-time Repetitive Control (RC) it is assumed that a SISO model

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t)\end{aligned}\tag{8.1}$$

of the plant in question exists with  $x(0) = x_0$ ,  $t \in [0, \infty)$ . Furthermore,  $A$ ,  $B$ ,  $C$  and  $D$  are finite-dimensional matrices of appropriate dimensions. From now on it is assumed that  $D = 0$ , because in practice it is very rare to find a system where the input function  $u(t)$  has an immediate effect on the output variable  $y(t)$ . Furthermore, a reference signal  $r(t)$  is given, and it is known that  $r(t) = r(t + T)$  for a given  $T$ . The control design objective is to find a feedback controller that makes the system (8.1) to track the reference signal as accurately as possible (i.e.  $\lim_{t \rightarrow \infty} e(t) = 0$ ,  $e(t) := r(t) - y(t)$ ), under the assumption that the reference signal  $r(t)$  is  $T$ -periodic. Note that the only difference between the RC and ILC problem is resetting: in ILC the state of the system is reset at the end of each period (iteration), whereas in RC the state at end of a period is the initial condition for the next period. As will be seen in this chapter, the fact that there is no resetting mechanism in RC, results in very different analysis and design methods for RC. In order to start the analysis of RC systems, note that in the Iterative Learning Control framework a necessary condition for asymptotic convergence is that a controller

$$[Mu](t) = [Ne](t)\tag{8.2}$$

where  $M$  and  $N$  are suitable operators, has to have an internal model or the reference signal inside the operator  $M$ . Because  $r(t)$  is  $T$ -periodic in RC, its internal model is  $1 - \sigma_T$ , where  $[\sigma_T v](t) = v(t - T)$  for  $v : \mathbb{R} \rightarrow \mathbb{R}$ . Hence in Yamamoto (1993) it was suggested that one possible (computationally simple) RC algorithm for the SISO case could be

$$u(t) = u(t - T) + e(t)\tag{8.3}$$

This algorithm has been analysed by several authors, see for example Yamamoto (1993) Arimoto & Naniwa (2000) and Owens *et al.* (2001), and it turns out that a sufficient condition for asymptotic convergence is that the system (8.1) is positive real. The definition of a positive real system from Anderson & Vongpanithred (1973) is given in the following

**Definition 8.1 (A positive real system - continuous-time case)**

Consider the transfer function matrix  $G(s)$  of the system (8.1) where  $G(s) = C(sI - A)^{-1}B + D$ . System (8.1) is positive real

- 1) Each element of the transfer function  $G(s)$  is analytic for  $\text{Re}[s] > 0$
- 2)  $G(s)$  is real for real positive  $s$
- 3)  $G(s) + G(s)^* \geq 0$  for  $\text{Re}[s] > 0$

where the superscript  $*$  denotes complex conjugation.

To see why positivity is required in the SISO case, consider now the following ‘relaxed’ algorithm

$$u(t) = \alpha u(t - T) + Ke(t) \quad (8.4)$$

where  $\alpha \in (0, 1)$  is a relaxation parameter and  $K \in \mathbb{R}, K > 0$ . An equivalent representation of the algorithm is given by

$$[1 + KG]u(t) = \alpha u(t - T) + Kr(t) \quad (8.5)$$

where it is assumed that  $r(\cdot) \in L_2^{loc}[0, \infty)$ . The Laplace-transform of (8.5) becomes (assuming that  $G(s)$  is stable with zero initial conditions)

$$u(s) = \frac{\alpha e^{-sT}}{1 + KG(s)}u(s) + \frac{Kr(s)}{1 + KG(s)} \quad (8.6)$$

Application of the small-gain theorem (see Zhou *et al.* (1996)) shows that a sufficient condition for stability (i.e.  $u(\cdot) \in L_2[0, \infty)$ ) is that

$$\sup_{\omega \geq 0} \left| \frac{\alpha}{1 + KG(j\omega)} \right| < 1 \quad (8.7)$$

where it is assumed that  $\frac{\alpha}{1 + KG(s)}$  is a stable system. Note that if  $\alpha = 1$  this inequality is never met if  $G(s)$  is strictly proper because, for a strictly proper  $G(s)$ ,  $\lim_{\omega \rightarrow \infty} G(j\omega) = 0$  and therefore  $\lim_{\omega \rightarrow \infty} \left| \frac{1}{1 + KG(j\omega)} \right| = 1$ . Stability can be achieved, however, if  $\alpha$  is selected to be sufficiently small, resulting in a non-zero tracking error.

It can be shown that the sufficient condition (8.7) implies that the control law converges to a  $T$ -periodic solution. In the limit the control law (8.4) becomes

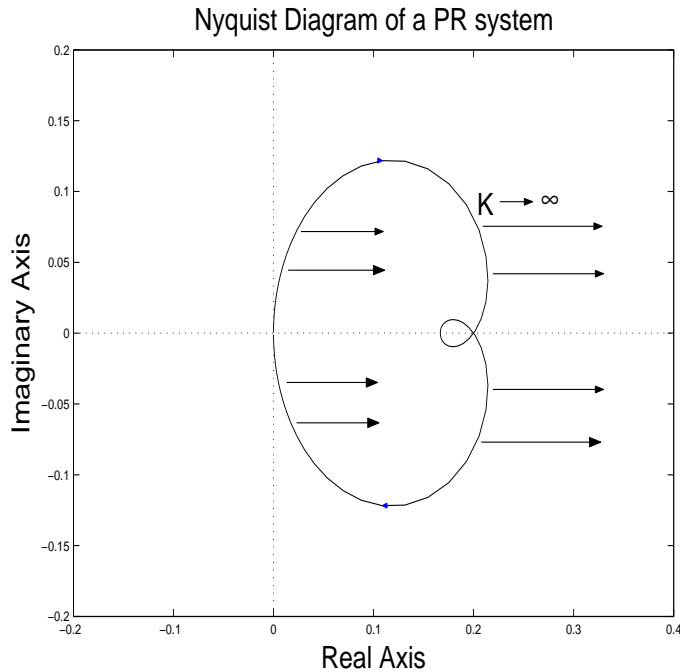
$$u(t) = \frac{K}{1 - \alpha} e(t) \quad (8.8)$$

and  $y(t)$  converges to a  $T$ -periodic solution

$$y(t) = \frac{\frac{K}{1 - \alpha} G}{1 + \frac{K}{1 - \alpha} G} r(t) = \frac{KG}{(1 - \alpha) + KG} r(t) \quad (8.9)$$

If  $\alpha$  now tends to one (giving the original law), it is clear that this simple control law (8.5) generates an infinite feedback gain in the limit based on (8.8) and  $y(t)$  converges to  $r(t)$  based on (8.9). However, a positive real system can tolerate infinite feedback gain, as was shown for example in Owens *et al.* (2001) with a Lyapunov-based approach for the MIMO multi-periodic case. Therefore (8.9) is stable for an arbitrary  $\alpha \in [0, 1]$ , and  $\alpha = 1$  results in  $y(t) = r(t)$ .

In the single-periodic SISO case there is a more visual way to prove that positive realness is a sufficient condition for convergence: as a starting point Definition 8.1 can be understood as a statement that the Nyquist diagram of the positive real system  $G$  lies in the right-half plane (see Fig. 8.1). Furthermore, it is a standard result in classical control theory that a control law  $u(t) = -Ke(t)$  results in a stable closed-loop system if the Nyquist diagram of  $KG$  does not encircle the critical point  $(-1, 0)$ . However, if  $G$  lies in the right-half plane and  $G$  is multiplied with a positive  $K$ , the resulting Nyquist diagram will be still in the right-half plane as shown in Fig. 8.1. Consequently a positive real system can tolerate an arbitrary large feedback gain  $K$ .



**Figure 8.1.** The Nyquist diagram of a positive real system.

## 8.2 A fundamental problem caused by implementation

As was shown in the previous section, the simple repetitive control law (8.4) will result in convergent learning when the original continuous-time system is positive real. However, this result is not as useful as it sounds because in practice it is impossible to implement a delay block using analogue components. Hence if the original plant is discretised with zero-order hold (which models exactly the behaviour of the system at the sampled time points), does the sampled system remain positive real if the original continuous time plant is strictly proper and positive real? To answer this question, first, the definition of a positive real system in the discrete-time case is needed (see Desoer & Vidyasagar (1975) for details):

**Definition 8.2 (A positive real system - discrete-time case)** *Consider the following LTI discrete-time system*

$$\begin{aligned}x(t+1) &= \Phi x(t) + \Gamma u(t) \\ y(t) &= Cx(t) + Du(t)\end{aligned}\tag{8.10}$$

and the corresponding transfer function matrix  $G(z) = C(zI - \Phi)^{-1}\Gamma + D$ . The system (8.10) is said to be positive-real, if

- 1)  $G(z)$  is analytic for  $|z| > 1$
- 2)  $G(z)$  is real for real positive  $z$
- 3)  $\text{Re}[G(e^{j\theta})] \geq 0$  for all  $\theta \in [0, 2\pi]$  (i.e. the Nyquist diagram of  $G(z)$  lies in the right-half plane)

The next proposition shows a rather disappointing result that for a discrete-time system (8.10) a necessary condition for positive realness is that  $D \neq 0$ , and hence strictly proper discrete-time systems cannot cope with infinite feedback gain:

**Proposition 8.1** *Suppose that the system (8.10) is positive real and  $CB \neq 0$  (the assumption that  $CB \neq 0$  is always true for a continuous-time system sampled with zero-order hold). This implies that  $D \neq 0$ .*

**Proof.** Assume that  $D = 0$  but (8.10) is positive real. Consider now the transfer function of (8.10) given by

$$G(z) = C(zI - \Phi)^{-1}\Gamma\tag{8.11}$$

It is a well-known result from classical control theory that the stability of the system can be analysed with the root-locus method, where the roots of the equation  $1 + KG(z)$  are plotted on the complex plane for different values of  $K > 0$  (see Ogata (1973)). Furthermore, according to Ogata (1973), the poles of the closed-loop system approach the open-loop zeros (zeros of  $G(z)$ ) when  $K \rightarrow \infty$ . Because the relative degree of  $G(z)$  is one, it has a zero at  $z = -\infty$ . Hence one of the closed-loop poles converges towards negative infinity as  $K \rightarrow \infty$ , and therefore it cannot stay inside the unit circle for arbitrarily large values of  $K$ . Consequently the Nyquist-diagram of  $G(z)$  does not lie entirely in the right-half plane, and hence system (8.10) is not positive-real when  $D = 0$ .  $\square$

However a similar reasoning as in the previous section shows that, if the algorithm  $u(t) = u(t - T) + Ke(t)$  (now as discrete-time law) gives convergent learning, the algorithm results in infinite feedback gain. Hence in practice it impossible to implement the control law (8.4) without instability and the 'relaxed' version (8.5), which does not give perfect tracking, must be used. Another possibility is to use a finite-dimensional approximation of the delay line (see Lee & Smith (1998)), but again, perfect tracking cannot be achieved.

To overcome this problem, a computationally more complex algorithm is proposed in the next section for the discrete-time case, which results in perfect tracking in the limit.

**Remark 8.1** *In the continuous-time case with  $D = 0$  an infinitely small time delay will destroy stability. This is due the fact that an equivalent condition for positive-realness is that the phase-lag of the plant  $G(j\omega)$  for  $\omega > 0$  does not exceed  $-90^\circ$ . A time delay, however, causes phase-lag*

$$\angle e^{-j\omega T} = -\omega T \text{ rad} \quad (8.12)$$

*Furthermore, because  $D = 0$ ,  $\angle G(j\omega) \rightarrow -\pi/2$  rad as  $\omega \rightarrow \infty$ , and if a time delay is connected in series with  $G(s)$ , there exists  $\omega_0$  so that for  $\omega \geq \omega_0$   $\angle G(j\omega) < -\pi/2$ rad. Consequently if the digital implementation of the control law (8.4) is modelled in continuous-time as the original controller connected in series with a time delay, the resulting system will be unstable.*

### Example 8.1 (Digital implementation can destroy stability)

Consider a continuous-time system

$$(p^2 + 5p + 1)y(t) = (p + 1)u(t) \quad (8.13)$$

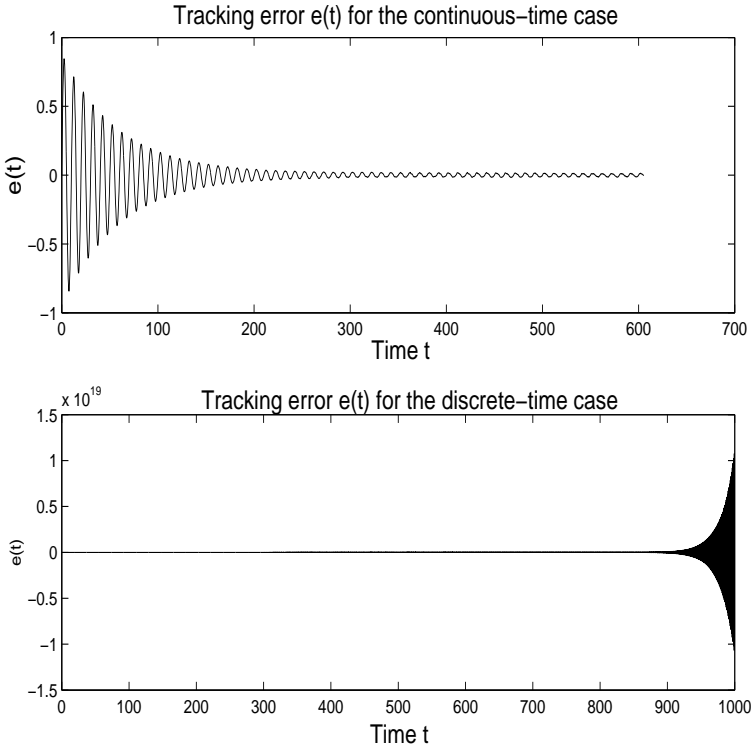
where  $p := \frac{d}{dt}$  and  $t \in [0, \infty)$ . The system is supposed to track a reference signal  $r(t) = \sin(t)$ . The control is chosen to be

$$u(t) = u(t - T) + e(t) \quad (8.14)$$

The corresponding error signal  $e(t)$  is shown in the upper part of Fig. 8.2 and suggests asymptotic convergence. To make the simulation more realistic, the simulation model is modified into a hybrid simulation model in order to take into account the effect of sampling. This achieved by adding zero-order sampling for both  $u(t)$  and  $y(t)$ , which models the effect of A/D and to D/A conversion. The sampling interval is chosen to be 0.1 seconds. Furthermore, the delay-line is implemented with a digital filter but the plant is simulated as it was a continuous-time system. The resulting tracking error is shown for the hybrid simulation model in the lower part of Fig. 8.2. Instability is evident as the theory suggests.

## 8.3 A new algorithm for discrete-time repetitive control

In this section a new algorithm is introduced for discrete-time Repetitive Control. The algorithm design is achieved by combining the polynomial systems approach



**Figure 8.2.** Tracking error  $e(t)$  for Example 8.1.

presented in Blomberg & Ylinen (1983) and optimal control. As a starting point consider a process model defined for  $t \in \mathbb{Z}$

$$A(z^{-1})y(t) = B(z^{-1})u(t) \quad (8.15)$$

where  $A(z^{-1}), B(z^{-1}) \in \mathbb{C}[z^{-1}]$  and  $z^{-1}$  is the backward shift-operator, i.e.  $(z^{-1}v)(k) = v(k-1)$  for  $v(k) \in \mathbb{R}^{\mathbb{Z}}$ . From now on it is assumed that process model (8.15) is both controllable and observable. Furthermore, a  $T$ -periodic reference signal  $r(t)$  is given, i.e.  $r(t+T) = r(t)$  and the control design objective is to make the output  $y(t)$  track this reference signal as accurately as possible by using a suitable feedback controller. Note that, because the reference signal is  $T$ -periodic, the polynomial  $D(z^{-1}) = 1 - z^{-T}$  is an annihilator (or an internal model) for  $r(t)$ , i.e.

$$D(z^{-1})r(t) = r(t) - z^{-T}r(t) = r(t) - r(t-T) = 0. \quad (8.16)$$

By multiplying both sides of (8.15) with  $D(z^{-1})$  gives

$$\begin{aligned} D(z^{-1})A(z^{-1})y(t) &= D(z^{-1})B(z^{-1})u(t) = B(z^{-1})D(z^{-1})u(t) \\ &= B(z^{-1})\tilde{u}(t) \end{aligned} \quad (8.17)$$

where  $\tilde{u}(t) := u(t) - u(t - T)$ . Furthermore, the left-hand side of (8.17) can be written as

$$\begin{aligned} D(z^{-1})A(z^{-1})y(t) &= A(z^{-1})D(z^{-1})y(t) = A(z^{-1})(y(t) - y(t - T)) \\ &= A(z^{-1})(y(t) - r(t) + r(t - T) - y(t - T)) = A(z^{-1})(-e(t) + e(t - T)) \\ &= -D(z^{-1})A(z^{-1})e(t) = \tilde{A}(z^{-1})e(t). \end{aligned} \tag{8.18}$$

where  $\tilde{A}(z^{-1}) := -D(z^{-1})A(z^{-1})$ .

**Remark 8.2** *Note that a similar method has been used in de Roover (1997) to guarantee asymptotic tracking in the repetitive control framework. However, an autonomous state-space representation for the internal model  $D(z^{-1})$  is used, whereas in this chapter the internal model is described using a polynomial description. The benefit from the polynomial approach in the derivation of the algorithm is clarity: it can be seen in a very concrete manner how the internal model blocks the reference from the output  $e(t)$ .*

Combining (8.17) and (8.18) gives

$$\tilde{A}(z^{-1})e(t) = B(z^{-1})u(t) \tag{8.19}$$

which is a controllable and observable dynamical system if  $D(z^{-1})$  and  $B(z^{-1})$  do not have common factors. Consequently by using the internal model  $D(z^{-1})$  the original tracking problem is converted into a regulation problem. Thus, the control objective is to find a feedback controller that drives the output  $e(t)$  of the modified system (8.19) to zero. Several different methods exist for achieving this. One of the more straightforward approaches is to use optimal control: (8.19) has a state-space representation

$$\begin{aligned} x_m(t+1) &= \Phi_m x_m(t) + \Gamma_m \tilde{u}(t) \\ e(t) &= C_m x_m(t) \end{aligned} \tag{8.20}$$

where the dimension of  $x_m(\cdot)$  is  $n + T$ , and  $n$  is the order of the original process model (8.15). Consider now the optimisation problem

$$\min_{\tilde{u} \in l_2} J(\tilde{u}, x_m(0)) \tag{8.21}$$

where

$$\begin{aligned} J(\tilde{u}, x_m(0)) &= \sum_{i=1}^{\infty} e(i)^T Q e(i) + \tilde{u}^T(i) R \tilde{u}(i) \\ &= \sum_{i=1}^{\infty} x_m(i)^T C_m^T Q C_m x_m(i) + \tilde{u}^T(i) R \tilde{u}(i) \end{aligned} \tag{8.22}$$

and  $Q$  and  $R$  are symmetric positive-definite weighting matrices. It is a well-known result from optimal control theory that the solution of the optimisation problem (8.21) is given by the control law  $\tilde{u}(t) = -Kx_m(t)$  or

$$u(t) = u(t - T) - Kx_m(t) \tag{8.23}$$

where  $K$  is given by the equation

$$K = (\Gamma_m^T S \Gamma_m + R)^{-1} \Gamma_m^T S \Phi_m \tag{8.24}$$

and  $S$  is obtained from the algebraic Riccati equation

$$S = \Phi_m^T [S - S\Gamma_m(\Phi_m^T S\Gamma_m + R)^{-1}\Gamma_m^T S]\Phi_m + Q \quad (8.25)$$

Unfortunately, in practice it is impossible to measure the state  $x_m(\cdot)$  directly. However, it is still possible to construct an observer for the state  $x_m(\cdot)$ , i.e. the states are estimated with the following equation

$$\hat{x}_m(t+1) = \Phi_m \hat{x}_m(t) + \Gamma_m \tilde{u}(t) + L(e(t) - C_m x_m(t)) \quad (8.26)$$

where  $L$  is the observer gain and the control law becomes

$$u(t) = u(t-T) - K\hat{x}_m(t) \quad (8.27)$$

Noise can be accounted in a straightforward manner in this framework: suppose that (8.20) would also contain zero mean Gaussian noise terms  $w(t)$  and  $v(t)$  in the following way:

$$\begin{aligned} x_m(t+1) &= \Phi_m x_m(t) + \Gamma_m \tilde{u}(t) + Gw(t) \\ e(t) &= C_m x_m(t) + v(t) \end{aligned} \quad (8.28)$$

Conceptually,  $w(t)$  describes uncertainty in the state-space model, whereas  $v(t)$  describes uncertainty in the measurement process. If the covariance matrix  $Q_n$  of  $v(t)$  and the covariance matrix  $R_n$  of  $w(t)$  are known, it is possible to find an optimal observer gain  $L$  that minimises the variance of the estimation error. It is also a standard result in optimal control (see Lewis & Syrmos (1995)) that by combining the optimal feedback controller and optimal observer the resulting closed loop system is stable, and hence the expected value of  $e(t)$  will go to zero as  $t \rightarrow \infty$ . The flow diagram of the proposed algorithm is shown in Fig 8.3.

**Remark 8.3** *It is important to understand that this approach also works for more complex reference signals. The only requirement is that the reference signal has an annihilator polynomial, and this polynomial does not have common factors with the  $B(z^{-1})$ -polynomial in the process model (8.15). A typical example would be a multi-periodic reference signal  $r(t) = r_1(t) + r_2 + \dots + r_n(t)$  where  $r_i(t) = r_i(t+T_i)$ . In this case the annihilator polynomial is*

$$D(z^{-1}) = D_1(z^{-1})D_2(z^{-1}) \dots D_n(z^{-1}) \quad (8.29)$$

where  $D_i(z^{-1}) = 1 - z^{-T_i}$ . It is also straightforward to show that if there is a  $T$ -periodic load disturbance acting on the input signal, the same controller structure is able to learn the correct control action to cancel out the effect of this disturbance on the output signal  $y(t)$ . All the results in this chapter can be generalised with minor effort to include MIMO-systems.

**Remark 8.4** *The repetitive controller given by equations (8.26) and (8.27) reveals an important fact about the structural differences between ILC and RC in terms of optimal controllers. In iterative learning control a typical controller uses information from previous trial, whereas the repetitive control suggested in this section uses information from a sliding window where the sliding window contains the time-interval  $[t - (T + n), t]$ . This conceptual difference is illustrated in Fig. 8.4.*

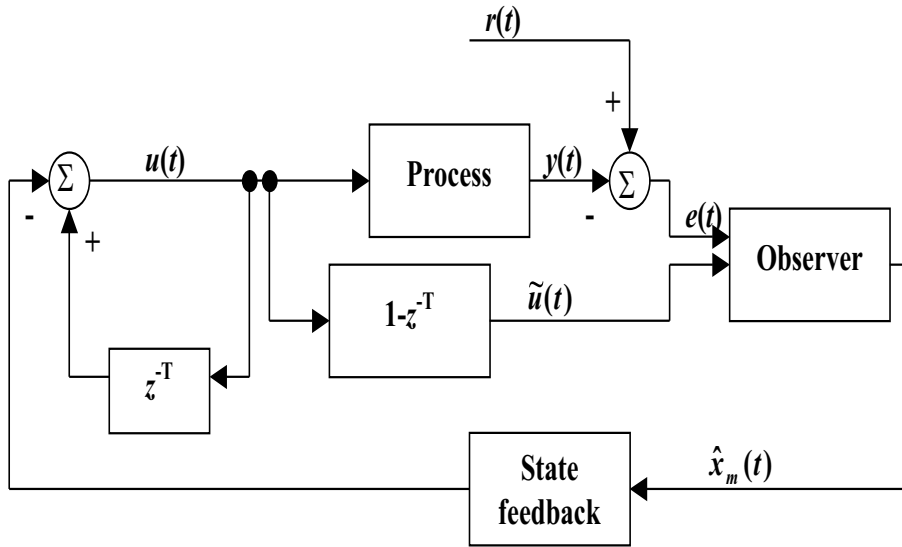


Figure 8.3. The flow diagram of the Repetitive Controller.

## 8.4 Simulation examples

**Example 8.2 (Single-periodic reference signal)** Consider the following noise-free dynamical system

$$(q^2 - 0.1851q + 0.006783)y(t) = (0.2011q - 0.06241)u(t) \quad (8.30)$$

where  $q$  is the standard forward shift operator. The objective is to make this system follow the repetitive reference signal in Fig. 8.5. The repetitive control design technique developed in this chapter is used. The weighting matrices are  $Q = 10$ ,  $R = 1$ , and the observer gain  $L$  is selected using a Kalman-filtering approach, where it is expected that the covariance matrices  $Q_n$  and  $R_n$  are unity. Fig. 8.6 shows the  $l_2$ -norm of the tracking error for each period (iteration). The tracking error is seen to approach zero as theory suggests.

**Example 8.3 (Multi-periodic reference signal)** Consider again the same dynamical system (8.30). This time the system is supposed to track a multi-periodic reference signal  $r(t) = \sin(2\pi t/T_1) + \sin(2\pi t/T_2)$  where  $T_1 = 11$  and  $T_2 = 20$  (i.e. the lengths of the periods are not commensurate). The internal model becomes

$$D(z^{-1}) = (1 - z^{-T_1})(1 - z^{-T_2}) = 1 - z^{-T_1} - z^{-T_2} + z^{-(T_1+T_2)} \quad (8.31)$$

The algorithm was run with the same setting as in the previous example. Fig. 8.7 shows the tracking error  $e(t)$ , which converges to zero.

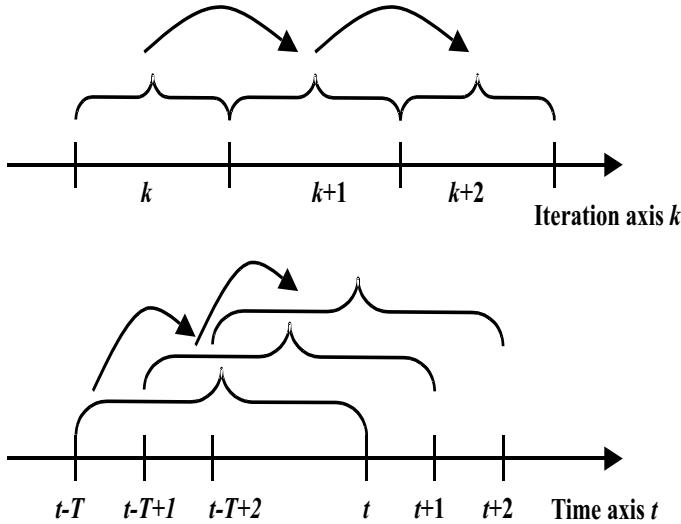


Figure 8.4. The structural difference between ILC and RC .

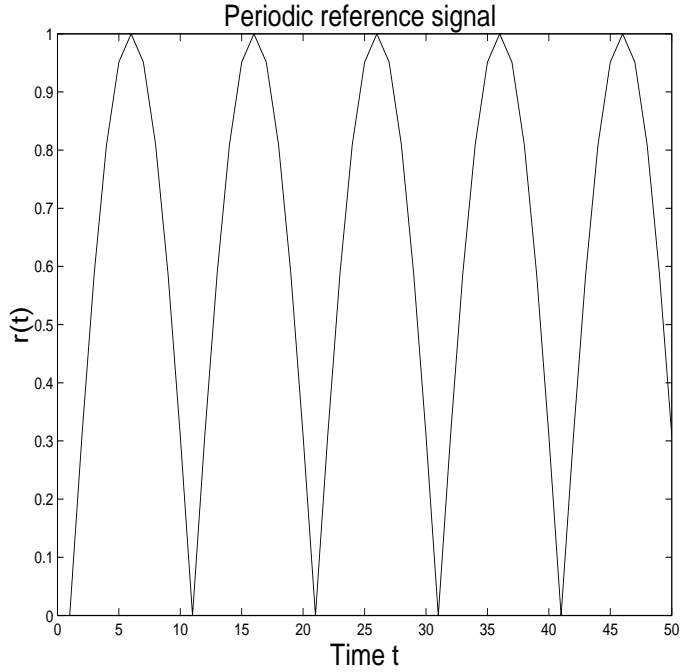
**Example 8.4 (Applying repetitive control in system inversion)** Consider again a system model

$$A(z^{-1})y(t) = B(z^{-1})u(t) \quad (8.32)$$

where  $t \in \mathbb{Z}$ . Furthermore, assume that the input signal  $u(t)$  is  $T$ -periodic ( $u(t) = u(t+T)$ ), and only the output signal  $y(t)$  can be measured. The objective is to estimate the input signal  $u(t)$  based on the measurement  $y(t)$ . This is a typical problem in signal processing, for example, where the dynamics  $A(z^{-1})y(t) = B(z^{-1})u(t)$  describe a channel model, and the receiver aims to estimate the original ‘message’  $u(t)$  based on  $y(t)$ . Suppose now that a simulation model

$$A_e(z^{-1})\tilde{y}(t) = B_e(z^{-1})\tilde{u}(t) \quad (8.33)$$

of the original plant (8.32) exists. For simplicity, assume that  $A_e(z^{-1}) = A(z^{-1})$  and  $B_e(z^{-1}) = B(z^{-1})$ . Under these assumptions the inversion problem can be solved with repetitive control by taking the measured  $y(t)$  as the reference variable, and making an output  $\hat{y}(t)$  of the simulation model (8.33) of the plant (8.32) to track the reference signal  $y(t)$ . If the tracking is exact, then the input to the computer model must match the input of the real plant, and hence the input  $\hat{u}(t)$  of the computer model can be used as an estimate of the original message (input)  $u(t)$ . Furthermore, perfect tracking can be naturally obtained by using the repetitive algorithm (8.26) and (8.27) presented in this section. The flow diagram of the resulting algorithm is shown in Fig. 8.8. As a numerical example consider again the model (8.30). The input  $u(t)$  driven into the plant (8.30) and the

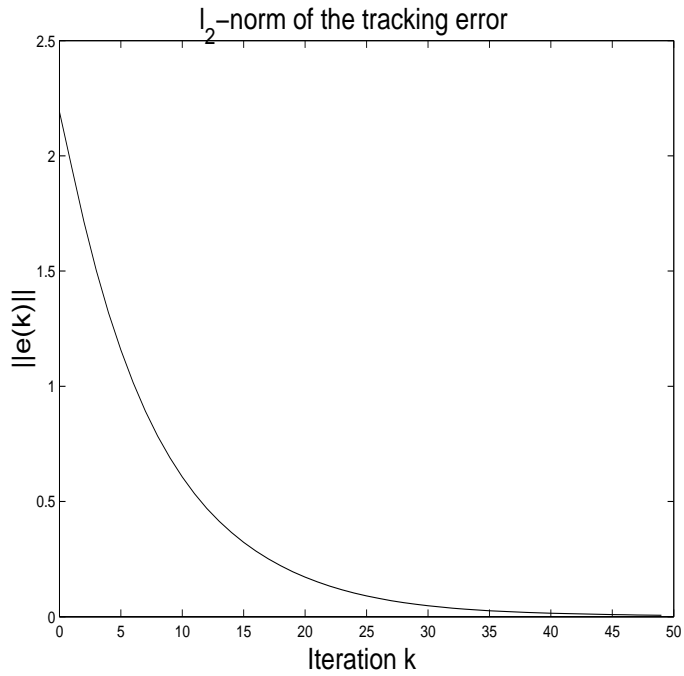


**Figure 8.5.** The periodic reference signal for Example 8.2 .

corresponding output  $y(t)$  are given in Fig. 8.9. The parameters of the algorithm were set to be  $Q = 10$ ,  $R = 1$ ,  $Q_n = 10$ , and  $R_n = 1$ . The resulting estimation error is shown in Fig. (8.10), where  $e_u(t) = u(t) - \hat{u}(t)$ .

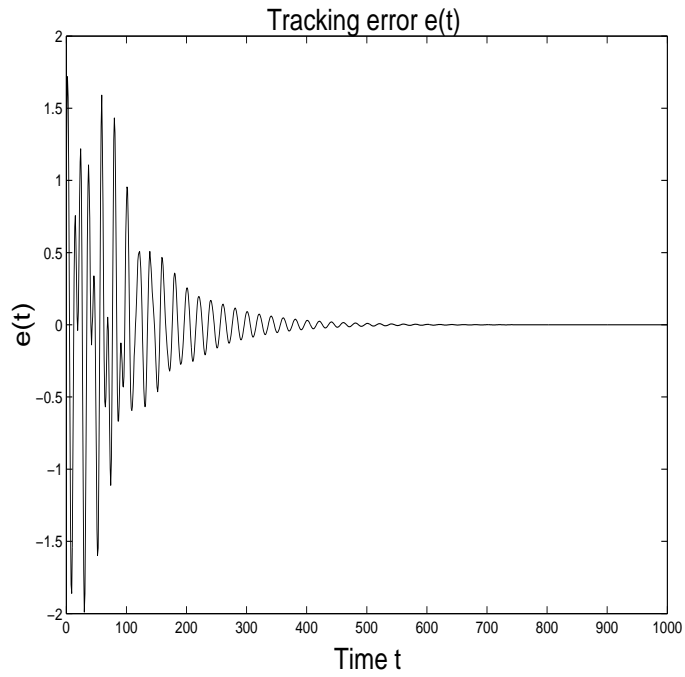
## 8.5 Summary

This chapter demonstrated how a well-known RC algorithm fails to converge if it is implemented using digital information processing. However, the internal model principle states that a RC algorithm has to contain a delay line and therefore digital implementation becomes a necessity. In order to overcome this problem a new discrete-time RC algorithm has been proposed in this chapter. The internal model of the periodic reference signal is used to convert the original tracking problem into a regulation problem. This results in a transform description of the original plant model where the new plant model maps a modified input function to the tracking error. Therefore any stabilising controller can be implemented on the modified plant model in order to drive the tracking error to zero in the limit. In this chapter a standard LQR controller was used together with an observer,



**Figure 8.6.** The norm of the tracking error for Example 8.2 .

because this method gives a straightforward mechanism to find a balance between tracking error signal energy and modified input signal energy (in the  $l_2[0, \infty)$ -sense). Furthermore, the LQR method can easily be extended to minimise the negative effects of measurement noise on the tracking accuracy.



**Figure 8.7.** The tracking error  $e(t)$  for Example 8.3.

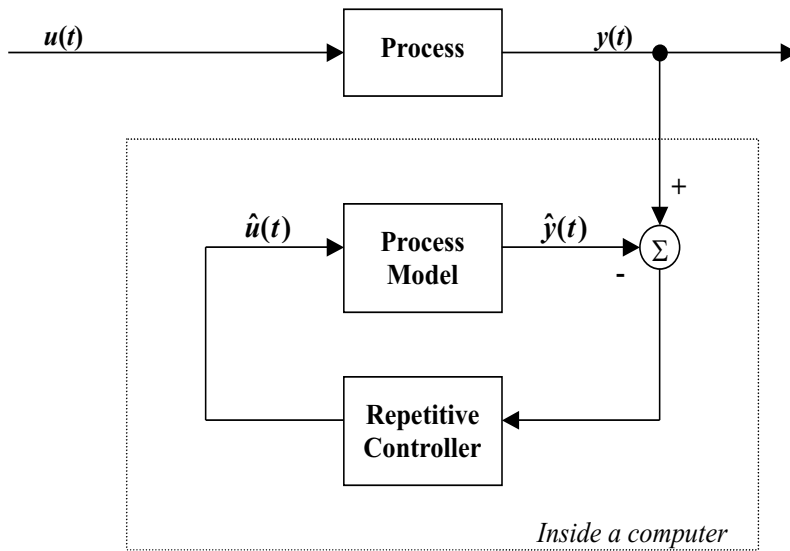


Figure 8.8. The flow diagram of the estimator algorithm.

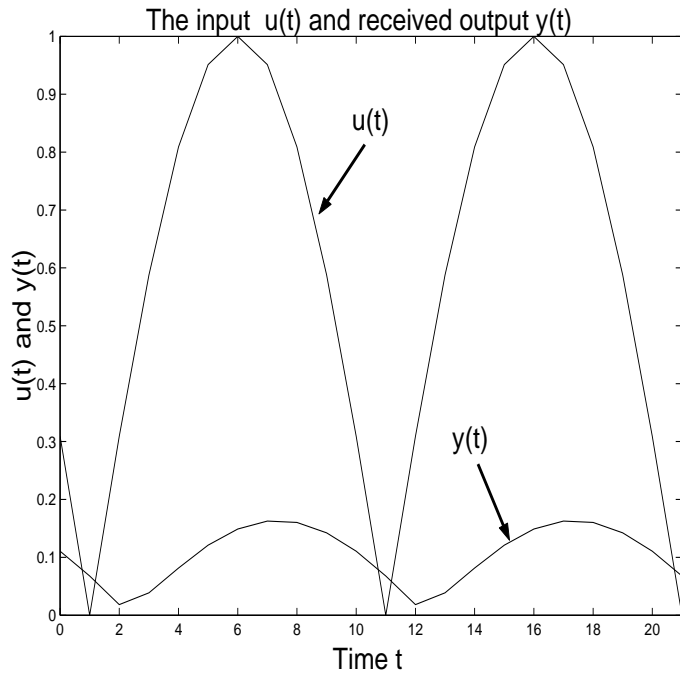


Figure 8.9. The periodic input signal and corresponding output for Example 8.4.

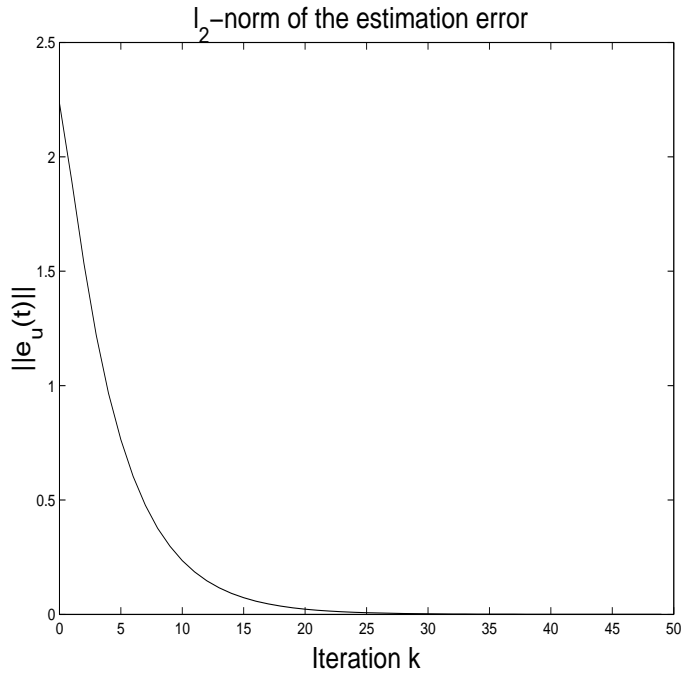


Figure 8.10. The estimation error for Example 8.4.

## 9 Conclusions and future work

### 9.1 Overview

In this thesis new algorithms for both Iterative Learning Control (ILC) and Repetitive Control (RC) have been derived and their convergence properties analysed. In both ILC and RC a dynamical system is tracking a repetitive or periodic reference signal. Because of the repetitive nature of the problem it is possible to use information from the previous executions of the task to improve the tracking performance. This results in learning mechanism where either an ILC or RC algorithm iteratively finds (learns) the input function that results in (near) perfect tracking. The main difference between ILC and RC is resetting: in ILC the state of the system is reset to the original initial condition when the system has reached the final time point in a period (trial). In RC, by contrast, the state of the system at the end of a period becomes the initial condition for the next period, which resembles more the situation in classical feedback systems.

The resetting mechanism in ILC systems requires special attention in terms of mathematical convergence analysis. This is due to the fact after applying ILC to a dynamical system, the resulting system has *two* independent variables, namely time axis and the iteration axis. Furthermore, the resetting mechanism results in a finite time-axis, but (in theory) the iteration axis is infinite. Note that the output of an LTI system can never become unbounded in finite time. Therefore, in ILC, stability along the time-axis does not play such a major role as in classical feedback control. More important is to investigate how the learning mechanism maps a tracking error from the current trial to next trial. This is done by establishing for a given ILC algorithm the learning operator  $L$  that maps  $e_k \rightarrow e_{k+1}$ , i.e.  $e_{k+1} = Le_k$ . After this the question of convergence learning can be answered by analysing the spectral radius  $\rho(L)$  of  $L$  or the operator norm  $\|L\|$  of  $L$ . If the spectral radius is strictly less than unity, the algorithm converges to zero tracking error *asymptotically*. If operator norm is strictly less than unity, the algorithm converges *monotonically* in norm to zero tracking error.

## 9.2 Analysis and design of discrete-time ILC systems

In the research literature several techniques have been used to analyse and design ILC systems. Furthermore, in most of the cases the convergence analysis results are only sufficient, because they are based on the assumption that the time-axis is infinite. In this thesis, however, it has been shown that there is a systematic and rigorous way of analysing discrete-time ILC systems. This theory is based on the fact that linear, time-invariant, discrete-time systems on a finite time-interval can be always represented equivalently with a matrix mapping. The matrix mapping description shows that the ILC problem is in fact a multivariable tracking problem, where the reference signal is a multivariable step function and the plant is a purely static system. As is shown in this chapter, this interpretation can be used to derive rather general convergence tests for high-order ILC algorithms. Furthermore, the Internal Model Principle shows that asymptotic convergence to zero tracking error requires that the corresponding ILC algorithm contains an integrator along the iteration axis.

The algebraic analysis also results in the important observation that most time-invariant ILC algorithms result in repeated poles along the iteration axis, possibly resulting in a poor tracking performance during earlier iteration rounds. Therefore time-varying ILC algorithms should be used to spread these poles into different locations (inside the unit circle) in order to improve the convergence behaviour.

## 9.3 NOILC and predictive extensions

A natural way to design ILC algorithms is to use optimisation. One possible way to do this is to use Norm-Optimal ILC and its predictive extension. In non-predictive NOILC the algorithm performance is measured using a cost function (i.e. a performance index). This cost function measures the norm of tracking error and the norm of the input difference. This reflects the design objective that the norm of the tracking error should be small during each iteration, but at the same time the current input function should not be too different from the input function used during the previous iteration. Convergence analysis originally derived in Amann (1996) shows that this algorithm results in monotonic convergence for an arbitrary LTI plant (either continuous-time or discrete-time). In the predictive extension of the algorithm the cost function also measures the norm of tracking errors and the norm input differences for future iterations. In Amann *et al.* (1998) this algorithm is implemented using a receding horizon principle. Convergence analysis carried out in Amann (1996) and Amann *et al.* (1998) shows that the predictive algorithm with the receding horizon principle results in faster convergence rate when compared to the non-predictive algorithm.

In order to increase even further the convergence speed of the predictive algorithm it is proposed that a convex combination of the input vectors that minimise the predictive cost function for a given iteration should be fed into the real plant. This can be done because of the unique resetting mechanism in ILC – it guaran-

tees that all the optimal inputs  $u_{k+j}$  for  $1 \leq j \leq n$  are calculated using the same initial condition  $x_o$ . As a new theoretical result it is shown that a considerable improvement in terms convergence speed can be achieved with this approach when compared to the receding horizon approach. In addition, it can be argued that the new convex combination approach has a straightforward mechanism to find a balance between robustness and convergence speed.

## 9.4 Parameter Optimal algorithms

In the NOILC case the optimal algorithm was forced to have an internal model of the reference signal by including the term  $\|u_{k+1} - u_k\|^2$  in the cost function. Another way to guarantee the presence of the internal model is to fix *a priori* the structure of the algorithm to be of the form  $u_{k+1} = u_k + K(\beta_{k+1})e_k$ . After the structure has been fixed, the free parameter(s)  $\beta_{k+1}$  of the algorithm is (are) selected using suitable optimisation techniques. The simplest algorithm of this special form is  $u_{k+1} = u_k + \beta_{k+1}e_k(t+1)$ , and in this thesis a cost function  $\|e_{k+1}\|^2 + w\beta_{k+1}^2$  was used to optimise  $\beta_{k+1}$ . Convergence analysis shows that resulting algorithm produces a monotonically decreasing sequence (in norm) of tracking errors. However, in order to guarantee that the algorithm converges to zero tracking error, it has to be assumed that the ‘lifted’ plant satisfies a positivity condition. As a variant of this basic structure the following algorithm  $u_{k+1}(t) = u_k(t) + \beta_{k+1}e_{k+1}(t+1)$  was also considered. This is clearly a non-causal algorithm, but using the plant model it can always written in equivalent causal form where the current input depends on the current state and the previous tracking error. The learning gain  $\beta_{k+1}$  is selected in a similar fashion as in the feedforward case. Convergence analysis shows that if the lifted plant satisfies the same positivity condition as in the feedforward case, the algorithm converges geometrically to zero tracking error. If the original plant does not satisfy the positivity condition, it is shown that for a certain class of discrete-time LTI systems an additional predictive feedback loop can be used to condition the plant so that resulting closed-loop satisfies the positivity condition. After this conditioning, both the feedforward and feedback algorithm can be used to achieve zero tracking in the limit.

To further increase the flexibility in algorithm tuning in this thesis a new *time-varying*, adaptive algorithm  $u_{k+1} = u_k + \beta_{k+1}(t)e_k(t+1)$  was proposed. The learning gains are selected using a similar optimisation routine as with the time-invariant, adaptive algorithms above. It has been shown that this new algorithm results in monotonic convergence to zero tracking error for an arbitrary linear, time-invariant, discrete-time plant. Note that in some sense this algorithm takes the concept of ‘learning’ to a new level (when compared to earlier ILC research) because the algorithm modifies its parameters both along the iteration and time-axis based on what happened during the previous iteration.

## 9.5 High-Order algorithms

One possible way of relaxing the positivity condition on the plant in the feed-forward, time-invariant POILC case could be to add more information from past inputs and tracking errors to the algorithm. The argument is that by using an increased number of past input and error vectors in the algorithm, span where input  $u_{k+1}$  must lie is increased. However, it turns out that resulting high-order Parameter Optimal ILC algorithms also require the positivity of the plant in order to guarantee that the algorithm converges to zero tracking error. A possible explanation for this is collinearity, i.e. as the algorithm converges towards the optimal solution, the tracking error and input vectors become highly collinear. Furthermore, because the input for iteration  $k + 1$  is a linear combination of previous input vectors and tracking error vectors, the possible collinearity suggests that performance of the high-order algorithm should be close to the performance of the first-order algorithm during terminal convergence. In other words, using previous inputs and tracking error vectors in the algorithm does not increase the span where  $u_{k+1}$  has to lie during terminal convergence when compared to first-order algorithm. In order to rectify this problem a set of fixed basis functions is added into the algorithm. Subsequent analysis shows that if these basis functions are selected appropriately, the algorithm will converge monotonically to zero tracking error for an arbitrary discrete-time LTI plant.

## 9.6 A robust steepest-descent algorithm

Chapter 7 proposes a new 'modified' steepest-descent algorithm. The modification is done in order to enhance the robustness properties of the standard steepest-descent algorithm. In the robustness analysis it is assumed that the nominal plant contains multiplicative uncertainty. As a main result it is shown that if this uncertainty satisfies a positivity condition, the algorithm will still converge monotonically to zero tracking error if a tuning parameter in the algorithm is chosen to be sufficiently large. This positivity condition can also be understood in terms of transfer functions: if multiplicative uncertainty matrix  $U$  is generated by a transfer function  $U(z)$ , the positivity result says that the phase of  $U(z)$  has to lie in the range  $(-\pi/2, \pi/2)$ . Therefore uncertainty in gain can be always compensated with the tuning parameter.

The modified steepest-descent algorithm is intimately related to the non-predictive NOILC algorithm. It has been shown that NOILC conditions the original plant with a state feedback controller, and applies the *non-adaptive* steepest-descent algorithm on the conditioned plant. However, the adaptive nature of the modified algorithm seems to be the key to good robustness properties. Consequently, there is a possibility that the NOILC algorithm should be also made adaptive in order to enhance its robustness properties.

The modified algorithm has been applied on an industrial-scale gantry robot with good results. Even when the linear nominal model of the plant neglects the

inherent nonlinearities in the robot system, the algorithm results in near perfect tracking as number of iterations increases.

## 9.7 Repetitive Control

In RC the main objective is to make a dynamical system track a  $T$ -periodic reference signal. For perfect tracking the Internal Model Principle states that for perfect asymptotic tracking the controller has to be of the form  $u(t) = u(t - T) + \delta(t)$ . Consequently several researchers have suggested that the algorithm  $u(t) = u(t - T) + Ke(t)$  could be used for continuous-time systems. However, in thesis it was shown that this algorithm will diverge if it is implemented using sampled data processing. Furthermore, sampled data processing is always required because of the delay line.

In order to overcome this limitation, a new discrete-time RC algorithm was derived in this thesis. The derivation uses the internal model of the reference signal to transform the original tracking problem into a regulation problem. In this thesis the tracking problem is solved using the standard LQR technique, i.e. an optimal state feedback controller is combined with an optimal state observer. Subsequent convergence analysis shows that this algorithm will drive asymptotically the tracking error to zero under mild observability and controllability conditions. In proposed approach the current control action  $u(t)$  depends on the tracking errors  $e(s)$  and inputs  $u(s)$  for  $s \in [0, T]$  (a moving window), which results in a high-order algorithm. However, with modern computation power, the implementation of this algorithm should not be a problem. The algorithm can also generalised straightforwardly for multi-periodic MIMO-systems.

## 9.8 Directions for future research

The work carried out in this thesis has generated a lot of open research problems for future work:

- i) The algebraic analysis in Chapter 3 shows how matrix mappings and ‘super vectors’ can be used to convert the original ILC problem into standard multivariable tracking problem. However, these matrices are highly redundant descriptions of the original dynamical system and they hide the information contained in the original state-space or transfer function descriptions. One possible way of avoiding this problem would be to use two-dimensional transfer functions. In order to achieve this, new theory is needed to take into account the fact that the time-axis is finite and noncausal information processing can be used for data from the previous iterations.
- ii) Robustness analysis of the NOILC algorithm is required. However, as Section 7.3 points out, the modified steepest-descent algorithm and the NOILC algo-

rithm are intimately related. Therefore one possible way forward is to use the robustness results derived in 7.2 in the NOILC case.

- iii) The adaptive algorithms introduced in this thesis have important implications in terms of future research work – as concrete examples they demonstrate that ILC algorithms containing adaptive and time-varying components can result in enhanced convergence properties when compared to fixed parameter ILC algorithms. Hence it can be expected that further research on adaptive (both iteration- and time-wise) learning mechanisms will provide a new useful source of high-performance ILC algorithms for LTI system.
- iv) The algorithms presented in this theses require a model of the plant. As was shown in publication P6, experimental identification of a plant can be both tedious and time-consuming. Therefore it would be important to find purely data-driven ILC algorithms, where the identification of the plant model is also embedded into the learning mechanism. Preliminary work to this direction has been done in Hatzikos *et al.* (2003b), but more work is required to understand what is the best way of combing ILC and identification techniques.
- v) In the single-period case the optimal RC algorithm in Section 8 implies that the correction term  $u(t) - u(t - T)$  is a weighted integral (in the discrete-time sense) of the tracking error  $e(s)$  for  $s \in [t, t - T]$ . Therefore it can be expected that in the continuous-time case the corresponding algorithm structure should be of the form

$$u(t) = u(t - T) + \int_t^{t-T} K(t, s)e(s)ds \quad (9.1)$$

where  $K(t, s)$  is a suitable kernel function. The potential utility of this structure is an interesting problem for future research.

- vi) It is important to experimentally verify the theoretical results presented in this thesis. This work has already commenced in publication P6, where the adjoint algorithm was applied to a multi-axis robot. In the near future also other algorithms from this thesis will be implemented on the same test-rig.

## Bibliography

- Amann, N. (1996) Optimal algorithms for Iterative Learning Control. Ph.D. thesis, University of Exeter.
- Amann, N., Owens, D. & Rogers, E. (1996) Iterative Learning Control for discrete-time systems with exponential rate of convergence. *IEE Proceeding of Control Theory and Applications* 143:217–244.
- Amann, N., Owens, D. & Rogers, E. (1998) Predictive Optimal Iterative Learning Control. *International Journal of Control* 69(2):203–226.
- Anderson, B. & Vongpanithred, S. (1973) *Network Analysis and Synthesis: A Modern System Theory Approach*. Prentice Hall, Englewood Cliffs, NJ.
- Arimoto, S. & Naniwa, T. (2000) Equivalence relations between learnability, output-passivity and strict positive realness. *International Journal of Control* 73(10):824–831.
- Arimoto, S., Kawamura, S. & Miyazaki, F. (1984) Bettering operations of robots by learning. *Journal of Robotic Systems* 1:123–140.
- Astrom, K. & Wittenmark, B. (1984) *Computer Controlled Systems: Theory and Design*. Prentice-Hall.
- Bien, Z. & Xu, J. (1998) *Iterative Learning Control: Analysis, Integration, and Application*. Kluwer Academic.
- Blomberg, H. & Ylinen, R. (1983) *Algebraic Theory for Multivariable Linear Systems*. Academic Press.
- Camacho, E. & Bordons, C. (1998) *Model Predictive Control*. Springer.
- Chen, Y. & Moore, K. (2000) Comments on United States patent 3,555,252 - learning control of actuators in control systems. In: *Proc. of the 2000 International Conference on Automation, Robotics, and Control*. Singapore.
- Cryer, B., Nawrocki, P. & Lund, R. (1976) A road simulation system for heavy duty vehicles. Tech. Rep. 760361, Society of Automotive Engineers.
- Cuyper, J. D., Coppens, D., Liefoghe, C. & Debille, J. (1999) Advanced system identification methods for improved service load simulation on multi-axial test rigs. *European Journal of Mechanical and Environmental Engineering* 44(1):27–39.
- de Roover, D. (1997) Motion control of a wafer stage - a design approach for speeding up IC production. Ph.D. thesis, Delft University of Technology.
- Desoer, C. & Vidyasagar, M. (1975) *Feedback Systems: Input-Output Properties*. Academic Press Inc.
- Dodds, C. & Plummer, A. (2001) Laboratory road simulation for full vehicle testing - A Review. In: *Proc. of SAE Symposium on International Automotive Technology*. Pune, India, pp. 487–494. (SAE 2001-01-0047).
- Edwards, J. B. & Owens, D. (1982) *Analysis and Control of Multipass Processes*. Research Studies Press.
- Francis, B. & Wonhan, W. (1975) The internal model principle for linear multivariable regulators. *Appl. Math. Opt.* pp. 107–194.

- Fung, R., Huang, J., Chien, C. & Wang, Y. (2000) Design and application of continuous time controller for rotation mechanisms. *International Journal of Mechanical Sciences* 42:1805–1819.
- Furuta, K. & Yamakita, M. (1987) The design of learning control systems for multivariable systems. In: *Proceedings of the IEEE International Symposium on Intelligent Control*. Philadelphia, Pennsylvania, U.S.A., pp. 371–376.
- Gantmacher, F. (1959) *The Theory of Matrices*. Chelsea Publishing Company.
- Garimella, S. & Srinivasan, K. (1996) Application of Repetitive Control to eccentricity compensation in rolling. *Journal of Dynamic Systems Measurement and Control-Transactions of the ASME* 118:657–664.
- Gorinevsky, D. M. (1992) Direct learning of feedforward control for manipulator path tracking. In: *Proc. IEEE International Symp. on Intelligent Control*. Glasgow, United Kingdom.
- Gunnarsson, S. & Norrlöf, M. (2001) On the design of ILC algorithms using optimisation. *Automatica* 37:2011–2016.
- Hatzikos, V., Hätönen, J. & Owens, D. (2003a) Basis functions and genetic algorithms in Norm-Optimal Learning Control. In: *Proceedings of the IFAC International Conference on Intelligent Control Systems and Signal Processing*. Faro, Portugal.
- Hatzikos, V., Hätönen, J. & Owens, D. (2003b) Basis functions, identification and genetic algorithms in Norm-Optimal Iterative Learning Control. In: *Proceedings of the European Control Conference*. Cambridge, UK.
- Hatzikos, V., Hätönen, J. & Owens, D. (2003c) An evolutionary based optimisation method for nonlinear Iterative Learning Control systems. In: *Proceedings of the American Control Conference*. Denver, U.S.A.
- Hatzikos, V., Hätönen, J. & Owens, D. (2004) Genetic algorithms in nonlinear Norm-Optimal Iterative Learning Control. *International Journal of Control* 77(1):188–197.
- Hillerström, G. (1996) Adaptive suppression of vibrations - a Repetitive Control approach. *IEEE Transactions on Control Systems Technology* 3(1):72–78.
- Hillerström, G. & Sternby, J. (1994) Application of Repetitive Control to a peristaltic pump. *the ASME Journal of Dynamic Systems, Measurement and Control* 116(4):786–789.
- Inouye, T., Nakano, M., Kubo, T., Matsumoto, S. & Baba, H. (1981) High accuracy control of a proton synchrotron magnet power supply. In: *Proceedings of the 8th IFAC World Congress*. Kyoto, Japan.
- Kaczorek, T. (1993) *Linear Control Systems - Synthesis of Multivariable Systems and Multidimensional Systems*, volume 2. Research Studies Press (John Wiley).
- Kaneko, K. & Horowitz, R. (1997) Repetitive and adaptive control of robot manipulators with velocity estimation. *IEEE Trans. on robotics and automation* 13:204–217.
- Kobayashi, Y., Kimura, T. & Yanabe, S. (1999) Robust speed control of ultrasonic motor based on  $H_\infty$  control with repetitive compensator. *JSME Int. Journal Series C* 42:884–890.
- Kreyszig, E. (1978) *Introductory Functional Analysis with Applications*. John Wiley.
- Kucera, V. (1979) *Discrete Linear Control-The Polynomial Equation Approach*. John Wiley & Sons.
- Lancaster, P. & Tismenetsky, M. (1985) *The Theory of Matrices*. Academic Press.
- Lee, J. & Lee, J. (1993) Design of Iterative Learning Controller with VCR servo system. *IEEE Transactions on Consumer Electronics* 39:13–24.
- Lee, K., Bang, S., Yi, S. & Yoon, S. (1996a) Iterative Learning Control of heat up phase for a batch polymerization reactor. *Journal of Process Control* 6(4):255–262.
- Lee, K., Kim, W. & Lee, J. (1996b) Model-based Iterative Learning Control with quadratic criterion for linear batch processes. *Journal of Control, Automation and Systems Engineering* 3:148–157.
- Lee, R. & Smith, M. (1998) Robustness and trade-offs in Repetitive Control. *Automatica* 34:889–896.
- Lewis, F. L. & Syrmos, V. L. (1995) *Optimal Control*. John Wiley & Sons, Inc.

- Longman, R. W. (2000) Iterative Learning Control and Repetitive Control for engineering practise. *International Journal of Control* 73(10):930–954.
- Luenberger, D. G. (1969) *Optimisation by Vector Space Methods*. John Wiley & Sons.
- MacLane, S. & Birkoff, G. (1971) *Algebra*. MacMillan.
- Mahawan, B. & Luo, Z. (2000) Repetitive Control of tracking systems with time-varying periodic references. *International Journal of Control* 73(1):1–10.
- Moore, K. (1993) *Iterative Learning Control for Deterministic Systems*. Springer-Verlag.
- Moore, K. (1998a) Multi-loop control approach to designing Iterative Learning Controllers. In: *Proc. of the 37th IEEE Conference on Decision and Control*. Tampa, USA.
- Moore, K. (2000a) A matrix fraction approach to high-order Iterative Learning control-2-D dynamics through repetition-domain filtering. In: *Proc. of the 2nd International Workshop on Multidimensional (nD) Systems*. Czocha Castle, Lower Silesia, Poland, pp. 99–104.
- Moore, K. (2000b) A non-standard Iterative Learning Control approach to tracking periodic signals in discrete-time non-linear systems. *International Journal of Control* 70(10):955–967.
- Moore, K. & Bahl, V. (2000) Iterative Learning Control for multivariable systems with an application to mobile robot path tracking. In: *Proc. of the 2000 IEEE International Conference on Automation, Robotics and Control*. Singapore.
- Moore, K. L. (1998b) Iterative Learning Control: An expository overview. *Applied and Computational Control, Signal Processing and Circuits* 1(1).
- Ogata, K. (1973) *Discrete-time Control Systems*. Prentice Hall, Englewood Cliffs, NJ.
- Owens, D. & Feng, K. (2003) Parameter optimization in Iterative Learning Control. *International Journal of Control* 76(11):1059–1069.
- Owens, D., Li, L. & Banks, S. (2001) MIMO-multiperiodic Repetitive Control systems: a Liapunov analysis. In: *Proceedings of the IFAC Workshop on periodic systems and control*. Cernobbio-Como, Italy.
- Owens, D., Rogers, E. & Moore, K. (2002) Analysis of linear iterative learning scheme using repetitive process theory. *Asian Journal of Control* 4(1):68–91.
- Owens, D. H. & Feng, K. (2002) Parameter optimisation in Iterative Learning Control. Research Report 822, Department of Automatic Control and Systems Engineering, The University of Sheffield.
- Pugh, C. (2002) *Real Mathematical Analysis*. Springer-Verlag.
- Rogers, E. & Owens, D. (1992) *Stability Analysis for Linear Repetitive Processes*. Springer-Verlag.
- Smith, C., Takeuchi, K. & Tomizuka, M. (1999) Cost effective Repetitive Controllers for data storage devices. In: *Proceedings of the 14th IFAC World Congress*. Beijing, China.
- Tao, K. M., Kosut, R. L. & Aral, G. (1994) Learning feedforward control. In: *Proc. of the American Control Conference*. Baltimore, Maryland, USA.
- Togai, M. & Yamano, O. (1985) Analysis and design of an optimal learning control scheme for industrial robots: a discrete system approach. In: *Proc. 24th IEEE Conf. on Decision and Control*. Ft. Lauderdale, Florida, USA.
- Tomizuka, M. (1992) Discrete time Repetitive Control algorithms - regulation under periodic disturbances. In: *Proceedings of the IEEE Workshop on Advanced Motion Control*. Nagoya, Japan.
- Townley, T. (2002) Predictive Iterative Learning Control. Ph.D. thesis, University of Exeter.
- Uchiyama, M. (1978) Formation of high speed motion pattern of mechanical arm by trial. *Transactions of the Society of Instrumentation and Control Engineers* 19(5):706–712.
- Yamamoto, Y. (1993) *Essays on Control: Perspectives in the Theory and Its Applications*, Birkhauser, chap. Learning Control and Related Problems in Infinite-Dimensional Systems, pp. 191–222.
- Zhou, K. & Wang, D. (2001) Digital repetitive learning controller for three-phase cvcf pwm inverter. *IEEE Transactions on Industrial Electronics* 48(4):820–830.
- Zhou, K., Doyle, J. & Glover, K. (1996) *Robust Optimal Control*. Prentice-Hall.
- Zilouchian, A. (1994) An Iterative Learning Control technique for a dual arm robotic system. In: *Proc. of the 1994 IEEE International Conference on Robotics and Automation*. San Diego, CA, USA, pp. 1528–1533.