Linköping Studies in Science and Technology. Dissertations
No. 653

Iterative Learning Control
Analysis, Design, and Experiments

Mikael Norrlöf

Department of Electrical Engineering
Linköpings universitet, SE–581 83 Linköping, Sweden

Linköping 2000

Cover Illustration:
The robot shown on the front page is the ABB IRB1400 that has been used in the experiments in the thesis. The diagram, lower left, is from Figure 4.2 and it illustrates the stability condition for ILC systems in the frequency domain. The block diagram, upper left, from Figure 7.1 shows an abstract view of the implementation of the functions *data-input* and *data-logger* in the robot control system. In the background the implementation of the adaptive ILC scheme from Algorithm 14.1 is shown.

Iterative Learning Control: Analysis, Design, and Experiments

mino@isy.liu.se
http://www.control.isy.liu.se
Division of Automatic Control
Department of Electrical Engineering
Linköpings universitet
SE–581 83 Linköping
Sweden

ISBN 91-7219-837-0 ISSN 0345-7524

Printed by Linus & Linea AB, Linköping, Sweden 2000

To Anna

Abstract

In many industrial robot applications it is a fact that the robot is programmed to do the same task repeatedly. By observing the control error in the different iterations of the same task it becomes clear that it is actually highly repetitive. Iterative Learning Control (ILC) allows to iteratively compensate for and, hence, remove this repetitive error.

In the thesis different aspects of iterative learning control are covered. Although stability is the most important in practice the design aspect is also highlighted. Several design schemes for iterative learning control methods are presented, including first order as well as second order iterative learning control. An adaptive approach to iterative learning control is also discussed. Many of the suggested design methods are also given with stability and robustness results.

The application, industrial robot control, that has been used as a testbed throughout the thesis is described. The description includes a general discussion on robot modeling and control as well as a specific discussion on the implementation of the functions needed in the commercial robot control software in order to make it possible to apply iterative learning control.

The suggested iterative learning control design methods are all tested on the robot. Some practical aspects on the path following problem for industrial robots using iterative learning control are discussed. A potential solution to the path tracking problem using additional sensors is given, although it is not yet implemented on the robot.

Preface

Some of the results presented in the thesis have been published before and the main references are:

The work presented in Part I and Part II is based upon the licentiate thesis,

> M. Norrlöf. On analysis and implementation of iterative learning contol. Licentiate thesis LIU-TEK-LIC-1998:62 Linköping Studies in Science and Technology. Licentiate Thesis No 727, Department of Electrical Engineering, Linköpings universitet, Oct 1998

also some of the algorithms and examples in Chapter 8, Part III, have the same origin. Some of the results in Norrlöf (1998) have also been published in the following conference papers,

> S. Gunnarsson and M. Norrlöf. On the use of learning control for improved performance in robot control systems. In *Proceedings of the European Control Conference 1997*, Brussels, Belgium, July 1997a

> S. Gunnarsson and M. Norrlöf. Some experiences of the use of iterative learning control for performance improvement in robot control systems. In *Preprints of the 5th IFAC symposium on robot control*, volume 2, Nantes, France, Sep 1997c

In Part I some of the results have also been published in a technical report,

> M. Norrlöf and S. Gunnarsson. Disturbance aspects of iterative learning control. Technical Report LiTH-ISY-R-2261, Department of Electrical Engineering, Linköping University, May 2000a

this material has also been accepted for publication in the journal, *Engineering Applications of Artificial Intelligence*.

Chapter 6 has been updated, compared to Norrlöf (1998), with some results from

> M. Norrlöf. Modeling of industrial robots. Technical Report LiTH-ISY-R-2208, Department of Electrical Engineering, Linköping University, Dec 1999

In Part III, Chapter 8, additional results, compared to Norrlöf (1998), come from a technical report and two conference papers,

> S. Gunnarsson and M. Norrlöf. On the design of ILC algorithms using optimization. Technical Report LiTH-ISY-R-2209, Department of Electrical Engineering, Linköping University, Dec 1999a

S. Gunnarsson and M. Norrlöf. Some aspects of an optimization approach to iterative learning control. In *Proc. of the 38th IEEE Conference on Decision and Control*, Pheonix, Arizona, USA, Dec 1999b

M. Norrlöf and S. Gunnarsson. A model based iterative learning control method applied to 3 axes of a commercial industrial robot. In *Preprints of the 6th IFAC symposium on robot control*, Vienna, Austria, Sep 2000b

The results from the technical report have been submitted for publication in *Automatica*. In Chapter 9, the example is taken from Norrlöf and Gunnarsson (2000b) and the presented solution comes from

S. Gunnarsson and M. Norrlöf. terative learning control of a flexible mechanical system using accelerometers. In *Preprints of the 6th IFAC symposium on robot control*, Vienna, Austria, Sep 2000

The material in Part IV has previously appeared in,

M. Norrlöf. Analysis of a second order iterative learning controller. Technical Report LiTH-ISY-R-2181, Department of Electrical Engineering, Linköping University, Feb 2000b

M. Norrlöf and S. Gunnarsson. A frequency domain analysis of a second order iterative learning control algorithm. In *Proc. of the 38th IEEE Conference on Decision and Control*, Pheonix, Arizona, USA, Dec 1999

and some results will appear in

M. Norrlöf. Comparative study on first and second order ILC – frequency domain analysis and experiments. In *Proc. of the 39th IEEE Conference on Decision and Control*, Sydney, Australia, Dec 2000c

The final part, Part V, contains an edited version of the technical report,

M. Norrlöf. Adaptive iterative learning control algorithms for disturbance rejection. Technical Report LiTH-ISY-R-2244, Department of Electrical Engineering, Linköping University, May 2000a

Acknowledgments

First I'd like to thank my supervisor, Dr. Svante Gunnarsson. Without your excellent guidance and your remarkable patience I don't think I would ever have finished this work. I also want to thank the people that have read and given me comments on different versions of the manuscript, Dr. Torgny Brogårdh, Dr. Svante Gunnarsson, and Professor Lennart Ljung. Your comments and suggestions have improved the final version of the thesis considerably. The things still wrong or unclear are completely my own fault.

I'm also grateful to Professor Lennart Ljung for letting me join the group. I'm happy and proud to be a student in your group.

Many people have contributed in one way or another to the results in the thesis. I'd just like to mention Lic. Eng. Fredrik Tjärnström and Dr. Valur Einarsson who have answered my (often trivial) questions and always have had time to discuss topics of my research (but also other things :-). For all the help that I've got with the lab I also want to thank our research engineer Sören Hansson. I'm also grateful to Måns Östring for helping me with the modeling of the robot (but also for keeping the LaTeX-installation up and running!).

All the friends in the Control & Communication group at Linköpings universitet are gratefully acknowledged. The coffee-breaks, with all the outrageous discussions on everything from philosophy to sport (well sometimes also research), the floorball games and all the other activities have given me many nice memories that I will never forget.

I'd like to thank the people at ABB Robotics for helping me with all the practical problems concerning the robot and the robot programs. In particular I'd like to mention Dr. Geir Hovland, Stig Moberg, Peter Ericsson, Dr. Steve Murphy, Dr. Håkan Fortell, Jan Rögde, and, of course, the founder of the project and my industrial mentor Dr. Torgny Brogårdh. Without your help and support this project would never have been possible.

This work was supported by ABB Robotics AB within NUTEK's Center of Excellence ISIS at Linköpings universitet, which is gratefully acknowledged. The ECSEL graduate school has provided me with PhD courses that has broaden my horizons and given me the opportunity to meet and discuss with PhD students from other departments.

I would also like to thank my family. Your endless love & support have brought me where I am today. Thank you!! Grazie also to *"piipii"* and all his family. Alla prossima partita a calcio...

Finally, I would like to thank Anna. I guess words are not enough... Your love, support, and encouragement have given me the energy to finish this work.
– *Ti Amo!*

Linköping, September 2000

Mikael Norrlöf

Contents

Notation

Symbols

$\mathbb{R}$	the sets of real numbers
$\mathbb{C}$	the set of complex numbers
$\mathbb{N}$	the set of natural numbers (0,1,2,...)
$\mathbb{Z}^+$	the set of positive integers (1,2,...)
I	the identity matrix
$x_k(t)$	time signal from iteration k, defined for $0 \leq t \leq t_f$
$\boldsymbol{x}_k$	vectorized time signal from iteration k, $\boldsymbol{x}_k = \begin{pmatrix} x_k(0) & \dots & x_k(t_f) \end{pmatrix}^T$
$X_{k,\omega}$	Fourier transform of $x_k(t)$ evaluated at frequency ω (equivalent to $X_k(\omega)$)
$\Phi_{x_k}(\omega)$	spectrum for the signal $x_k(t)$
G(q)	discrete linear time invariant system representation
$\boldsymbol{G}$	matrix realization of the discrete time system $G(q)$
G_ω	Fourier transform of $G(q)$ evaluated at frequency ω (equivalent to $G(e^{i\omega t_s})$)
t_s,t_f	sampling time, final time (length of) iteration

Operators and Functions

q	time shift operator, $q^{-1}x_k(t) = x_k(t - t_s)$
q_k	iteration shift operator, $q_k^{-1}x_k(t) = x_{k-1}(t)$
$\dot{x}, \ddot{x}, x^{(i)}$	first, second and i^{th} derivative of x w.r.t. t
$\equiv, \Longrightarrow$	equivalent to, implication
$\exists$, $\forall$	existential and universal quantifier
A^*	adjoint of matrix A. $A = (a_{ij})$; $A^* = (\bar{a}_{ji})$, where $\bar{a}$ is complex conjugate of a
$A^\dagger$	matrix pseudo inverse, $A^\dagger = A^T(AA^T)^{-1}$
$\rho(A)$	spectral radius (maximum absolute value of the eigenvalues of A)
$\text{tr}\, A$	trace of matrix A
$\underline{\sigma}(A)$, $\overline{\sigma}(A)$	maximum and minimum singular value of matrix A
$E\{\cdot\}$	mathematical expectation
$\text{Var}\{\cdot\}$	variance
$\text{sign}\ x$	sign function, returns $+1$ if $x > 0$, -1 if $x < 0$, and 0 if $x = 0$
$\|v\|_2$	vector 2-norm, defined as $\|v\|_2 = \sqrt{\sum_{i=1}^{n} \|v_i\|^2}$
$\|v\|_\infty$	vector ∞-norm, defined as $\|v\|_\infty = \max_{1 \le i \le n} \|v_i\|$

Abbreviations

ABB	Asea Brown Boveri
ARX	AutoRegressive with eXternal input
CAD/CAM	Computer Aided Design/Computer Aided Manufacturing
DFT	Discrete Fourier Transform
DOF	Degrees of Freedom
DSP	Digital Signal Processor
ETFE	Empirical Transfer Function Estimate
FTP	File Transfer Protocol
IFT	Iterative Feedback Tuning
ILC	Iterative Learning Control
I/O	Input/Output
IRB	Industrial Robot
ISIS	Information Systems for Industrial Control and Supervision
LTI	Linear Time Invariant
LTV	Linear Time Variant
MIMO	Multiple Input Multiple Output
MINO	Multiple Input No Output
NFS	Network File System
PLC	Programmable Logic Controller
SISO	Single Input Single Output
TCP	Tool Center Point
TCP/IP	Transmission Control Protocol/Internet Protocol

1

Introduction

Combining theory and practice is one of the aims of this work. In the thesis a commercial control system has been used in order to compare different *Iterative Learning Control (ILC)* strategies in, what could be, a possible industrial application. The stability theory for linear systems using ILC is extensively covered and many design methods for ILC updating schemes are presented.

1.1 Thesis Outline

The thesis is divided into six parts; I. Background, II. The Application, III. Classical ILC, IV. Second Order ILC, V. An Adaptive Approach to ILC, and finally VI. Final Remarks.

Part I consists of three chapters. The first, Chapter 2, gives a general introduction to ILC, including the history of ILC as well as a first example showing the applicability of ILC. Chapters 3 and 4 present the framework used throughout the thesis. Chapter 3 focuses on the formulation of the ILC problem and the general description of ILC updating formulas. Definitions concerning stability of systems using ILC are also included. In the thesis two different formulations of ILC are used, the *tracking formulation* and the *disturbance rejection formulation*, in Chapter 3 the

two formulations are described in detail. It is also shown how the two formulations relate to each other in the case when the controlled system is linear. In the last chapter of Part I a general theory for stability of systems using ILC is presented. This theory is the base for the results in Parts III-V.

The application part, Part II, is more stand-alone than the other parts of the thesis. The main reason for this is that the other parts are all based on Part I while the application part can be read independently from the other. The first of the three chapters in Part II (Chapter 5) gives a background to why the robot application was chosen as the testbed for ILC in the thesis. It also contains a general discussion on the work that has been done in the commercial robot control system in order to make it possible to apply ILC. In Chapter 6 the robot application is described from a general perspective, including robot modeling and control. This chapter gives a review of material well known in the area, see e.g., Spong and Vidyasagar (1989), Spong et al. (1992), and Stadler (1995). Chapter 6 also contains an overview of the ABB industrial robot family. The actual platform used in all the experiments in the thesis is described in Chapter 7. This chapter also includes a description of how the models, used in the ILC design in Parts III – V, are found using data from the system and standard tools for system identification (Ljung, 1995).

In Part III the classical ILC (as it is defined by the author) is discussed. Chapter 8 contains many design strategies for first order ILC and the proposed design methods are also tested on the industrial robot. In Chapter 9 some limitations on ILC are highlighted. The limitations stem from the fact that the controlled variable and the measured variable are not the same in many applications. This implies that although the measured error is minimized using ILC, the true control error is not minimized. A possible solution for the case where ILC is applied to industrial robots is also presented but implementing the proposed solution on the industrial robot is left for future work.

Part IV gives a frequency domain approach to the second order ILC analysis and design. In Chapters 10 and 11 a background and a general theory for stability of second order ILC systems are given. Chapter 12 suggests possible design methods for second order ILC and the methods are also applied to the industrial robot. Part IV is concluded in Chapter 13.

Part V is the most recent work and it is on an adaptive approach to ILC. The inspiration to this approach comes from the optimization based approach to ILC, presented in Part III, and from general estimation theory, see for example Anderson and Moore (1979). The resulting adaptive ILC algorithm is described in Chapter 14 and in Chapter 15 it is applied to the industrial robot. The conclusions from this part are presented in Chapter 16. Finally, in Part VI the work is summarized and some areas are pointed out where further work is needed.

1.2 Contributions

The main contributions of the thesis are found in Part II, Chapter 7, and in Parts IV–V. Some of the results in Part I, Chapter 4, and in Part III are also new. The main contributions are summarized according to:

- The disturbance rejection formulation of ILC in Section 3.2, originally presented in Norrlöf (1998), and the ILC method based on disturbance estimation in Section 3.5 and Section 4.3.
- A general framework for analysis of linear iterative systems in Section 4.1 using ideas from Norrlöf (1998). Section 4.1.4, the comparison of the stability result in time domain with the result in frequency domain.
- The implementations of the functions needed in the commercial robot control system in order to incorporate ILC, presented in Chapter 7.
- The experiments on the industrial robot, described in Chapter 8 for the first order ILC, Chapter 12 for the second order ILC, and finally, Chapter 15 for the adaptive approach to ILC.
- In Chapter 9, to highlight the possible limitations of ILC when the "true" control error is not measurable. In the same chapter a possible solution is presented for the path following problem in the industrial robot case. The solution is based upon additional accelerometer sensors mounted on the wrist of the robot.
- Frequency domain analysis of second order ILC systems in Part IV. The frequency domain behavior described in Chapter 11 and the design schemes resulting in two second order ILC algorithms in Chapter 12.
- Part V, the combination of an optimization based approach to ILC and ideas from signal estimation theory. The filter interpretation of the matrix formulation of the linear iterative systems in Section 8.2.5 and Chapter 14 leading to an efficient implementation of the adaptive ILC algorithm in Section 14.1.6.

1.3 Reading Directions

For readers already familiar with ILC, Chapter 3 and Chapter 4 could be browsed in order to catch up with the notation used in the thesis. Most important are Section 3.1 and Section 3.2 where the ILC formulations used throughout the thesis are presented. In Chapter 4, the stability results for ILC systems are given in Section 4.2 and the theory behind these results is explained in Section 4.1. It is however possible to use the stability results in Section 4.2 without reading Section 4.1.

Part II can be skipped by readers only interested in the ILC theory. To completely understand the application examples presented in Parts III–V it can however be necessary to browse through Section 6.2. Also Chapter 5 is of general interest.

Part III to Part V contain more specialized material on ILC. An application oriented reader finds probably what he/she needs in Part III, Chapter 8. For more advanced applications where the controlled system is time variant and/or iteration variant the adaptive ILC approach in Part V could be worth considering.

For the reader already familiar with first order ILC and first order ILC design Chapter 8 can be skipped. Chapter 9 however should be read since it highlights a problem that is present in many applications of ILC. In Chapter 9 ideas on how to solve this problem are also presented. Part IV and Part V contain material of general interest also for a reader familiar with ILC. Both parts are constructive and results in design methods for ILC updating schemes.

The conclusions and the discussion on possible future work in Part VI is of general interest.

Part I

Background

2

Introducing the Ideas

2.1 Background

Before going into a more technical discussion on ILC we give a short note on the history and a brief overview of possible connections with other areas in the control field.

2.1.1 A brief history

The idea of using an iterative method to compensate for a repetitive error is not new. When letting a machine do the same task repeatedly it is, at least from an engineering point of view, very sound to use knowledge from previous iterations of the same task to try to reduce the error next time the task is performed. The first academic contribution to what today is called ILC appears to be a paper by Uchiyama (1978). Since it was published in Japanese only, the ideas did not become widely spread. What is a bit remarkable is however that an application for a US patent on "Learning control of actuators in control systems" (Garden, 1971) was done already in 1967 and it was accepted as a patent in 1971. The idea in the patent is to store a "command signal" in a computer memory and iteratively update this command signal using the error between the actual response and the desired

response of the actuator. This is clearly an implementation of ILC, although the actual ILC updating equation is not explicitly formulated in the patent.

From an academic perspective it was not until 1984 that ILC started to become an active research area. In 1984 Arimoto et al. (1984a), Casalino and Bartolini (1984), and Craig (1984), independently published papers about a method that iteratively could compensate for model errors and disturbances. The name Iterative Learning Control was first introduced in Arimoto et al. (1984b). In e.g., Arimoto et al. (1984a), the method was simply referred to as a "bettering process".

The development of ILC stems originally from the robotics area where repetitive motions show up naturally in many applications. Examples of contributions where ILC is applied in robotics are Arimoto et al. (1984a), Mita and Kato (1985), Arimoto (1985, 1991), Bondi et al. (1988), Poloni and Ulivi (1991), Burdet et al. (1997), Casalino and Bartolini (1984), Guglielmo and Sadegh (1996), Horowitz et al. (1991), Horowitz (1993), Jiang et al. (1999), and Lange and Hirzinger (1999).

Examples of surveys on ILC are, Horowitz (1993), Moore (1993), Moore (1998a), and Bien and Xu (1998). Moore (1998a) contains a very good overview of the ILC research. A categorization of much of the publications on ILC up to 1998 is also given in the article.

The focus for the ILC research in the late 90's and in the beginning of the 00's is not so easy to establish but it seems that it has moved from being very focused on stability towards also considering design and performance. Examples in this direction are Bien and Xu (1998), Norrlöf (2000a), Lee et al. (2000), and Longman (2000).

2.1.2 ILC in relation to other techniques

The classical formulation of the Iterative Learning Control problem is, given a reference trajectory and a system, find (using an iterative procedure) the input to the system such that the output follows the desired trajectory as well as possible. Clearly, if a description of the system is available, the optimal solution is to invert this description (if possible) and use this to calculate the input that produces the desired output. This is a *one-step* procedure which can be considered as a *feed-forward* control scheme. This approach has been applied successfully in for example robotics control where it is referred to as *inverse dynamics* (Spong and Vidyasagar, 1989).

If the system representation, describing the mapping from input to output, is not completely known, then it is obvious that the inverse dynamics approach will never achieve a perfect tracking. If instead it is assumed that the structure of the system is known while the exact value of one or more of the parameters are unknown, another well known technique can be applied, namely *identification*. Normally identification together with control is referred to as *adaptive control*. The adaptive control approach is very appealing since it will, theoretically, give a good behavior

for all input signals, during all working conditions. It should be noted that this is true as long as the structure of the system description is correct and some conditions on excitation are met.

Iterative Learning Control is an alternative to the inverse dynamics and the adaptive control approaches in the case when, given a particular reference trajectory and a system, the input signal shall be calculated such that the output follows the desired output as well as possible. Using ILC the control signal is found by an iterative procedure. This can be seen as an iterative search procedure which obviously has to converge to give a successful result. Convergence, or stability as it will be referred to in the thesis, has been and still is an important research field for ILC. The research in the area focuses more and more on the transient behavior and the design of ILC schemes that give a desired transient behavior. This means that practical aspects such as convergence speed and robust performance become more and more important.

2.2 An Introductory Example of ILC

The concepts of Iterative Learning Control are best presented in an example. Assume that a reference signal $r(t)$ over a finite time interval $[0, t_f]$ is given and that a system should track this reference trajectory repeatedly with a high accuracy. A typical application where this problem arises is in the control of robot arms, see e.g., Arimoto et al. (1984a), Arimoto (1991), Horowitz et al. (1991), Horowitz (1993), Craig (1988), and Burdet et al. (1997). The robot arm will also be used as an example in the thesis and it is described in more detail in Part II.

Note that also other control strategies apply to the given control problem. For example it is straightforward to apply an adaptive controller (Slotine and Li, 1991) or a tuning scheme for the feedback and feedforward controller, e.g., iterative feedback tuning (IFT) (Gunnarsson et al., 1999; Hjalmarsson et al., 1994).

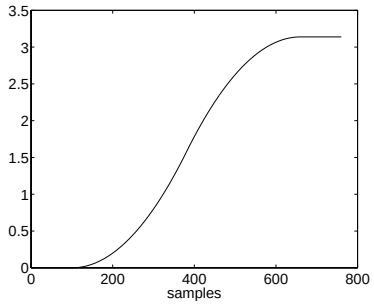
Now consider the reference signal and the system depicted in Figure 2.1. Assume that the reference signal is the position of one joint of a robot. The system G_C can be seen as a discrete time SISO model of the closed loop involving the robot joint and its controller. The joint motion starts in the origin and it is assumed that every time this motion is repeated the system starts from the same initial condition, i.e., the same initial position, initial speed, and so on.

Assume that it exists an initial guess of the input to the system, denoted by $u_0(t)$. Feeding the system with this signal gives,

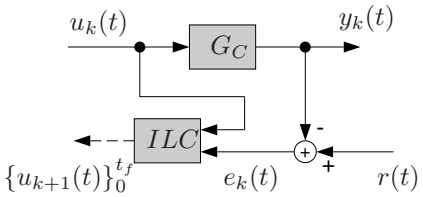
$$y_0(t) = G_C(q)u_0(t) \tag{2.1}$$

where q is the time shift operator. The tracking error is defined as

$$e_0(t) = r(t) - y_0(t) \tag{2.2}$$



(a) The reference signal.



(b) The system using ILC.

Figure 2.1 *An example of a system controlled using ILC.*

i.e., the difference between the desired output and the actual system output.

The index on y, u, and e is called *iteration index* and indicates how many times the iterative motion has been repeated. Note that the first time the motion is performed there have been no repetitions, and accordingly the index is 0.

The idea behind ILC is now introduced. Assume that

1. the initial condition is reset every new iteration and
2. G_C is time invariant.

These assumptions will be made more general in the next chapters.

This leads to the conclusion that the tracking error will be the same if the input signal is kept the same. For simplicity it is assumed that there are no disturbances acting on the system. The ILC algorithm utilizes the assumptions and, by using an iterative search method, it finds the optimal input to the system. Optimal in this particular case means to find the signal $u(t) = G_C^{-1}(q)r(t)$. One possible approach to an ILC updating formula for the control input of the system is given by

$$u_1(t) = u_0(t) + L(q)e_0(t) \tag{2.3}$$

where $L(q)$ is a linear discrete time filter. This updating formula is the most common in the ILC literature, used in e.g., Arimoto et al. (1984a), Togai and Yamano (1985), and Park and Hesketh (1993). The results in (2.2), (2.1), and (2.3) can be generalized according to

$$e_k(t) = r(t) - y_k(t) \tag{2.4a}$$
$$y_k(t) = G_C(q)u_k(t) \tag{2.4b}$$
$$u_{k+1}(t) = u_k(t) + L(q)e_k(t) \tag{2.4c}$$

This is a 2-dimensional problem with a time dimension and an iteration dimension. The latter is introduced since there is a coupling between iterations by the updating of the control signal in (2.4c). In Figure 2.1, the ILC algorithm is shown as a block having u_k and e_k as input. The output is a sequence, $\{u_{k+1}(t)\}_0^{t_f}$, of control actions. The output is chosen like this to stress the fact that the ILC algorithm does not have to be causal, i.e., $u_{k+1}(t)$ can depend on $u_k(\bar{t})$ and $e_k(\bar{t})$ for $\bar{t} > t$.

Using (2.4a), (2.4b), and (2.4c), it is possible to arrive at the following expression for the transformation of the error between the iterations,

$$e_{k+1}(t) = (1 - G_C(q)L(q))e_k(t) \tag{2.5}$$

A sufficient condition for the error to decrease is that

$$|1 - G_C(e^{i\omega})L(e^{i\omega})| < 1 \tag{2.6}$$

for all $\omega \in [-\pi, \pi]$. The sampling time is here assumed to be 1. This has become a standard stability criterion for the ILC updating formula in (2.4c) and it is presented in this form in e.g., Mita and Kato (1985), Poloni and Ulivi (1991), and Gunnarsson and Norrlöf (1997a).

To better understand the stability properties of the method we give a numerical example.

Example 2.1 A Numerical Example

Consider the system

$$G_C(q) = \frac{0.09516q^{-1}}{1 - 0.9048q^{-1}} \tag{2.7}$$

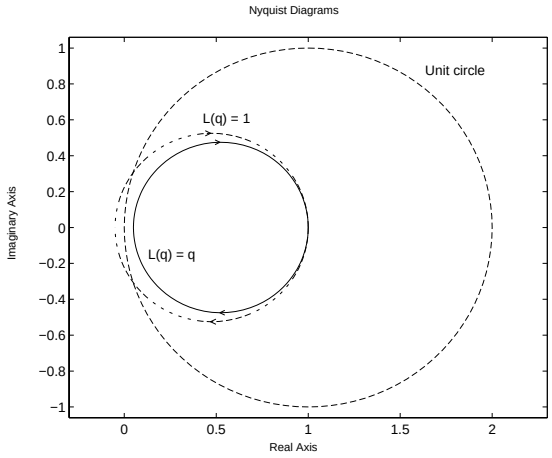
Assume that that it starts from zero initial condition and that the reference signal is given by Figure 2.2(b). Two simulations using the updating formula for the control signal in (2.4c) with two different $L(q)$ are performed where $L(q)$ is chosen according to

$$L(q) = q \tag{2.8}$$

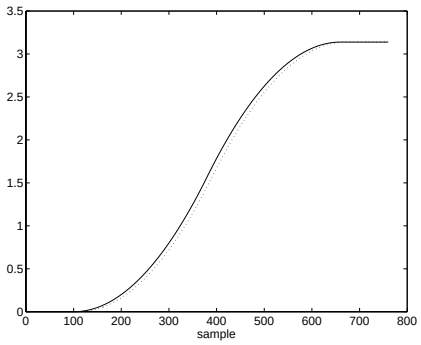
and

$$L(q) = 1 \tag{2.9}$$

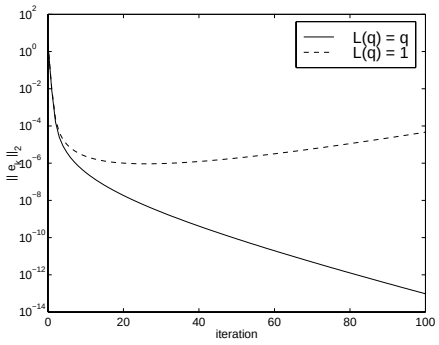
The criterion for stability is given by (2.6). The expression can be interpreted in the following way: *The Nyquist curve for $G_C(q)L(q)$ should be contained in a region in the complex plane given by a unit circle centered at* 1. In Figure 2.2(a) the Nyquist curves are shown for the two choices of $L(q)$ given by (2.8) and (2.9). When $L(q)$ is chosen according to (2.8) the criterion is fulfilled for all frequencies.



(a) A graphical interpretation of the criterion for stability.



(b) The reference signal (solid) and the system output signal $y_0(t)$ (dotted).



(c) $\|e_k(t)\|_2$ with k between 0 and 100 for two different choices of $L(q)$.

Figure 2.2 *Analysis of the stability of the ILC feedback given by Example 2.1 by examining the criterion* (2.6) *and by doing simulations.*

Filter L	Result from execution of `dhfnorm` in MATLAB™
$L(q) = q$	norm between 0.95002 and 0.95097 achieved near 3.1416
$L(q) \equiv 1$	norm between 1.05 and 1.051 achieved near 3.1416

Table 2.1 *Results of ∞-norm calculation.*

Choosing $L(q)$ as in (2.9), however, gives a Nyquist curve that leaves the stability region for high frequencies.

Another method of analyzing the stability is to calculate

$$\sup_{\omega\in[-\pi,\pi]} |1 - G_C(e^{i\omega})L(e^{i\omega})| \tag{2.10}$$

An approximate measure of (2.10) can be found in MATLAB™ using the command `dhfnorm`. Results from this calculation are shown in Table 2.1. Note that using $L(q) = q$ results in a norm less than 1 while using $L(q) = 1$ gives a norm greater than 1. This is the same result as was found using the Nyquist diagram. Yet another way of examining the stability is of course simulations. In Figure 2.2(c) the result of feeding the system with the initial guess of the input, $u_0(t) = r(t)$, is depicted. We can see that there is a small tracking error. By using the ILC method the error can be reduced and in Figure 2.2(c) the results from two simulations using the two different $L(q)$ are shown. The figure shows how the 2-norm or the energy of the error signal develops over time. Clearly, using $L(q) = q$ gives a stable behavior but $L(q) = 1$ results in a growing error after a first phase with rapidly decreasing error. ❑

Note that, in the stable case, a non-causal filter $L(q)$ was used.

Remark 2.1
Although the stability criterion in (2.6) *is only sufficient it is shown in Figure 2.2(c) that in the case when the criterion is not fulfilled in Example 2.1 it leads to a bad behavior. The size of the error is reduced in the first iterations before it starts to grow. The reason for this behavior can be understood by considering the frequency response of* $|1-G_C(e^{i\omega})L(e^{i\omega})|$ *in* (2.5)*. For low frequencies the amplitude is much smaller than one but for high frequencies the error will be attenuated since the amplitude is larger than one, cf. Figure 2.2(a). In Example 2.1 the energy of the error is dominated by the low frequency components in the first iterations. With the iterations this part of the error is however rapidly decreased. After some iterations the high frequency part of the error will instead have grown so that the energy in this part is greater than the low frequency part. This is what can be seen after about 10 iterations in Figure 2.2(c).*

Remark 2.2
From an application point of view the difference between the two suggested ILC algorithms in Example 2.1 might not be so evident since, normally, the learning is switched off after a few iterations. As indicated by Figure 2.2(c) the difference is not so large in the first 5-10 iterations.

In this introductory example of ILC a frequency domain approach for the stability analysis is used. This approach has been applied in more general cases in, e.g., Gunnarsson and Norrlöf (1997b); Horowitz (1994); Mita and Kato (1985); Norrlöf and Gunnarsson (1998), and it will be returned to in the analysis in Chapter 4. A time domain approach to solve the same problem is used in e.g., Arimoto (1985); Arimoto et al. (1984a, 1985). This is also covered in Chapter 4.

The essential ideas and properties of Iterative Learning Control can now be summarized:

- The controlled system repeats the same course of events over and over again. This is typically the case, for instance, in robotics where the same trajectory is repeated every time the same robot program is executed.
- The system starts from the same initial conditions at every iteration.
- Applying ILC to the system leads to a reduced control error, measured in some norm. This is a property that is required of the applied ILC updating equation.

In the following chapters these items will be more thoroughly discussed.

3

Problem Description

In this chapter two formulations of the ILC problem will be discussed. The first, called the *tracking formulation*, is a traditional formulation of the ILC problem, cf. e.g., Moore (1993). The second, the *disturbance rejection* formulation, was introduced in Norrlöf (1998) and it is an alternative way of posing the ILC problem. As will be shown in the thesis, sometimes it will be an advantage to use the first formulation while in other cases the second is to be prefered. Furthermore the ILC updating formula is discussed and the definition of stability of an ILC system is made.

3.1 The Tracking Formulation

Consider the system depicted in Figure 3.1. It is a system with four inputs, the reference signal $r(t)$, an externally generated control signal $u(t)$, and two non-controllable disturbances $w(t)$ and $v(t)$. The measured output is $y(t)$ and the controlled variable is denoted $z(t)$. Note that the system T can have an internal feedback.

The general goal in the tracking formulation is to use the information from the measured output to iteratively update the control signal $u(t)$ such that $z(t)$ tracks

the reference as well as possible. This can be seen as an optimization problem where the optimization problem is solved iteratively, using the true system. The problem of iteratively finding the optimal control input has been discussed in, e.g., Plant (1968) but the tracking problem is not covered there.

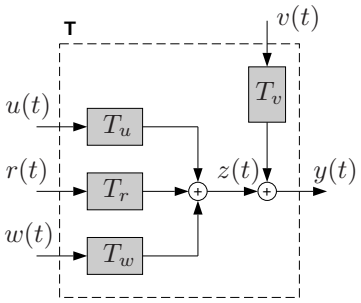


Figure 3.1 *The system representation used in the tracking formulation.*

3.1.1 System description

Time domain representation

The system T that will be used in the thesis is given by

$$z_k(t) = T_r(q)r(t) + T_u(q)u_k(t) + T_w(q)w_k(t) \tag{3.1a}$$
$$y_k(t) = z_k(t) + T_v(q)v_k(t) \tag{3.1b}$$

This is obviously a special case of a more general system description. For example, it might not always be true that a noise corrupted $z_k(t)$ can be measured as in (3.1b). In Chapter 9 some problems concerning this will be discussed.

It is also possible to pose the system in (3.1) in a matrix form,

$$\boldsymbol{z}_k = \boldsymbol{T}_{r,k}\boldsymbol{r} + \boldsymbol{T}_{u,k}\boldsymbol{u}_k + \boldsymbol{T}_{w,k}\boldsymbol{w}_k \tag{3.2a}$$
$$\boldsymbol{y}_k = \boldsymbol{z}_k + \boldsymbol{T}_{v,k}\boldsymbol{v}_k \tag{3.2b}$$

with

$$\boldsymbol{z}_k = \big(z_k(0), \dots, z_k(n-1)\big)^T \tag{3.3}$$

and $\boldsymbol{r}$, $\boldsymbol{u}_k$, $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ defined accordingly. Compared to (3.1a) this is a more general description since it captures both time variant as well as iteration variant systems.

A linear time invariant and causal system, $T_r(q)$, is in matrix form described by a Toeplitz matrix

$$\boldsymbol{T}_r = \begin{pmatrix} g_{T_r}(0) & 0 & \dots & 0 \\ g_{T_r}(1) & g_{T_r}(0) & & \vdots \\ \vdots & & \ddots & 0 \\ g_{T_r}(n-1) & g_{T_r}(n-2) & \dots & g_{T_r}(0) \end{pmatrix} \tag{3.4}$$

where $g_{T_r}(t)$, $t \in [0, n-1]$ are the impulse coefficients of T_r, the sampling time is assumed to be 1, and n is the number of samples. The impulse coefficients represent the response of the system to an impulse input. In (3.4) it is also assumed that the system description is not k-dependent. If the system is linear time variant, the matrix $\boldsymbol{T}_r$ does not become a lower triangular Toeplitz matrix but instead a general lower triangular matrix. The matrices $\boldsymbol{T}_u$, $\boldsymbol{T}_w$, and $\boldsymbol{T}_v$ are given in the same way. The symbols describing the vectors and matrices in the matrix description are throughout the thesis given in bold face to make it easier to distinguish between the representation in (3.1) and the matrix description.

Note that the disturbances have been divided into two classes in the system descriptions in (3.1) and (3.2). The first disturbance, $w_k(t)$, is a system disturbance, which acts as a disturbance on the controlled variable. This disturbance should be compensated for by the ILC algorithm. The other disturbance, $v_k(t)$, acts on the measurements only and this disturbance should, if possible, be neglected by the ILC algorithm.

Frequency domain representation

For the frequency domain analysis the system is assumed to be linear time invariant as in (3.1) with $T_r(q)$, $T_u(q)$, $T_w(q)$, and $T_v(q)$ stable. The corresponding frequency domain representation is found using the Fourier transform,

$$T_r(e^{i\omega}) = \sum_{l=0}^{\infty} g_{T_r}(l) e^{-i\omega l} \tag{3.5}$$

where $g_{T_r}(l)$ are the impulse coefficients for the system $T_r(q)$. Given $T_r(q)$, the frequency domain representation can also be found by simply replacing q with $e^{i\omega}$ in $T_r(q)$. The frequency domain representations of $T_u(q)$, $T_w(q)$, and $T_v(q)$ are found in the same way.

The signals $z_k(t)$, $r(t)$, $u_k(t)$, $w_k(t)$, $y_k(t)$, and $v_k(t)$ are transformed to the frequency domain using

$$X(\omega) = \sum_{l=0}^{\infty} x(l) e^{-i\omega l} \tag{3.6}$$

and it is assumed that this sum exists and is finite for all ω.

This gives the resulting frequency domain representation,

$$Z_k(\omega) = T_r(e^{i\omega})R(\omega) + T_u(e^{i\omega})U_k(\omega) + T_w(e^{i\omega})W_k(\omega) \tag{3.7a}$$
$$Y_k(\omega) = Z_k(\omega) + T_v(e^{i\omega})V_k(\omega) \tag{3.7b}$$

For iterative systems this is an approximation since in the calculations of the Fourier transform it is assumed that the time horizon is infinite. For ILC systems this is not the case. Instead, the tracking performance is considered over a finite time horizon. When using the frequency domain representation to analyze ILC systems the fact that it is an approximation has to be taken into account.

3.1.2 ILC formulation

Now the tracking formulation of ILC is defined.

Definition 3.1 (The tracking formulation of ILC)
Assume that a system T according to Figure 3.1 is given and that it can be described by one of the system descriptions given in (3.1) *and* (3.2).

The tracking formulation of ILC *uses an iterative updating formula for the control signal, according to*

$$\boldsymbol{u}_{k+1} = h(\boldsymbol{r}, \boldsymbol{y}_k, \dots, \boldsymbol{y}_{k-j}, \boldsymbol{u}_k, \boldsymbol{u}_{k-1}, \dots, \boldsymbol{u}_{k-l}) \quad j, l \geq 0 \tag{3.8}$$

to make the input $\boldsymbol{u}_k$ and the controlled variable $\boldsymbol{z}_k$ converge to the minimum of some criterion function

$$V(\boldsymbol{r}, \boldsymbol{z}_k, \boldsymbol{u}_k) \geq 0 \tag{3.9}$$

when $k \to \infty$. Additional constraints can be applied to the control signal, $u_k(t)$.

Remark 3.1
Note that in many of the presented ILC schemes, both in the literature as well as in the thesis, the criterion that should be minimized is never explicitly stated.

Now an example of a case where the tracking formulation of ILC applies.

Example 3.1

An example of a realization of the system T from Figure 3.1 is shown in Figure 3.2. In this case it contains both a feedback controller F and a feedforward controller F_f,

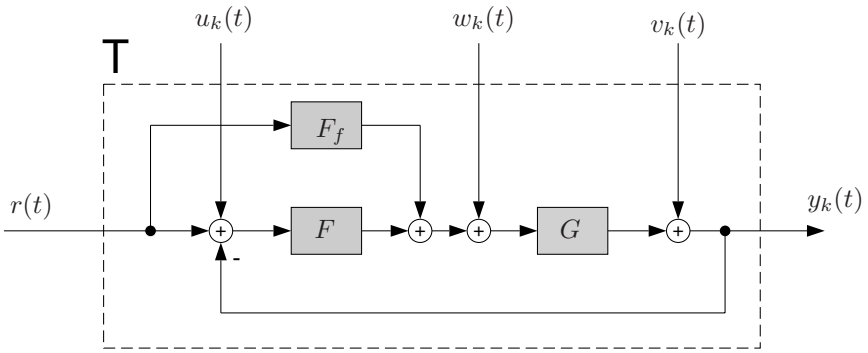


Figure 3.2 *An example of a realization of the system T.*

controlling the system G. Given a reference signal and an ILC updating formula, like (2.4c), the tracking formulation of ILC applies with,

$$T_r(q) = \frac{(F(q) + F_f(q))G(q)}{1 + F(q)G(q)}$$
$$T_u(q) = \frac{F(q)G(q)}{1 + F(q)G(q)}$$
$$T_w(q) = \frac{G(q)}{1 + F(q)G(q)}$$
$$T_v(q) = \frac{1}{1 + F(q)G(q)}$$

Note that a minimization criterion is not explicitly stated in this case, cf. Remark 3.1. ❑

Definition 3.1 is important in order to understand exactly what an ILC algorithm formulated according to the tracking formulation actually does. In fact, most of the ILC schemes found in the literature so far have the structure of the tracking formulation. It is possible to add also $\boldsymbol{y}_{k+1}$ as an argument to the mapping in (3.8). This generalization makes it possible to include in this formulation also some approaches which adapt the feedback controller online, e.g., Owens and Munde (1997), or that use measurements not only from the previous iteration but also from the current iteration, e.g., Chen et al. (1996a,b). The latter approach is also called *Current Iteration Tracking Error* approach (CITE). It is important to stress that the feedback of the current iteration tracking error in ILC is nothing else than what is called feedback in traditional control theory.

Before giving a general background to the ILC updating equations we shall describe the disturbance rejection formulation of ILC.

3.2 The Disturbance Rejection Formulation

In the previous section it was shown how ILC can be used to solve a tracking problem. Another case where ILC can be applied is when an unknown but partly repetitive disturbance is acting on a system. This will here be referred to as the disturbance rejection formulation of ILC.

3.2.1 System description

The goal in ILC is usually to, iteratively, find the input to a system such that some error is minimized. In the disturbance rejection formulation, the goal is to find an input $u_k(t)$ such that the output $z_k(t)$ is minimized while possibly having some restriction on the control signal. If the system is known and invertible, and the disturbance $d_k(t)$ is known, the obvious approach would be to filter $d_k(t)$ through the inverse of the system and then use the resulting $u_k(t)$ as a control input. This means that the optimal input can be expressed as

$$u_k(t) = -(G^0(q))^{-1} d_k(t)$$

if it is assumed that the system is discrete time linear time invariant. More generally the system description is considered to be

$$\boldsymbol{z}_k = \boldsymbol{G}^0 \boldsymbol{u}_k + \boldsymbol{d}_k \tag{3.10a}$$

$$\boldsymbol{y}_k = \boldsymbol{z}_k + \boldsymbol{n}_k \tag{3.10b}$$

where, as described in Section 3.1, bold face letters are used to make it easier to separate the scalar description, e.g., $y_k(t) = z_k(t) + n_k(t)$, from the vector description. Assume that the disturbance d_k can be described as

$$\boldsymbol{d}_{k+1} = \boldsymbol{d}_k + \boldsymbol{\Delta}_{d_k} \tag{3.11}$$

The vectors $\boldsymbol{z}_k, \boldsymbol{u}_k, \boldsymbol{d}_k, \boldsymbol{y}_k, \boldsymbol{n}_k, \boldsymbol{\Delta}_{d_k} \in \mathbb{R}^n$ are defined according to (3.3) and the matrices $\boldsymbol{G}^0 \in \mathbb{R}^{n \times n}$ in the same way as in Section 3.1.1.

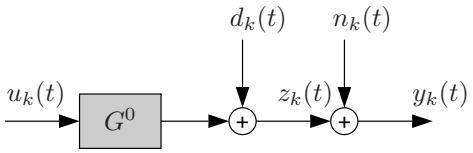


Figure 3.3 *The system considered in the disturbance rejection approach.*

3.2.2 ILC formulation

Using the system description from the previous section, the general disturbance rejection formulation of ILC can now be defined.

Definition 3.2 (The Disturbance Rejection formulation of ILC)
Suppose that there is a system G^0, as in Figure 3.3 that can be described by (3.10) *with a disturbance given by* (3.11)*, acting on the output.*

The disturbance rejection formulation of ILC *uses an iterative updating formula for the control signal, according to*

$$\boldsymbol{u}_{k+1} = h(\boldsymbol{y}_k, \ldots, \boldsymbol{y}_{k-j}, \boldsymbol{u}_k, \boldsymbol{u}_{k-1}, \ldots, \boldsymbol{u}_{k-l}) \quad j, l \geq 0 \tag{3.12}$$

to make the control signal, $u_k(t)$, and the output signal $z_k(t)$ converge such that some criterion function

$$V(\boldsymbol{z}_k, \boldsymbol{u}_k) \geq 0$$

is minimized when $k \to \infty$. Additional constraints can be applied to the control signal, $u_k(t)$.

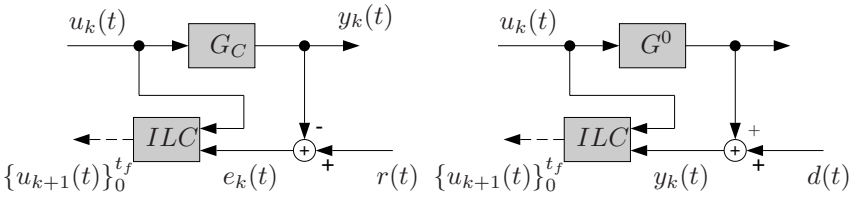
In the next section the two formulations of ILC are compared in the case when the system to which ILC is applied is linear.

3.3 Comparison of the Two ILC Formulations

The tracking formulation is a well established method of formulating the ILC problem and therefore it can be of general interest to see how it relates to the disturbance rejection formulation.

First consider the introductory example of ILC in Section 2.2. This example is formulated according to the tracking formulation but it can easily be transformed into the disturbance rejection formulation and in Figure 3.4 this is shown. In the disturbance rejection formulations the system G_C is replaced with $G^0 = -G_C$, the output is redefined as $y_k(t) = e_k(t)$. The goal in the disturbance rejection formulation in Figure 3.4(b) becomes to find a control signal $u_k(t)$ such that the disturbance $d(t) = r(t)$ is completely compensated for.

It is also possible to reformulate the general tracking formulation in (3.1) as a disturbance rejection problem. In the scalar case, let $G^0(q) = T_u(q)$, $d_k(t) = (T_r(q) - 1)r(t) + T_w(q)w_k(t)$, and $n_k(t) = T_v v_k(t)$, to apply the disturbance rejection formulation of ILC. In this case the reference, $r(t)$, is treated a bit different compared to Figure 3.4(b) but for the ILC algorithm the goal is still to minimize $z_k(t)$ and compensate for the disturbance $d_k(t)$.



(a) The ILC example from Section 2.2.

(b) Corresponding disturbance rejection formulation.

Figure 3.4 *The example from Figure 2.1 and its corresponding disturbance rejection formulation.*

To transform the disturbance rejection formulation into a tracking formulation in general, just let $T_u(q) = G^0(q)$, $T_w(q) = 1$, $w_k(t) = d_k(t)$, $T_v(q) = 1$, $v_k(t) = n_k(t)$, and $r(t) = 0$. The goal for the ILC algorithm then becomes to track a zero reference signal.

3.4 The ILC Updating Formula

In this section some different approaches to updating the signal $u_k(t)$ in the ILC algorithms will be discussed. The ILC updating formulas can be categorized in two groups according to how the information from previous iterations is utilized. The two groups are now defined formally.

Definition 3.3 (First order ILC)
An ILC updating formula that only uses measurements from the previous iteration is called a first order ILC.

An example of a first order ILC algorithm is the one used in Example 2.1.

Definition 3.4 (High order ILC)
When the ILC updating formula uses measurements from more than the previous iteration it is called a high order ILC. *The term* second order, third order, *etc., is used when the order of the ILC algorithm should be emphasized.*

An ILC algorithm is not only characterized by whether it is a *first order* or a *high order* ILC algorithm. In the next two sections the ILC updating formula will be divided into the linear case and the nonlinear case.

3.4.1 Linear ILC

The class of linear ILC updating formulas are naturally divided into first order and higher order ILC algorithms.

First order ILC

Several ILC algorithms and ILC algorithm structures have been suggested in literature. Common for most of the suggested algorithms, e.g., Arimoto et al. (1984a), Arimoto (1985), Mita and Kato (1985), Hara et al. (1988), Tomizuka et al. (1989), Moore (1993), Bien and Xu (1998), Craig (1988), Horowitz (1993), and de Roover (1996a), is that the structure is given by

$$u_{k+1}(t) = Q(q)(u_k(t) + L(q)e_k(t)) \tag{3.13}$$

where $Q(q)$ and $L(q)$ are considered to be linear transfer operators or simply discrete time filters. In many of the contributions, the ILC is considered to be implemented in continuous time. This is, for example, the case in Arimoto et al. (1984a). Equation (3.13) is easily adjusted to this case: Just change the delay operator q to the derivative operator p. The general linear first order ILC in (3.13) has been presented stepwise with early contributions by Togai and Yamano (1985), and Mita and Kato (1985). The use of the Q filter is suggested in Hara et al. (1988) and Tomizuka et al. (1989). In many of the references, the Q-filter is chosen as a constant equal to 1.

An even more general form of the first order ILC updating formula, is given by the following generalization of (3.13),

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}_k(\boldsymbol{u}_k + \boldsymbol{L}_k\boldsymbol{e}_k) \tag{3.14}$$

Here, the matrices $\boldsymbol{Q}_k$ and $\boldsymbol{L}_k$ can be realizations (according to (3.4)) of iteration as well as time variable filters.

High order ILC

Although most contributions on ILC have been on the first order case, the idea of utilizing the measurements from more than the previous iteration has been covered in many articles. In Liang and Looze (1993) two dimensional transforms are used to analyze the behavior of the system in both the time and the iteration directions. In the paper by Arimoto (1991) the errors from previous iterations are used in an indirect way. Chen et al. have also investigated the use of high order ILC and the main reference is Chen et al. (1998), but the issue is also discussed in Chen et al. (1997a,b). High order ILC has also been covered in e.g., Bien and Huh (1989), Bien and Xu (1998) and Norrlöf and Gunnarsson (1999).

A general linear time invariant high order ILC can be written according to,

$$u_{k+1}(t) = \sum_{j=k-N+1}^{k} \Big(Q_{k-j+1}(q)\big(u_j(t) + L_{k-j+1}(q)e_j(t)\big)\Big) \tag{3.15}$$

where $Q_j(q)$ and $L_j(q)$, $j = 1, \ldots, N$ represent linear transfer operators. The control signal for the next trial is calculated from the control signals and the errors in the N previous iterations. This represents an Nth order ILC algorithm.

To understand the concept of high order ILC algorithms, assume that the filters L_j are constants, i.e.,

$$L_j(q) = l_j, \quad j = 1, \ldots, N \tag{3.16}$$

and $Q_1 = 1$, $Q_j = 0$ for $j > 1$. From (3.15) it follows that,

$$u_{k+1}(t) = u_k(t) + \sum_{j=k-N+1}^{k} l_{k-j+1}e_j(t) \tag{3.17}$$

The control signal at time t in iteration $k+1$ is equal to the previous control signal plus a weighted sum of the errors at time t in the N previous iterations. This can be interpreted as filtering of the error in the iteration direction with the filter impulse coefficients l_j.

The time shift operator q is well known, but here an iteration shift operator will be introduced. This operator, denoted q_k is defined as,

$$q_k u_k(t) = u_{k+1}(t) \tag{3.18}$$

For a first order ILC

$$u_{k+1}(t) = u_k(t) + \gamma e_k(t) \tag{3.19}$$

the corresponding filter interpretation

$$u_k(t) = P(q_k)e_k(t)$$

becomes

$$P(q_k) = \frac{\gamma}{q_k - 1} \tag{3.20}$$

i.e., a pure integrator. By introducing also e_{k-1} in the updating equation a *second-order* ILC is obtained,

$$u_{k+1}(t) = u_k(t) + \gamma_1 e_k(t) + \gamma_2 e_{k-1}(t) \tag{3.21}$$

This kind of ILC algorithm is often referred to as a *two-step algorithm*. The filter corresponding to (3.21) is

$$P(q_k) = \frac{\gamma_1 q_k + \gamma_2}{q_k(q_k - 1)} \tag{3.22}$$

which, in the interpretation of a controller, is of PI-type.

In Carlsson (2000) a kind of high order ILC is suggested that has the following structure: The filters Q_j, $1 \leq j \leq N$, are chosen as

$$Q_j(q) = \begin{cases} 1 & j = 1 \\ 0 & j > 1 \end{cases}$$

and the filters $L_{k,j}(q)$ are chosen as

$$L_{k,j}(q) = \gamma(k)\frac{L(q)}{N}$$

with $L(q)$ designed using a design method for first order ILC algorithms. The function $\gamma(k)$ is defined as

$$\gamma(k) = \begin{cases} 1 & \text{if } k = N \cdot l \text{ with } l \in \mathbb{Z}^+ \\ 0 & \text{otherwise} \end{cases}$$

With this method it is possible to run a batch of iterations and use the average of the errors in the updating equation. This can be used as a way to reduce the effect of measurement disturbances in the ILC algorithm. Other possible approaches for doing this will be presented in Section 3.5 and in Part V.

General linear ILC

Inspired by the results from the previous section it is possible to introduce a general linear ILC updating formula. As seen before, the updating formula can be interpreted as a filtering of the error and the control signal in both the t- and the k-direction. This can be formalized by introducing a 2-dimensional filter in the ILC updating formula. The ILC algorithm is in this formalism given by,

$$u_k(t) = P(q_k, q)e_k(t), \;\; k = 0, 1, 2, \ldots \quad t \in [0, t_f] \tag{3.23}$$

with q_k according to (3.18) and where $P(q_k, q)$ is a rational function

$$P(q_k, q) = \frac{P_B(q_k, q)}{P_A(q_k, q)} \tag{3.24}$$

The functions $P_A(\cdot, \cdot)$ and $P_B(\cdot, \cdot)$ are both polynomial functions in q_k and q. The polynomials P_A and P_B can be written as polynomials in q_k with coefficients being polynomials in q. This is shown in an example.

Example 3.2

The ILC,

$$u_{k+1}(t) = u_k(t) + L_1(q)e_k(t) + L_2(q)e_{k-1}(t)$$

can be expressed using the representation with q_k and q_t as

$$u_{k+1}(t) = \frac{L_1(q)q_k + L_2(q)}{q_k - 1}e_k(t) = \frac{P_B(q_k, q)}{P_A(q_k, q)}e_k(t)$$

where P_B and P_A are both polynomials in q_k, with coefficients that are polynomials in q. ❑

Next nonlinear ILC will be discussed briefly.

3.4.2 Nonlinear ILC

Most of the work in the area of ILC has been done on linear ILC updating formulas. In the tracking formulation of ILC presented previously in this chapter the description of the ILC updating equation was,

$$\boldsymbol{u}_{k+1} = h(\boldsymbol{r}, \boldsymbol{y}_k, \boldsymbol{y}_{k-1}, \dots, \boldsymbol{y}_{k-j}, \boldsymbol{u}_k, \boldsymbol{u}_{k-1}, \dots, \boldsymbol{u}_{k-l})$$

This mapping can be a general mapping from the reference signal, the previous measurements, and the previous control signals. In this very general framework not so many results are available. There are, however, some results in the survey on ILC by Moore (1993) and in a recent book edited by Bien and Xu (1998). Moore has devoted a chapter to a discussion on the use of Artificial Neural Networks (ANN) in ILC. This can be seen as a kind of nonlinear black-box identification approach and in this context not only the control signal changes over the iterations but also the ILC algorithm. For a more thorough discussion of this kind of ILC approach see Moore (1993), Bien and Xu (1998) and the references found there.

Another possible approach that leads to an overall nonlinear ILC is to combine a system identification and model based design procedure for the ILC algorithm. This is discussed in Norrlöf (2000a) and in Part V.

3.5 ILC Using Disturbance Estimation

After the general discussion on ILC updating formulas in the previous section an example is now given on of how an ILC algorithm can be found in a systematic way. This approach can be applied to both formulations of ILC but here the disturbance rejection formulation will be used.

If the system G^0 in (3.10) is a discrete time linear time invariant system, then the following equations give a mathematical description of the behavior of the system,

$$\begin{aligned} z_k(t) &= G^0(q)u_k(t) + d(t) \\ y_k(t) &= z_k(t) + n_k(t) \end{aligned} \tag{3.25}$$

For simplicity it is assumed that the system disturbance $d(t)$ is k-independent.

Now, assume $G(q)$ to be a model of $G^0(q)$. Using the model of the system the disturbance $d(t)$ can be estimated using the measurement from the system and the model,

$$\tilde{y}_k(t) = y_k(t) - G(q)u_k(t) \tag{3.26}$$

Let $\hat{d}_k(t)$ be the estimate of the disturbance in the kth iteration. A straightforward approach to estimate the disturbance is to minimize the loss function

$$V_{k,t}(\hat{d}_k(t)) = \frac{1}{2}\sum_{j=0}^{k-1}(\tilde{y}_j(t) - \hat{d}_k(t))^2 \tag{3.27}$$

and the corresponding estimate is given by,

$$\hat{d}_k(t) = \frac{1}{k}\sum_{j=0}^{k-1}\tilde{y}_j(t) \tag{3.28}$$

This can also be written in a recursive form as,

$$\hat{d}_{k+1}(t) = \frac{k}{k+1}\hat{d}_k(t) + \frac{1}{k+1}\tilde{y}_k(t) \tag{3.29}$$

The corresponding ILC algorithm is an updating equation for the control signal $u_k(t)$. In order to minimize $y_k(t)$ the best choice for the input is $u_{k+1}(t) = -\frac{1}{G(q)}\hat{d}_{k+1}(t)$ which means that,

$$u_{k+1}(t) = u_k(t) - \frac{1}{(k+1)G(q)}y_k(t) \tag{3.30}$$

Note the similarity with the standard first order ILC updating equation,

$$u_{k+1}(t) = Q(q)(u_k(t) + L(q)e_k(t))$$

where $e_k(t)$ is the error. In the disturbance rejection approach this is simply the output $y_k(t)$. In (3.30) the Q-filter is chosen as $Q \equiv 1$ and the L-filter is an iteration dependent filter since the gain is reduced every iteration. This means that $L_k(q) = -\frac{1}{(k+1)G(q)}$ which, normally, is a non-causal filter. Since $e_k(t)$, $0 \leq t \leq n-1$, is available when calculating $u_{k+1}(t)$ this is not a problem.

By just observing the output in the first iteration it is possible to find an estimate of d. Since there is a measurement disturbance the estimate can however be improved and this is what the algorithm iteratively will do. Note that since the gain of the L_k-filter is reduced with k the algorithm will not work very well if $d(t)$ is varying as a function of iteration. The gain of the L-filter will actually tend to zero when $k \to \infty$. An ILC algorithm that can work well also with a iteration varying $d(t)$ is presented in Part V. The analysis of the proposed algorithm is done in the next chapter.

3.6 Stability

Stability is a general and very important property for a control system. For systems using ILC questions related to stability and convergence are typically:

- Will the error approach zero as the number of iterations grow?
- If the error does not approach zero, is it bounded?

Before introducing the definitions on stability some underlying assumptions on the controlled system are given.

3.6.1 Postulates

In order for the stability to be well defined there are some natural postulates that have to be introduced about the system that is controlled using ILC. The postulates are originally formulated by Arimoto and can be found in, e.g., Spong et al. (1992). They are reformulated such that they fit the framework used in the thesis.

(P1) Every iteration ends in a fixed time of duration t_f.

(P2) A desired output $r(t)$ is given a priori over $t \in [0, t_f]$.

(P3) Repetition of the initial setting is satisfied. This means that if the controlled system is written in state space form the initial state $x_k(0)$ fulfills,

$$x_k(0) = x_0(0), \quad k \in \mathbb{Z}^+$$

(P4) Invariance of the system dynamics is ensured throughout the iterations.

(P5) Every output $z_k(t)$ can be measured and therefore the tracking error signal, $\epsilon_k(t) = r(t) - z_k(t)$, can be utilized in the calculation of $u_{k+1}(t)$.

(P6) Given a reference trajectory $r(t)$, $t \in [0, t_f]$, with a piecewise continuous derivative, it exists a unique input trajectory $u_d(t)$ on the same time interval such that $z(t)$ equals $r(t)$.

The postulates (P1) and (P2) are very basic and are also included in the description of the two ILC formulations in Section 3.1 and Section 3.2. (P3) is a bit restrictive and it is possible to formulate a more practical postulate:

(P3a) The system is initialized at the beginning of every iteration such that the error in the initial state is limited. This means that if the controlled system is written in state space form the initial state $x_k(0)$ fulfills,

$$|x_k(0) - x_0(0)| < \epsilon \quad k \in \mathbb{Z}^+$$

for some constant ϵ.

This is also found in Spong et al. (1992). As indicated in the system descriptions in (3.2) and (3.10) (P5) is not realistic and it could instead be formulated as:

(P5a) A noise corrupted output, $y_k(t) = z_k(t) + n_k(t)$, is possible to measure where $n_k(t)$ is a measurement disturbance.

In Chapter 9 a case is discussed where it is not possible to measure the noise corrupted controlled variable as indicated above.

The last postulate, (P6), is important since it says that it actually exists an input that makes it possible to track the desired reference signal.

3.6.2 Definitions

To describe the notion of stability for systems using ILC it is necessary to give some definitions. First the definition of ϵ-convergence for a system using ILC is considered. It is assumed that it exists an input u_d that, according to postulate (P6), gives exactly the desired output.

Definition 3.5 (ϵ-convergence)
A system using ILC is ϵ-convergent *in a norm* $\|\cdot\|$ *if*

$$\limsup_{k\to\infty} \|u_d - u_k\| < \epsilon$$

A system using ILC is called *stable* if it is ϵ-convergent with $\epsilon < \infty$. Note that stability does not imply that the error when $k \to \infty$ is smaller than what is achieved without using ILC.

An important difference between Definition 3.5 and most of the previous stability definitions found in the literature, see e.g., Hideg (1992), is that there is no explicit assumption that the ILC system will convergence to zero error.

Now it is possible to continue and define asymptotic stability, exponential stability and global stability.

Definition 3.6 (Asymptotic stability)
A system using ILC is asymptotically stable *if,*

$$\limsup_{k\to\infty} \|u_d - u_k\| = 0$$

Definition 3.7 (Exponential Stability)
A system using ILC is exponentially stable *if*

$$\exists \alpha, \beta > 0 \implies \forall k > 0, \quad \|u_d - u_k\| \leq \alpha \|u_d - u_0\| e^{-\lambda k}$$

If the system controlled by the ILC algorithm is non-linear it might happen that not all initial trajectories (z_0 or u_0) will be stable. If this is still the case, i.e., all initial trajectories lead to stability, then the word *global* can be added in the definitions above. For linear time invariant systems, stability always implies global stability.

4

Analysis

In this chapter, the stability of systems using ILC is analyzed for the two formulations presented in the previous chapter. Before going into the analysis of the stability and the performance, however, a system theoretic background to a class of systems called *linear iterative systems* is given. This theory will be used in the analysis of the ILC systems later in the chapter.

4.1 Linear Iterative Systems

First we give a motivation to why the linear iterative systems are introduced for the analysis of ILC systems.

4.1.1 Motivation

In Norrlöf (1998) a class of systems called *linear iterative systems* is introduced. A special case is the linear time invariant iterative system that can be described as

$$z_{k+1}(t) = F(q)z_k(t) + F_r(q)r(t) \tag{4.1}$$

where $z_k(t) \in \mathbb{R}^N$, $F(q)$ and $F_r(q)$ are matrices of discrete time transfer operators, and $r(t)$ is a scalar. What characterizes an iterative system is that the time, t, is discrete and assumed to be limited to a bounded interval.

As a motivation for the introduction of the system description in (4.1), Example 2.1 is used. From (2.4) it follows that

$$\begin{aligned} e_k(t) &= r(t) - y_k(t) \\ y_k(t) &= G_C(q)u_k(t) \\ u_{k+1}(t) &= u_k(t) + L(q)e_k(t) \end{aligned}$$

and the last equation can be written as

$$u_{k+1}(t) = (1 - L(q)G_C(q))u_k(t) + L(q)r(t) \tag{4.2}$$

by simply using the definitions of e_k and y_k. With $z_k(t) = u_k(t)$, $F(q) = (1 - L(q)G_C(q))$ and $F_r(q) = L(q)$ it is clear that (4.2) is a special case of (4.1) and this also motivates why a general theory for linear iterative systems can be useful for the analysis of ILC systems.

The important stability property of the linear iterative system in (4.1) is bounded-input, bounded-output (BIBO) stability which is defined as follows.

Definition 4.1 (BIBO stability)
A linear iterative system is BIBO stable if a bounded input, $\|\boldsymbol{r}\| < \infty$, generates a bounded output, $\|\boldsymbol{z}_k\| < \infty$, for all k.

The input $\boldsymbol{r}$ and the output $\boldsymbol{z}_k$ are defined as

$$\boldsymbol{r} = \begin{pmatrix} r(0) & r(1) & \dots & r(n-1) \end{pmatrix}^T \tag{4.3}$$

and

$$\boldsymbol{z}_k = \begin{pmatrix} z_k^T(0) & z_k^T(1) & \dots & z_k^T(n-1) \end{pmatrix}^T \tag{4.4}$$

respectively (the sampling time is assumed to be 1).

If BIBO stability can be established it is obvious that $\|z_k\|$ will be less than infinity for all k. The BIBO stability for the linear iterative system therefore implies that the corresponding ILC algorithm gives a stable ILC system. For the particular case described by (4.2), BIBO stability of the corresponding linear iterative system implies ϵ-convergence, for some non-specified $\epsilon < \infty$.

The class of systems that can be described by an iterative system is easily extended by letting the system description become,

$$\boldsymbol{z}_{k+1} = \boldsymbol{F}_k \boldsymbol{z}_k + \boldsymbol{F}_{r,k} \boldsymbol{r} \tag{4.5}$$

which includes both time and iteration variant systems and where $\boldsymbol{z}_k$ and $\boldsymbol{r}$ are defined as in (4.4) and (4.3).

The linear iterative systems presented here can be seen as realizations of special cases of *repetitive systems*, covered in Rogers and Owens (1992). In a general repetitive systems it is assumed that the number of samples, n, is k-dependent. Some conference articles, e.g., Amann et al. (1994) and Rogers and Owens (1994), discuss the idea of using 2D-systems theory for the analysis of stability of ILC systems. The approach taken here is however slightly different also compared to the 2-D case since the transfer operator matrices are only one dimensional. They depend on the time delay operator q only. In the 2D approach, the system in (4.1) is instead written in operator form,

$$z(k+1,t) = T(q_k, q)z(k,t) + F_r(q)r(t) \tag{4.6}$$

where $T(q_k, q)$ is assumed to be a rational function in the delay operators, q_k and q. The q operator is the standard time shift operator and q_k is the iteration shift operator defined in (3.18). Considering (4.6), the dimensionality of 2 of the problem becomes obvious. This fact has also been discussed in, e.g., Rogers and Owens (1992) and Galkowski et al. (1997). In their analysis it has been shown that it is possible to apply results from the two dimensional systems theory.

Using the system description in (4.1) and (4.5) as a starting point, the time domain and the frequency domain analysis tools and methods are now discussed.

4.1.2 Time domain

The time domain analysis is divided into two parts. The first part considers *linear iteration invariant* systems while the second part deals with the *linear iteration variant* case. The difference between the two cases is that in the first, the dynamics does not change with k while in the second this might be the case. Most of the results presented here are well known from linear systems theory, see for example Rugh (1996) or Kailath (1980). Two different measures of the size of a matrix are used. The first is the spectral radius which is defined as,

$$\rho(\boldsymbol{F}) = \max_{i=1,\dots,n} |\lambda_i(\boldsymbol{F})| \tag{4.7}$$

where $\lambda_i(\boldsymbol{F})$ is the ith eigenvalue of the matrix $\boldsymbol{F} \in \mathbb{R}^{n\times n}$. The second is the maximum singular value, defined as,

$$\bar{\sigma}(\boldsymbol{F}) = \sqrt{\rho(\boldsymbol{F}^T\boldsymbol{F})} \tag{4.8}$$

The maximum singular value gives a bound of the gain of a matrix by the fact that,

$$\|\boldsymbol{F}\boldsymbol{x}\| \leq \bar{\sigma}(\boldsymbol{F})\|\boldsymbol{x}\|$$

If the maximum singular value is less than one it is clear that the norm of the result decreases every time $\boldsymbol{x}$ is mapped by $\boldsymbol{F}$. This is an important observation that will be used in many of the stability results for linear iterative systems.

Linear iteration invariant case

In the time domain analysis it is common that a matrix description of the system is adopted. Obviously, if it is known that t is limited to an interval $0 \leq t \leq n-1$, then

$$y_k(t) = T_u(q)u_k(t) \tag{4.9}$$

can be written as

$$\boldsymbol{y}_k = \boldsymbol{T}_u \boldsymbol{u}_k \tag{4.10}$$

where $\boldsymbol{y}_k, \boldsymbol{u}_k \in \mathbb{R}^n$, and $\boldsymbol{T}_u \in \mathbb{R}^{n\times n}$. If $T_u(q)$ is a causal system then realizing (4.9) in the matrix form described in (4.10) will lead to a matrix $\boldsymbol{T}_u$ in the same form as in (3.4), i.e., a lower triangular Toeplitz matrix. The matrix description of the system $\boldsymbol{T}_u$ in (4.10) does not cover only linear time invariant systems, as described by (4.9). $\boldsymbol{T}_u$ can easily be used to describe a linear time variant causal system by letting it become a general lower triangular matrix.

For a general linear iterative system in the matrix form,

$$\boldsymbol{z}_{k+1} = \boldsymbol{F}\boldsymbol{z}_k + \boldsymbol{F}_r\boldsymbol{r} \tag{4.11}$$

it is possible to apply the stability results directly from linear systems theory, see e.g., Rugh (1996). Uniform exponential stability, for example, is achieved if the spectral radius, $\rho(\boldsymbol{F})$, is less than one. Uniform (exponential) stability is defined as follows.

Definition 4.2 (Uniform (exponential) stability)
The linear iterative system,

$$\boldsymbol{z}_{k+1} = \boldsymbol{F}_k\boldsymbol{z}_k + \boldsymbol{F}_{r,k}\boldsymbol{r}$$

is uniformly stable if, for $r(t) = 0$ and any $\boldsymbol{z}_0$, k_0,

$$\|\boldsymbol{z}_k\| \leq \gamma\|\boldsymbol{z}_0\|,\ k \geq k_0$$

where γ is a positive constant. Uniform exponential stability is achieved if, in addition to the conditions above,

$$\|\boldsymbol{z}_k\| \leq \gamma\lambda^{k-k_0}\|\boldsymbol{z}_0\|,\ k \geq k_0$$

for $0 < \lambda \leq 1$.

The following theorem gives the condition for bounded-input, bounded-output stability, see for example a text book on linear systems theory e.g., Rugh (1996).

Theorem 4.1 (BIBO stability)
If $\rho(\boldsymbol{F}) < 1$ in the linear iterative system,

$$\boldsymbol{z}_{k+1} = \boldsymbol{F}\boldsymbol{z}_k + \boldsymbol{F}_r\boldsymbol{r}$$

then the system is bounded-input, bounded-output stable.

BIBO stability is explained in Definition 4.1.

If the system described by $\boldsymbol{F}$ in (4.11) is causal linear time invariant and having relative degree 0, then a necessary and sufficient condition for bounded-input, bounded-output stability is that the first Markov parameter is less than 1, i.e., that the diagonal elements of $\boldsymbol{F}$ are less than 1. This result has been discussed, e.g., in Moore (1998b).

A useful result can be formulated as a corollary based on Theorem 4.1.

Corollary 4.1
When the conditions in Theorem 4.1 are satisfied the system described by (4.11) *will converge according to*

$$\boldsymbol{z}_\infty = (I - \boldsymbol{F})^{-1}\boldsymbol{F}_r\boldsymbol{r}$$

Proof If the stability criterion is fulfilled the signal $\boldsymbol{z}_k$ will converge and by letting $k \to \infty$ in (4.11) the result follows. ■

Although the result in Theorem 4.1 is necessary and sufficient for BIBO stability of the linear iterative system in (4.11) it might not be enough from a transient response point of view. The following well known result guarantees a monotonously decreasing 2-norm when considering the homogenous part of (4.11).

Theorem 4.2
If the maximum singular value fulfills $\bar{\sigma}(\boldsymbol{F}) < 1$ in

$$\boldsymbol{z}_{k+1} = \boldsymbol{F}\boldsymbol{z}_k$$

then $|\boldsymbol{z}_{k+1}| < |\boldsymbol{z}_k|$.

Finally, one remark about the description in (4.11): It is possible to formulate and analyze higher order linear iterative systems in the framework presented above.

Assume for example that

$$\boldsymbol{\zeta}_{k+1} = \boldsymbol{F}_1\boldsymbol{\zeta}_k + \boldsymbol{F}_2\boldsymbol{\zeta}_{k-1} + \boldsymbol{r} \tag{4.12}$$

where $\boldsymbol{\zeta}_k, \boldsymbol{r} \in \mathbb{R}^n$ and $\boldsymbol{F}_1, \boldsymbol{F}_2 \in \mathbb{R}^{n\times n}$. This system can be written according to

$$\boldsymbol{z}_k = \begin{bmatrix} \boldsymbol{\zeta}_k \\ \boldsymbol{\zeta}_{k-1} \end{bmatrix}, \quad \boldsymbol{F} = \begin{bmatrix} \boldsymbol{F}_1 & \boldsymbol{F}_2 \\ I & 0 \end{bmatrix}, \text{and} \quad \boldsymbol{F}_r = \begin{bmatrix} I \\ 0 \end{bmatrix} \tag{4.13}$$

Using Theorem 4.1 it is obvious that the the resulting linear iterative system is BIBO stable if $\rho(\boldsymbol{F}) < 1$. The fact that also higher order systems can be written in this form has also been explored by, e.g., Rogers and Owens (1992) and Amann et al. (1994).

Linear iteration variant case

As was noted in the previous section the matrix description in (4.10) is more general than the description in (4.9). By letting the matrix $\boldsymbol{T}_u$ be a general lower triangular matrix the corresponding description captures a general, causal, linear time variant system. The general linear time and iteration variant iterative system becomes

$$\boldsymbol{z}_{k+1} = \boldsymbol{F}_k \boldsymbol{z}_k + \boldsymbol{F}_r \boldsymbol{r} \tag{4.14}$$

In the iteration invariant case it was enough to check the spectral radius of the $\boldsymbol{F}$-matrix in order to establish uniform exponential stability. When considering the iteration variant case this test does not work alone. Next, some of the possible ways to check for uniform exponential stability are given.

The first result involves assumptions on the structure of the variability. This result is useful in the analysis of a class of adaptive ILC algorithms that will be discussed in Part V.

Lemma 4.1 (Uniform exponential stability with structured variability) *Assume that* $k = 1, \dots, M$ *and* $\boldsymbol{F}_k = \boldsymbol{V}\boldsymbol{J}_k\boldsymbol{V}^{-1} \in \mathbb{C}^{n\times n}$ *with spectral radius* $\rho(\boldsymbol{F}_k) < 1$, ρ_m *defined as*

$$\rho_m = \max_{k=1,\dots,M} \rho(\boldsymbol{F}_k) \tag{4.15}$$

and J_k *being a matrix with the following structure*

$$\begin{aligned}
\boldsymbol{J}_k &= diag(\boldsymbol{J}_{1,k}, \boldsymbol{J}_{2,k}, \dots, \boldsymbol{J}_{l,k}) \\
\boldsymbol{J}_{i,k} &= diag(\boldsymbol{J}_{i1,k}, \boldsymbol{J}_{i2,k}, \dots, \boldsymbol{J}_{im_i,k}) \\
\boldsymbol{J}_{ij,k} &= \begin{bmatrix} \lambda_{i,k} & c_k & 0 & \dots & 0 \\ 0 & \lambda_{i,k} & c_k & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ & & & \lambda_{i,k} & c_k \\ 0 & \dots & & 0 & \lambda_{i,k} \end{bmatrix} \in \mathbb{C}^{n_{ij}\times n_{ij}}
\end{aligned}$$

then $\exists\bar{\rho}$, $\rho_m \le \bar{\rho} < 1$, *and* $\gamma < \infty$ *(independent of* M*) such that* $\forall \boldsymbol{z}_0 \in \mathbb{C}^n$,

$$|\boldsymbol{F}_M \boldsymbol{F}_{M-1} \dots \boldsymbol{F}_1 \boldsymbol{z}_0| \le \gamma \bar{\rho}^M |\boldsymbol{z}_0|$$

Proof For the proof see Appendix 4.B. ■

If the matrix $\boldsymbol{F}_k$ fulfills the conditions in Lemma 4.1 it means that the corresponding linear iterative system is uniformly exponentially stable.

Another result that can be used to show uniform exponential stability is the following.

Theorem 4.3 (Uniform exponential stability, slowly varying dynamics) *Suppose that for the linear state equation*

$$\boldsymbol{z}_{k+1} = \boldsymbol{F}_k \boldsymbol{z}_k + \boldsymbol{F}_{r,k} \boldsymbol{u}_k$$

there exist constants $\alpha > 0$ and $0 \leq \mu < 1$ such that for all k, $\|\boldsymbol{F}_k\| \leq \alpha$ and $\rho(\boldsymbol{F}_k) \leq \mu$. Then there exists a positive constant β such that the system described by the linear state equation above is uniformly exponentially stable if $\|\boldsymbol{F}_k - \boldsymbol{F}_{k-1}\| \leq \beta$ for all k.

This is a standard result from linear systems theory, see e.g., Rugh (1996), and it is here given without proof. This result applies immediately to the linear time variant iterative systems. A drawback is however that it just gives the existence of a limit of the variation of the dynamics.

Finally, a useful result for showing uniform exponential stability of time variant systems is presented. This is a slightly reformulated version of Theorem 24.2 and Corollary 24.4 in Rugh (1996).

Theorem 4.4 (Uniform exponential stability) *The linear time variant iterative system in* (4.14) *is uniformly exponentially stable if it exists a constant C and a constant λ, $0 \leq \lambda < 1$, such that the maximum singular value of the matrices $\boldsymbol{F}_i$ fulfill*

$$\prod_{i=j}^{k} \bar{\sigma}(\boldsymbol{F}_i) \leq C\lambda^{k-j}$$

for all k, j such that $k \geq j$.

Proof The solution to (4.11) is uniformly exponentially stable if there exist constants C_1 and $0 \leq \lambda_1 < 1$,

$$\|\boldsymbol{z}_k\| \leq C_1 \lambda_1^{k-k_0} \|\boldsymbol{z}_0\|, \quad k \geq k_0$$

for all k_0 and $\boldsymbol{z}_0$. Obviously

$$\|z_k\| = \|\boldsymbol{F}_k \boldsymbol{F}_{k-1} \dots \boldsymbol{F}_0 \boldsymbol{z}_0\| \leq \prod_{i=0}^{k} \bar{\sigma}(\boldsymbol{F}_i) \|\boldsymbol{z}_0\| \leq C\lambda^k \|\boldsymbol{z}_0\| = C_2 \lambda^{k-k_0} \|\boldsymbol{z}_0\|$$

where the last equality holds for $C_2 = C\lambda^{k_0}$ and the result follows since $0 \leq \lambda < 1$. ■

The condition in Theorem 4.4 is rather restrictive but can be useful to prove stability for some linear time variant iterative systems. It is also possible to formulate a corollary based on the theorem.

Corollary 4.2
If $\bar{\sigma}(\boldsymbol{F}_k) < 1$ for all k then the linear iterative system in (4.11) *is uniformly exponentially stable.*

Proof Follows from Theorem 4.4 by choosing

$$\lambda = \max_{k=0,1,\ldots} \bar{\sigma}(\boldsymbol{F}_k).$$

■

To show BIBO stability for the system in (4.14), some standard results from linear system theory can be used. The following result is given without proof (for the proof see e.g., Rugh (1996)).

Lemma 4.2 (BIBO stability)
Given a uniformly exponentially stable linear state equation

$$\boldsymbol{z}_{k+1} = \boldsymbol{F}_k \boldsymbol{z}_k + \boldsymbol{F}_{r,k} \boldsymbol{r}$$

If there exists a finite constant β such that

$$\|\boldsymbol{F}_{r,k}\| \leq \beta$$

for all k, then the system is uniformly bounded-input, bounded-output stable.

This result applies immediately to the system description in (4.11) since the matrix $\boldsymbol{F}_r$ does not depend on k. If there is a k dependency for the $\boldsymbol{F}_r$-matrix, then the condition $\|\boldsymbol{F}_{r,k}\| \leq \beta$ for some finite constant β has to be checked.

To use the result in Lemma 4.2 it is necessary to establish uniform exponential stability. As was shown in the previous section: If the system is iteration invariant it is just to check that the spectral radius fulfills $\rho(\boldsymbol{F}) < 1$. For an iteration variant case the results in Lemma 4.1, Theorem 4.3, Theorem 4.4, or Corollary 4.2 have to be used. These are not the only results that can be used. Many other results from the linear system theory apply, the reader is referred to e.g., Rugh (1996) and Kailath (1980) where general linear systems theory is discussed in detail.

4.1.3 Frequency domain

This section is based on Norrlöf (2000b), Norrlöf (2000c), and Norrlöf and Gunnarsson (1999), where the use of linear iterative systems for ILC analysis is discussed. The system that will be considered in this section is given by

$$Z_{k+1}(\omega) = F(e^{i\omega}) Z_k(\omega) + F_r(e^{i\omega}) R(\omega) \tag{4.16}$$

where $Z_k(\omega)$ and $R(\omega)$ are derived from the time representation in (4.1) as defined in (3.6). The frequency domain representation of $F(q)$ and $F_r(q)$ in (4.1) is found according to (3.5) and the discussion in Section 3.1.1.

For the frequency domain representation to be well defined it is necessary that $F(q)$ and $F_r(q)$ are stable. It is also important to stress that for the case when the signals are time limited (as is the case in linear iterative systems) the frequency domain representation is an approximation. This will be investigated more thoroughly in the next section. The results given in this section are for the case when the number of samples n goes to infinity, i.e., they are only valid asymptotically.

First a result that will be used in the analysis of stability for linear iterative systems in the form (4.1) is presented.

Corollary 4.3 (An upper bound for a linear mapping)
Assume $\boldsymbol{F} \in \mathbb{C}^{n\times n}$, the spectral radius $\rho(\boldsymbol{F}) < 1$, and $\boldsymbol{v}_{\in}\mathbb{C}^n$ then

$$|\boldsymbol{F}^M v| \leq C\bar{\rho}^M$$

where $\rho(\boldsymbol{F}) \leq \bar{\rho} < 1$ and $C < \infty$ independent of M.

Proof Follows as a special case of Lemma 4.1 when $\boldsymbol{F}_k = \boldsymbol{F}$ for all k. ■

Now, a theorem is formulated that gives the condition for BIBO stability using the frequency domain representation of the linear iterative system.

Theorem 4.5 (BIBO stability)
The linear iterative system,

$$Z_{k+1}(\omega) = F(e^{i\omega})Z_k(\omega) + F_r(e^{i\omega})R(\omega)$$

is BIBO stable if the spectral radius fulfill

$$\bar{\rho} = \sup_{\omega\in[0,\pi]} \rho(F(e^{i\omega})) < 1$$

Proof For the proof see Appendix 4.C. ■

The description in (4.1) can also be used for high order linear iterative systems, e.g.,

$$\zeta_{k+1}(t) = F_1(q)\zeta_k(t) + F_2(q)\zeta_{k-1}(t) + r(t) \tag{4.17}$$

cf. Section 4.1.2. This iterative system can be written in the same form as (4.1) where

$$z_k(t) = \begin{bmatrix} \zeta_k(t) \\ \zeta_{k-1}(t) \end{bmatrix}, \quad F(q) = \begin{bmatrix} F_1(q) & F_2(q) \\ 1 & 0 \end{bmatrix}, \text{and} \quad F_r(q) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{4.18}$$

Theorem 4.5 gives the BIBO stability condition, $\rho(F(e^{i\omega})) < 1$, $\forall\omega$.

4.1.4 Relations between the two domains

First it is shown that the frequency domain result actually implies the time domain result in the case when $z_k(t)$ is a scalar.

Theorem 4.6
When $F(q)$ is causal and $z_k(t)$ is a scalar, then if

$$\sup_{\omega\in[0,\pi]} |F(e^{i\omega})| < 1$$

the first impulse coefficient fulfill,

$$|g_f(0)| < 1$$

Proof Follows by using that

$$|g_f(0)| = \Big|\frac{1}{2\pi}\int_{-\pi}^{\pi} F(e^{i\omega})d\omega\Big| < 1$$

since $|F(e^{i\omega})| < 1$ for all ω. ■

This result says that from the frequency domain representation, $F(e^{i\omega})$, it is possible to say about the time domain matrix formulation,

$$\boldsymbol{z}_{k+1} = \boldsymbol{F}\boldsymbol{z}_k + \boldsymbol{F}_r\boldsymbol{r}$$

that $\rho(\boldsymbol{F}) < 1$. This follows from the fact that $\boldsymbol{F}$ is a lower triangular Toeplitz matrix with constant diagonal elements equal to $g_f(0)$. All the eigenvalues to this matrix becomes equal to $g_f(0)$ and since the absolute value is less than one the linear iterative system is BIBO stable (according to Theorem 4.1).

Note that, if the first impulse coefficient fulfill $|g_f(0)| < 1$ it does not imply $|F(e^{i\omega})| < 1$. The result in Theorem 4.6 gives a relation between the frequency domain results and the time domain results for a finite time horizon. It is actually possible to formulate a result even stronger than $\rho(\boldsymbol{F}) < 1$ in the time domain realization (cf. Theorem 4.6).

Theorem 4.7
Suppose $F(q)$ stable and causal, $z_k(t)$ scalar, and

$$\sup_{\omega\in[0,\pi]} |F(e^{i\omega})| < 1$$

then the largest singular value of $\boldsymbol{F}_n \in \mathbb{R}^{n\times n}$ in the matrix representation of the linear iterative system

$$\boldsymbol{z}_{k+1} = \boldsymbol{F}_n\boldsymbol{z}_k + \boldsymbol{F}_r\boldsymbol{r}$$

fulfills,

$$\bar{\sigma}(\boldsymbol{F}_n) < 1$$

Proof Follows from Section 9.6 in Grenander and Szegö (1984) where it is shown that

$$\bar{\sigma}(\boldsymbol{F}_n) \leq \bar{\sigma}(\boldsymbol{F}_{n+1}) \leq \ldots \leq \lim_{n\to\infty} \bar{\sigma}(\boldsymbol{F}_n) = \sup_{\omega\in[0,\pi]} |F(e^{i\omega})|$$

and since $|F(e^{i\omega})| < 1$ for all ω the result follows. ■

Next an example shows that although the linear iterative system is BIBO stable, $|F(e^{i\omega})|$ is not necessarily less than 1.

Example 4.1

Consider the following transfer function $F(q)$,

$$F(q) = \frac{0.9014 - 1.387q^{-1} + 0.6207q^{-2} - 0.1353q^{-3}}{1 - 1.154q^{-1} + 0.657q^{-2} - 0.1353q^{-3}} \tag{4.19}$$

The poles of $F(q)$ are $0.3931 \pm 0.4619i$ and 0.3678 all with absolute value less than 1. The corresponding Nyquist diagram is shown in Figure 4.1(a) and it goes outside the unit circle for some frequencies. This means that $|F(e^{i\omega})| < 1$ is not fulfilled for all frequencies. Calculating the first Markov parameter for $F(q)$ gives $f(0) = 0.90$, which is less than 1. In Figure 4.1(b) the 2-norm of z_k is shown where $z_0(t)$ is chosen as a sequence of 15 random numbers and

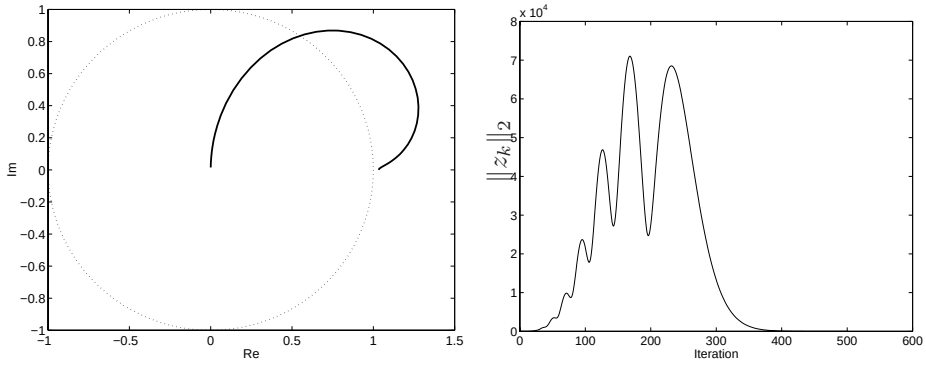
$$z_{k+1}(t) = F(q)z_k(t), \quad k \geq 0, \quad 0 < t \leq 14$$

The simulation shows that, although the value of the 2-norm first grows during a few hundred iterations it goes to zero as the number of iterations grows. ❑

Of course, in practice the behavior shown in Figure 4.1(b) might not be desirable. Instead it would be better to have a case where the 2-norm decreases as a function of iteration, i.e., never exceeds the level achieved in iteration zero. As shown in Theorem 4.7 this is achieved when $|F(e^{i\omega})| < 1$ for all ω and hence this can be considered to be a good design goal. At least if the aim is to achieve a well behaved linear iterative system. In the ILC literature the behavior shown in Figure 4.1 has been discussed in, e.g., Longman (2000). The choice of design criterion in the frequency domain comes as a natural conclusion also in Longman (2000), although the result in Theorem 4.7 is not presented explicitly.

4.2 The Tracking Formulation

Now the actual convergence of ILC systems using the *tracking formulation* will be analyzed. The analysis leads to requirements that have to be met in order to establish stability (according to the discussion in Section 3.6.2).



(a) Nyquist diagram for $F(q)$ in (4.19).

(b) The 2-norm of $z_k(t)$ when $z_{k+1}(t) = F(q)z_k(t)$.

Figure 4.1 *Example showing that the frequency domain result is not necessary for asymptotic stability.*

The analysis will be done using the results from the previous section. The results will be formulated in time domain as well as in the frequency domain. Initially the disturbance free case will be studied.

4.2.1 Disturbance free case

The starting point for the stability discussion is the disturbance free case. This means that the disturbances $v_k(t)$ and $w_k(t)$ are neglected. The system description therefore becomes completely deterministic and in the matrix form the system is described by,

$$\boldsymbol{y}_k = \boldsymbol{T}_{r,k}\boldsymbol{r} + \boldsymbol{T}_{u,k}\boldsymbol{u}_k \tag{4.20}$$

The "optimal input" for this case is found by letting $\boldsymbol{y}_k = \boldsymbol{r}$ and solve for $\boldsymbol{u}_{d,k}$,

$$\boldsymbol{u}_{d,k} = \boldsymbol{T}_{u,k}^{\dagger}(I - \boldsymbol{T}_{r,k})\boldsymbol{r} \tag{4.21}$$

where $\boldsymbol{T}_{u,k}^{\dagger}$ is the pseudo inverse, defined as

$$\boldsymbol{T}_{u,k}^{\dagger} = \boldsymbol{T}_{u,k}^{T}(\boldsymbol{T}_{u,k}\boldsymbol{T}_{u,k}^{T})^{-1}$$

Remark 4.1
If the system T_u is non-minimum phase, then the inverse transfer function becomes unstable. When considering a finite time horizon, as in the linear iterative systems

case, the gain of the matrix $\boldsymbol{T}_u^\dagger$ will not become infinite but the gain will still be considerable. This means that $\|\boldsymbol{u}_d\|$ might be very big and in practice impossible to achieve.

We will return to, and do a short note on, the case of non-minimum phase systems in Section 8.2.5. Now assume that the ILC updating formula is of first order type.

First order ILC

Here the ILC updating formula according to (3.14) is used,

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}(\boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{e}_k) \tag{4.22}$$

where $\boldsymbol{Q}$ and $\boldsymbol{L}$ can be matrix realizations of linear time invariant filters $Q(q)$ and $L(q)$, or realizations of time variant systems, Q and L. It is also possible to have iteration dependent $\boldsymbol{Q}$ and $\boldsymbol{L}$ matrices, as in (3.14), but first it will be assumed that this is not the case.

Now, using the system description in (4.20) and the ILC updating formula in (4.22) , it is clear that

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}(\boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{e}_k) = \boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_{u,k})\boldsymbol{u}_k + \boldsymbol{L}(I - \boldsymbol{T}_{r,k})\boldsymbol{r} \tag{4.23}$$

Using the results from Section 4.1 it is possible to find conditions for stability of the ILC system created by controlling the system in (4.20) using the ILC algorithm in (4.22).

Corollary 4.4 (Stability, slowly varying dynamics)
There exists a constant $\gamma > 0$ such that the system

$$\boldsymbol{y}_k = \boldsymbol{T}_{r,k}\boldsymbol{r} + \boldsymbol{T}_{u,k}\boldsymbol{u}_k$$

controlled with the ILC updating equation

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}(\boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{e}_k)$$

is stable if, for all k,

1. *$\|\boldsymbol{L}(I - \boldsymbol{T}_{r,k})\|$ and $\|\boldsymbol{F}_k\|$ are bounded by some constants,*
2. *$\rho(\boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_{u,k})) < 1$, and*
3. *$\|\boldsymbol{L}(\boldsymbol{T}_{u,k} - \boldsymbol{T}_{u,k-1})\| < \gamma$.*

Proof Use $\boldsymbol{u}_{d,k}$ from (4.21). From the results in Lemma 4.2, and Theorem 4.3 it follows that the system is BIBO stable. This means that

$$\sup_{k=0,1,\ldots} \|\boldsymbol{u}_{d,k} - \boldsymbol{u}_k\| < \infty$$

and there exists an ϵ such that ϵ-convergence can be established. ■

Corollary 4.5 (Stability, singular value condition)
The system

$$\boldsymbol{y}_k = \boldsymbol{T}_{r,k}\boldsymbol{r} + \boldsymbol{T}_{u,k}\boldsymbol{u}_k$$

controlled with the ILC updating equation

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}(\boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{e}_k)$$

is stable if

1. *for all k, $\|\boldsymbol{L}(I - \boldsymbol{T}_{r,k})\|$ is bounded by some constant and*
2. *there exist constants C and $0 \leq \lambda < 1$ such that the singular values fulfill,*

$$\prod_{i=j}^{k} \bar{\sigma}(\boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_{u,i})) \leq C\lambda^{k-j}$$

for all k, j such that $k \geq j$.

Proof Similar to the proof of Corollary 4.4 but using the results from Lemma 4.2 and Theorem 4.4. ■

The second condition in Corollary 4.5 can be replaced, according to Corollary 4.2, by $\bar{\sigma}(\boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_{u,i})) < 1$ if this is fulfilled.

When the controlled system is linear time invariant the following result gives a condition for stability.

Corollary 4.6 (Stability, linear iteration invariant case)
The system

$$\boldsymbol{y}_k = \boldsymbol{T}_r\boldsymbol{r} + \boldsymbol{T}_u\boldsymbol{u}_k$$

controlled using the ILC updating equation

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}(\boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{e}_k)$$

is stable if $\rho(\boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_u)) < 1$.

Proof Follows from Theorem 4.1 and the fact that there exists an $\epsilon < \infty$ such that

$$\sup_{k=0,1,\ldots} \|\boldsymbol{u}_d - \boldsymbol{u}_k\| < \epsilon$$

since Theorem 4.1 implies BIBO stability. $\boldsymbol{u}_d$ is from (4.21) when the system matrices are non k-dependent. ■

Before considering the actual convergence rate of the ILC control signal the asymptotic value of the control signal is presented in a lemma.

Lemma 4.3
If the system

$$\boldsymbol{y}_k = \boldsymbol{T}_r\boldsymbol{r} + \boldsymbol{T}_u\boldsymbol{u}_k$$

controlled using the ILC updating equation

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}(\boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{e}_k)$$

is stable, then the control signal will converge to

$$\boldsymbol{u}_\infty = \big(I - \boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_u)\big)^{-1}\boldsymbol{Q}\boldsymbol{L}(I - \boldsymbol{T}_r)\boldsymbol{r}$$

which implies that the asymptotic error will be

$$\boldsymbol{e}_\infty = \boldsymbol{r} - \boldsymbol{y}_\infty = \Big(I - \boldsymbol{T}_r - \boldsymbol{T}_u\big(I - \boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_u)\big)^{-1}\boldsymbol{Q}\boldsymbol{L}(I - \boldsymbol{T}_r)\Big)\boldsymbol{r}$$

Proof Use the same technique as in the proof of Corollary 4.1. Solve

$$\boldsymbol{u}_\infty = \boldsymbol{Q}(\boldsymbol{u}_\infty + \boldsymbol{L}\boldsymbol{e}_\infty)$$

with

$$\boldsymbol{e}_\infty = \boldsymbol{r} - \boldsymbol{y}_\infty = \boldsymbol{r} - \boldsymbol{T}_r\boldsymbol{r} - \boldsymbol{T}_u\boldsymbol{u}_\infty$$

and the result follows. ■

Another useful result that has an impact on the design of the ILC updating formulas is the following.

Theorem 4.8 (Condition for monotone exponential convergence of $\boldsymbol{u}_k$)
If the system

$$\boldsymbol{y}_k = \boldsymbol{T}_r\boldsymbol{r} + \boldsymbol{T}_u\boldsymbol{u}_k$$

is controlled using the ILC updating equation

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}(\boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{e}_k)$$

and $\bar{\sigma}(\boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_u)) < 1$ then the ILC system is stable and

$$\|\boldsymbol{u}_\infty - \boldsymbol{u}_k\| \leq \lambda^k\|\boldsymbol{u}_\infty - \boldsymbol{u}_0\|$$

with $\boldsymbol{u}_\infty$ defined according to Lemma 4.3.

Proof $\rho(A) < \bar{\sigma}(A)$ for a general matrix A which implies stability according to Corollary 4.6. If $\bar{\sigma}(\boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_u)) < 1$ then $\tilde{\boldsymbol{u}}_k = \boldsymbol{u}_\infty - \boldsymbol{u}_k$ will converge according to

$$\|\tilde{\boldsymbol{u}}_{k+1}\| = \|\boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_u)\tilde{\boldsymbol{u}}_k\| \leq \|\boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_u)\|\|\tilde{\boldsymbol{u}}_k\| \leq \lambda^{k+1}\|\tilde{\boldsymbol{u}}_0\|$$

where $\|\boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_u)\| = \bar{\sigma}(\boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_u)) \leq \lambda < 1$. ■

From Theorem 4.7 it follows that the condition on the maximum singular value in Theorem 4.8 can be replaced by a condition on the frequency domain representation according to

$$|1 - L(e^{i\omega t})T_u(e^{i\omega t})| < |Q^{-1}(e^{i\omega t})|, \quad \forall\omega \tag{4.24}$$

This coincides with the very common frequency domain condition given in the ILC literature. In Theorem 4.8 it is also shown that when this condition is fulfilled u_k will converge to the limit value, u_∞, exponentially and without overshoot. The result does not say anything about the resulting error compared to the true optimal input $u_d(t)$. This will be discussed next.

The true performance

It is not only stability that is of interest when designing an ILC updating formula. Also in the disturbance free case it is interesting to see what the true error becomes asymptotically.

Theorem 4.9 (Zero error convergence in the disturbance free case) *If the system*

$$\boldsymbol{y}_k = \boldsymbol{T}_r\boldsymbol{r} + \boldsymbol{T}_u\boldsymbol{u}_k$$

with the ILC updating equation

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{e}_k$$

is stable, then $\|\boldsymbol{L}\boldsymbol{e}_\infty\| = 0$.

Proof From the stability assumption, convergence of $\boldsymbol{u}_k$ to $\boldsymbol{u}_\infty$ follows. Now consider

$$\boldsymbol{u}_\infty = (I + \boldsymbol{\epsilon})\boldsymbol{u}_\infty + (I + \boldsymbol{\epsilon})\boldsymbol{L}\boldsymbol{e}_\infty$$

with $\boldsymbol{Q} = I + \boldsymbol{\epsilon}$. This means that

$$\|\boldsymbol{\epsilon}\boldsymbol{u}_\infty\| = \|(I + \boldsymbol{\epsilon})\boldsymbol{L}\boldsymbol{e}_\infty\|$$

and for $\epsilon = 0$, i.e., $\boldsymbol{Q} = I$, the result follows. ■

Note that $\|\boldsymbol{L}\boldsymbol{e}_\infty\| = 0$ does not imply $\|\boldsymbol{e}_\infty\| = 0$. If $\boldsymbol{L}$ has a null space with rank greater than 0 it is possible that $\|\boldsymbol{e}_\infty\| \neq 0$. There is no risk, however, that $\|\boldsymbol{e}_k\|$ will grow infinitely because

$$\boldsymbol{e}_k = \boldsymbol{r} - \boldsymbol{y}_k = (I - \boldsymbol{T}_r)\boldsymbol{r} - \boldsymbol{T}_{u,k}\boldsymbol{u}_k$$

and since $\|\boldsymbol{u}_k\| < \infty$ from the stability assumption, $\|\boldsymbol{e}_k\|$ will be bounded. It is also clear that when $\|\boldsymbol{L}\boldsymbol{e}_k\| = 0$

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{e}_k = \boldsymbol{u}_k$$

and since $\boldsymbol{u}_{k+1} = \boldsymbol{u}_k$ it is true that $\boldsymbol{e}_{k+1} = \boldsymbol{e}_k$ in the disturbance free case.

When $\boldsymbol{Q}$ is not chosen as an identity matrix ($Q(q)$ chosen as something else than 1) it is possible to calculate the asymptotic control signal and the corresponding error using Lemma 4.3.

High order ILC

It is also possible to formulate some results for stability using high order ILC updating formulas according to Section 3.4.1. Consider the ILC algorithm from (3.15) in the matrix form,

$$\boldsymbol{u}_{k+1} = \sum_{j=k-N+1}^{k} \Big(\boldsymbol{Q}_{k-j+1}\big(\boldsymbol{u}_j + \boldsymbol{L}_{k-j+1}\boldsymbol{e}_j\big)\Big) \tag{4.25}$$

Using the definition of $\boldsymbol{e}_k$, this can be rewritten as

$$\boldsymbol{u}_{k+1} = \sum_{j=k-N+1}^{k} \boldsymbol{Q}_{k-j+1}(I - \boldsymbol{L}_{k-j+1}\boldsymbol{T}_u)\boldsymbol{u}_j + \sum_{j=1}^{N} \boldsymbol{Q}_j\boldsymbol{L}_j(I - \boldsymbol{T}_r)\boldsymbol{r} \tag{4.26}$$

Define $\boldsymbol{U}_k$ as

$$\boldsymbol{U}_k = \begin{bmatrix} \boldsymbol{u}_k^T & \boldsymbol{u}_{k-1}^T & \dots & \boldsymbol{u}_{k-N+1}^T \end{bmatrix}^T \tag{4.27}$$

this means that (4.26) can be written as

$$\boldsymbol{U}_{k+1} = \begin{bmatrix} \boldsymbol{F}_1 & \boldsymbol{F}_2 & \dots & \boldsymbol{F}_N \\ I & 0 & \dots & 0 \\ & \ddots & \ddots & \vdots \\ 0 & \dots & I & 0 \end{bmatrix} \boldsymbol{U}_k + \begin{bmatrix} \sum_{j=1}^{N} \boldsymbol{Q}_j\boldsymbol{L}_j(I - \boldsymbol{T}_r) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \boldsymbol{r} \tag{4.28}$$

with $\boldsymbol{F}_j = \boldsymbol{Q}_j(I - \boldsymbol{L}_j\boldsymbol{T}_u)$. The condition for stability of the system using the high order ILC algorithm can be formulated according to the next corollary.

Corollary 4.7 (Stability, linear iteration invariant case)
The system

$$\boldsymbol{y}_k = \boldsymbol{T}_r \boldsymbol{r} + \boldsymbol{T}_u \boldsymbol{u}_k$$

controlled with the ILC updating equation

$$\boldsymbol{u}_{k+1} = \sum_{j=k-N+1}^{k} \Big(\boldsymbol{Q}_{k-j+1} \big(\boldsymbol{u}_j + \boldsymbol{L}_{k-j+1} \boldsymbol{e}_j \big) \Big)$$

is stable if the spectral radius fulfills

$$\rho\left(\begin{bmatrix} \boldsymbol{F}_1 & \boldsymbol{F}_2 & \dots & \boldsymbol{F}_N \\ I & 0 & \dots & 0 \\ & \ddots & \ddots & \vdots \\ 0 & \dots & I & 0 \end{bmatrix} \right) < 1$$

where $\boldsymbol{F}_j = \boldsymbol{Q}_j (I - \boldsymbol{L}_j \boldsymbol{T}_u)$.

Proof It is straightforward to apply Theorem 4.1 in order to show BIBO stability which implies that it exists an $\epsilon < \infty$ such that

$$\lim_{k \to \infty} \| \boldsymbol{U}_d - \boldsymbol{U}_k \| < \epsilon$$

$\boldsymbol{U}_d$ is a vector according to (4.27) with all elements equal to $\boldsymbol{u}_d$ defined as in (4.21). ■

If the condition on the spectral radius in Corollary 4.7 can be replaced with the maximum singular value then the system is stable and $\| \boldsymbol{U}_\infty - \boldsymbol{U}_k \|$ will converge monotonically and exponentially towards zero (cf. Theorem 4.8).

A result similar to Corollary 4.5 can be formulated also for the high order ILC case. First let

$$\boldsymbol{U}_{k+1} = \boldsymbol{F}_k \boldsymbol{U}_k + \boldsymbol{F}_{r,k} \boldsymbol{r} \tag{4.29}$$

with

$$\begin{aligned} \boldsymbol{F}_k &= \begin{bmatrix} \boldsymbol{F}_{1,k} & \boldsymbol{F}_{2,k} & \dots & \boldsymbol{F}_{N,k} \\ I & 0 & \dots & 0 \\ & \ddots & \ddots & \vdots \\ 0 & \dots & I & 0 \end{bmatrix}, \\ \boldsymbol{F}_{r,k} &= \begin{bmatrix} \sum_{j=k-N+1}^{k} \boldsymbol{Q}_{k-j+1} \boldsymbol{L}_{k-j+1} (I - \boldsymbol{T}_{r,j}) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \end{aligned} \tag{4.30}$$

and $\boldsymbol{F}_{j,k} = \boldsymbol{Q}_j (I - \boldsymbol{L}_j \boldsymbol{T}_{u,(k-j+1)})$.

Corollary 4.8 (Stability, slowly varying dynamics)
There exists a constant $\gamma > 0$ such that the system

$$\boldsymbol{y}_k = \boldsymbol{T}_{r,k}\boldsymbol{r} + \boldsymbol{T}_{u,k}\boldsymbol{u}_k$$

controlled by the ILC updating formula

$$\boldsymbol{u}_{k+1} = \sum_{j=k-N+1}^{k} \Big(\boldsymbol{Q}_{k-j+1}\big(\boldsymbol{u}_j + \boldsymbol{L}_{k-j+1}\boldsymbol{e}_j\big)\Big)$$

is stable if, for all k,

1. *$\|\boldsymbol{F}_k\|$ and $\|\boldsymbol{F}_{r,k}\|$ are bounded by some constants,*
2. *$\rho(\boldsymbol{F}_k) < 1$, and*
3. *$\|\boldsymbol{F}_k - \boldsymbol{F}_{k-1}\| < \gamma$*

with $\boldsymbol{F}_k$ and $\boldsymbol{F}_{r,k}$ according to (4.30).

Proof Similar to the proof of Corollary 4.4. ■

Corollary 4.9 (Stability, singular value condition)
The system

$$\boldsymbol{y}_k = \boldsymbol{T}_{r,k}\boldsymbol{r} + \boldsymbol{T}_{u,k}\boldsymbol{u}_k$$

controlled by the ILC updating formula

$$\boldsymbol{u}_{k+1} = \sum_{j=k-N+1}^{k} \Big(\boldsymbol{Q}_{k-j+1}\big(\boldsymbol{u}_j + \boldsymbol{L}_{k-j+1}\boldsymbol{e}_j\big)\Big)$$

is stable if

1. *for all k, $\|\boldsymbol{F}_{r,k}\|$ is bounded by some constant and*
2. *there exist constants C and $0 \leq \lambda < 1$ such that the singular values fulfill,*

$$\prod_{i=j}^{k} \bar{\sigma}(\boldsymbol{F}_k) \leq C\lambda^{k-j}$$

for all k, j such that $k \geq j$.

Proof Similar to the proof of Corollary 4.5. ■

If $\bar{\sigma}(\boldsymbol{F}_k) < 1$ for all k stability follows as was shown in Corollary 4.2.

The frequency domain results will be discussed for the case of a second order ILC algorithm in Part IV. Next the case where the disturbances cannot be neglected will be discussed.

4.2.2 Disturbance aspects for the first order ILC

In this section we will discuss questions related to what happens when ILC is applied to systems where the disturbances can not be neglected. The system description that will be used here is from (3.1), i.e.,

$$\begin{aligned} z_k(t) &= T_r(q)r(t) + T_u(q)u_k(t) + T_w(q)w_k(t) \\ y_k(t) &= z_k(t) + T_v(q)v_k(t) \end{aligned}$$

with the ILC updating equation

$$u_{k+1}(t) = Q(q)(u_k(t) + L(q)e_k(t)$$

from (3.13). The error $e_k(t)$ is defined as

$$e_k(t) = r(t) - y_k(t)$$

Next the equation relating the errors in two iterations will be discussed.

Error equation

A fundamental issue for a successful ILC algorithm is that the error is reduced as a function of iteration. The error is here defined as

$$\epsilon_k(t) = r(t) - z_k(t) \tag{4.31}$$

First the equation that describes how the error evolves is presented, the result is limited to the case where the system is scalar.

Lemma 4.4 (Error equation)
Consider the system

$$\begin{aligned} z_k(t) &= T_r(q)r(t) + T_u(q)u_k(t) + T_w(q)w_k(t) \\ y_k(t) &= z_k(t) + T_v(q)v_k(t) \end{aligned}$$

with the ILC updating equation

$$u_{k+1}(t) = Q(q)\big(u_k(t) + L(q)e_k(t)\big)$$

with $e_k(t) = r(t) - y_k(t)$. *Define* $\tilde{\epsilon}$ *as*

$$\tilde{\epsilon}(t) = (1 - T_r(q))r(t) \tag{4.32}$$

i.e., the disturbance free error signal obtained without ILC when $u(t) = 0$. *The error,* $\epsilon_k(t) = r(t) - z_k(t)$, *is updated as*

$$\begin{aligned} \epsilon_{k+1}(t) = Q(q)(1 - T_u(q)L(q))\epsilon_k(t) + (1 - Q(q))\tilde{\epsilon}(t) \\ + T_u(q)Q(q)L(q)T_v(q)v_k(t) + T_w(q)(Q(q)w_k(t) - w_{k+1}(t)) \end{aligned}$$

Proof Using the system description and the ILC algorithm the following expression for ϵ_{k+1} can be found,

$$\epsilon_{k+1}(t) = \tilde{\epsilon}(t) - T_u(q)Q(q)u_k(t) - T_u(q)Q(q)(\epsilon_k(t) - T_v(q)v_k) - T_w(q)w_{k+1}(t)$$

and from the fact that

$$-T_u(q)u_k(t) = \epsilon_k(t) - \tilde{\epsilon}(t) + T_w(q)w_k(t)$$

the result follows. ■

A similar result as in Lemma 4.4 is also presented in Panzieri and Ulivi (1995) for an open loop case ($T_u(q)$ represents an open loop system) and for the system disturbance only.

Lemma 4.4 shows that there are three types of driving terms for the error. There is the term $\tilde{\epsilon}(t)$ that comes from the fact that the true system does not exactly correspond to the ideal system. The other driving terms come from the measurement noise and the system disturbance, and they enter the equation in slightly different ways. The long term effects of the two last terms will be discussed in the next sections.

System disturbances

Neglect the measurement disturbance, $v_k(t)$, i.e., let $v_k(t) = 0$, and assume that $w_k(t) = w(t)$, i.e., the system disturbance is not k-dependent. If one of the stability conditions from Section 4.2.1 is fulfilled, then the final value of the error becomes,

$$\lim_{k\to\infty} \epsilon_k(t) = \frac{(1 - Q(q))\tilde{\epsilon} + T_w(q)(Q(q) - 1)w(t)}{1 - Q(q)(1 - T_u(q)L(q))} \tag{4.33}$$

It is possible to interpret the result in (4.33) in the frequency domain,

$$\lim_{k\to\infty} \mathcal{E}_k(\omega) = \frac{(1 - Q(e^{i\omega}))\tilde{\mathcal{E}}(\omega) + T_w(e^{i\omega})(Q(e^{i\omega t}) - 1)W(e^{i\omega})}{I - Q(e^{i\omega})(1 - T_u(e^{i\omega})L(e^{i\omega}))} \tag{4.34}$$

where the transformation to the frequency domain is done as discussed in Section 3.1.1. This expression illustrates that the frequency content of the system disturbance $w_k(t)$, and the cut-off frequencies of $T_w(e^{i\omega})$ and $1 - Q(e^{i\omega})$ respectively, will determine how well the ILC algorithm handles system disturbances of repetitive character.

When the measurement disturbance is neglected, we get from Lemma 4.4,

$$\begin{aligned}\epsilon_{k+1}(t) = Q(q)(1 - L(q)T_u(q))\epsilon_k(t) + (1 - Q(q))\tilde{\epsilon}(t) \\ + T_w(q)(Q(q)w_k(t) - w_{k+1}(t))\end{aligned}$$

If $Q(q) = 1$ then the system disturbances contribute to the error by their difference between the iterations. If the disturbances are repetitive, in the sense that the disturbance signals $w_k(t) = w(t)$ for all k, then the contribution to the error difference equation is zero.

When $Q(q) \neq 1$, the system disturbance will act as a driving term similar to the initial error $\tilde{\epsilon}(t)$. In a typical case are $T_w(q)$ and $Q(q)$ both of low pass type, which means that $1 - Q(q)$ is of high pass type. Multiplying these two will give a band pass filter and the possibilities to reduce the effects of the load disturbance hence depends on the relationship between the frequency content of the load disturbance and the cut-off frequency of the filters $T_w(q)$ and $Q(q)$.

Measurement disturbances

To study the effect of measurement disturbances alone, assume that the reference input and the system disturbance are equal to zero. This means that $r(t) = 0$ and $w_k(t) = 0$. Using Lemma 4.4 the error can therefore be expressed as

$$\epsilon_{k+1}(t) = Q(q)(1 - T_u(q)L(q))\epsilon_k(t) + T_u(q)Q(q)L(q)T_v(q)v_k(t) \tag{4.35}$$

Assume that the measurement disturbance $v_k(t) = \nu(k \cdot t)$ where ν is a stationary stochastic process with spectral density $\Phi_\nu(\omega)$. Asymptotically this means that the spectral density of the error $\Phi_\epsilon(\omega)$ can be written according to

$$\Phi_\epsilon(\omega) = \frac{|T_u(e^{i\omega})Q(e^{i\omega})L(e^{i\omega})T_v(e^{i\omega})|^2}{1 - |Q(e^{i\omega})(1 - T_u(e^{i\omega})L(e^{i\omega}))|^2}\Phi_\nu(\omega) \tag{4.36}$$

It is clear from (4.36) that the magnitude of the error spectral density will be very large for frequencies where $|Q(e^{i\omega})(1-L(e^{i\omega})T_u(e^{i\omega}))|$ is close to one. Obviously the Q-filter can be used to make $|Q(e^{i\omega})(1 - L(e^{i\omega})T_u(e^{i\omega}))|$ less than one by choosing $|Q(e^{i\omega})|$ small when $|(1 - L(e^{i\omega})T_u(e^{i\omega}))|$ is close to one.

Simulations

To illustrate the disturbance properties a simulation example is used. For details see Norrlöf and Gunnarsson (2000a). In Figure 3.2 the controlled system is shown as a block diagram. It is a simplified description of a single robot joint modeled as a double integrator. The robot joint is controlled by a discrete time PD-regulator and the feed-forward filter is, simply, a double backward differentiation approximation.

The filter $L(q)$ is chosen in a model based way according to

$$L(q) = \widehat{T}_u^{-1}(q)(1 - H_B(q)) \tag{4.37}$$

where $\widehat{T}_u(q)$ denotes a nominal closed loop transfer function obtained using the model of the open loop system and the known controller. The design algorithm

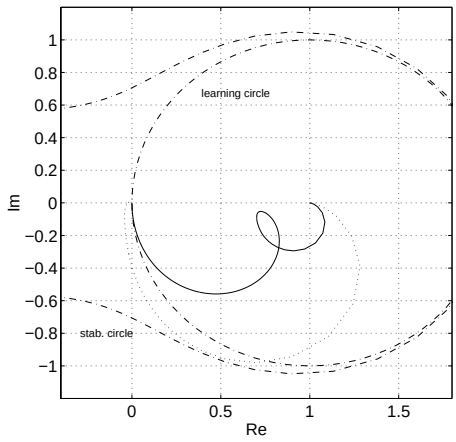


Figure 4.2 *T_u (dotted), T_uL for L designed using a nominal model (solid) and $|Q^{-1}(e^{e^{i\omega}})|$ for $Q = 1$ and Q chosen as a low pass filter (dash-dotted).*

will be presented in more detail in Chapter 8. It is assumed that there is a 30% model error in the moment of inertia in the model of the system. The filter H_B is a Butterworth high pass filter (here of second order) for which the gain tends to one for high frequencies. Choosing L according to this design rule, with cut-off frequency of the high pass filter equal to 0.4 times the Nyquist frequency, gives the Nyquist curve depicted in Figure 4.2. T_u is also shown in Figure 4.2 for comparison.

Figure 4.2 also shows the right hand side of (4.24) for $Q = 1$ and Q chosen as a low-pass filter respectively and it is possible to see how the stability region is increased when Q is chosen as a low-pass filter.

The first goal is to investigate how the system disturbance influences the ILC algorithm. A disturbance signal, $w(t)$ in Figure 3.2, is applied at the same time every cycle as a repetitive disturbance. Figure 4.3 shows the spectrum of ϵ_k as a function of iteration. Clearly the introduction of a filter Q reduces the convergence speed of the ILC algorithm and the convergence will no longer be to an error having zero size. Already after one iteration, when the filter Q is used, the energy in the error has almost reached its final value and also in the spectrum it is possible to see that there is not much change after the first iteration.

The next step is to introduce a measurement disturbance, v_k. This disturbance is chosen as a discrete time white noise process with a constant standard deviation. The frequency domain properties of the resulting errors are shown in Figure 4.4. The plot of the energy clearly illustrates that the use of the Q-filter reduces the error energy caused by the measurement disturbance. In Norrlöf and Gunnarsson

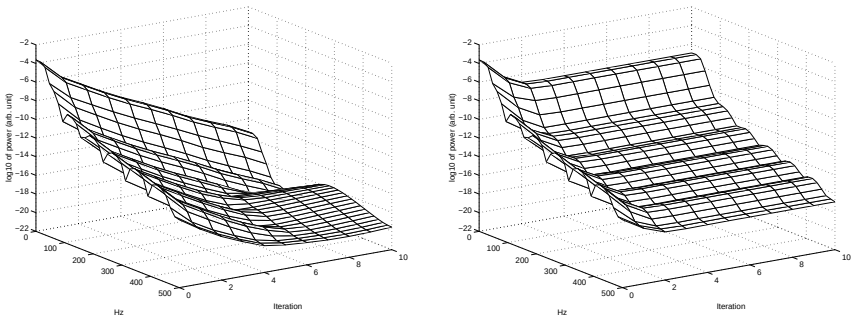


Figure 4.3 *Error signal spectrum without Q filter (left) and error signal spectrum with Q filter (right).*

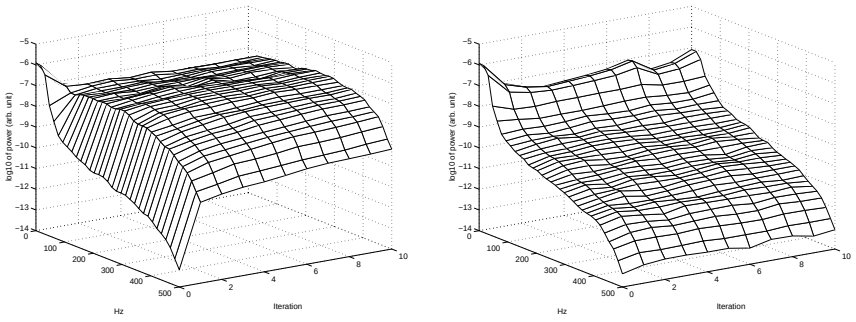


Figure 4.4 *Error signal spectrum, without Q filter (left) and error signal spectrum with Q filter (right).*

(2000a) it is shown that the maximum error in time domain behaves in a similar way for the two choices of Q. This is explained by the fact that the maximum error is of low frequency character and hence not affected by the choice of Q. In Figure 4.4 the spectrum of the position error signal $r - z_k$ is shown with and without Q filter. The reduction of the high frequency part of the error spectrum is clearly shown.

As was shown in the example it is possible to reduce the measurement disturbance impact on the ILC algorithm by introducing the Q-filter. The price for this, as was shown in Theorem 4.9, is that the asymptotic error, not even in the disturbance free case, is guaranteed to be zero. Trajectory following and measurement disturbance rejection will in this sense be contradictory.

4.3 The Disturbance Rejection Formulation

Now we will analyze an ILC algorithm using the *disturbance rejection formulation*. The analysis is limited to the algorithm proposed in Section 3.5. The aim is to show some aspects of iteration variant ILC updating schemes and the continuation of this work is presented in Part V.

Before doing the analysis, some assumptions on the disturbances and the system is presented.

4.3.1 Assumptions

The system description that will be used here is the LTI version of (3.10), i.e.,

$$\begin{aligned} z_k(t) &= G^0(q)u_k(t) + d(t) \\ y_k(t) &= z_k(t) + n_k(t) \end{aligned} \tag{4.38}$$

with the ILC updating equation from (3.30),

$$u_{k+1}(t) = u_k(t) - \frac{1}{(k+1)G(q)} y_k(t) \tag{4.39}$$

where $u_0(t)$ is chosen as $u_0(t) = 0$. In the system description in (4.38) it is assumed that the system disturbance $d(t)$ is repetitive, i.e., does not depend on the iteration k. The measurement disturbance $n_k(t)$ is assumed to be equal to $\nu(\bar{t})$ where $\bar{t} = k \cdot t$ and $\nu(\bar{t})$ represents a white stationary stochastic process with zero mean and variance r_n. The expected value, $E\{n_k(t)\}$, is therefore with respect to the underlying process ν, and

$$E\{n_k(t)\} = 0$$

The variance becomes

$$\mathrm{Var}\{n_k(t)\} = r_n$$

and since ν is white $E\{n_i(t)n_j(t)\}$ equal r_n if and only if $i = j$ and 0 otherwise. This is true also for different t in the same iteration, i.e., $E\{n_k(t_1)n_k(t_2)\}$. Note that since $d(t)$ is a deterministic signal, the expected value becomes $E\{d(t)\} = d(t)$.

The goal for the ILC algorithm applied to the system in (4.38) is to find an input signal $u_k(t)$ such that the disturbance $d(t)$ is completely compensated for. Clearly the optimal solution is to find a u_k such that

$$u_k(t) = \left(G^0(q)\right)^{-1} d(t)$$

which has also been discussed in Section 3.2.2. In the next sections, different iterative solutions to this problem will be discussed.

4.3.2 $G^0(q)$ is known

Consider the estimator from (3.28),

$$\hat{d}_k(t) = \frac{1}{k}\sum_{j=0}^{k-1}\Big(y_k(t) - G(q)u_k(t)\Big)$$

When the system is known, i.e., $G(q) = G^0(q)$, and the disturbance $n_k(t)$ is defined as in Section 4.3.1, then asymptotically, it gives an unbiased estimate of the disturbance $d(t)$,

$$\lim_{k\to\infty}\hat{d}_k(t) = \lim_{k\to\infty}\frac{1}{k}\sum_{j=0}^{k-1}\Big(d(t) + n_j(t)\Big) = d(t) \tag{4.40}$$

From the ILC perspective this implies that the algorithm will converge to zero error. Obviously, it is not only the fact that the estimate is unbiased that is of interest. Also the variance of the estimate is an important property. The variance is given by,

$$\begin{aligned}
&\mathrm{Var}(\hat{d}_k(t)) = E\{\hat{d}_k^2(t)\} - (E\{\hat{d}_k(t)\})^2 = \\
&\quad E\Big\{\frac{1}{k^2}\sum_{i=0}^{k-1}\big(d(t) + n_i(t)\big)\sum_{j=0}^{k-1}\big(d(t) + n_j(t)\big)\Big\} - d^2(t) = \\
&\quad \frac{1}{k^2}\sum_{i=0}^{k-1}\sum_{j=0}^{k-1}E\{d^2(t) + d(t)(n_i(t) + n_j(t)) + n_i(t)n_j(t)\} - d^2(t) = \frac{r_n}{k}
\end{aligned} \tag{4.41}$$

Where the last equality follows from the fact that $d(t)$ is deterministic, $E\{n_i(t)\} = 0$, and that $E\{n_i(t)n_j(t)\} = r_n$ if $i = j$ and 0 otherwise, see Section 4.3.1.

Interesting is also to see how the resulting control, $u_k(t)$, develops. Using the updating equation in (4.39) with the true system $G^0(q)$ instead of $G(q)$ and $u_0(t) = 0$ the output $z_1(t)$ becomes,

$$z_1(t) = G^0(q)u_1(t) + d(t) = -G^0(q)\frac{1}{G^0(q)}(d(t) + n_0(t)) + d(t) = -n_0(t) \tag{4.42}$$

This means that $d(t)$ is completely compensated for and the mathematical expectation of $z_1(t) = 0$ when the system is known. What can be improved is however the variance of $z_k(t)$. The variance of z_1 is readily calculated as

$$\mathrm{Var}(z_1(t)) = r_n \tag{4.43}$$

The best result that can be achieved is when the disturbance $d(t)$ is perfectly known. This gives

$$z(t) = -G^0(q)\frac{1}{G^0(q)}d(t) + d(t) = 0 \tag{4.44}$$

i.e., zero variance.

The proposed algorithm from (4.39) is evaluated in a simulation. The measure utilized in the evaluation is

$$V_k = \frac{1}{r_n} \cdot \frac{1}{n-1} \sum_{t \in [0, t_f]} z_k^2(t) \tag{4.45}$$

i.e., the variance of z_k normalized with the variance of the measurement disturbance. From (4.43) it is clear that $V_1 = 1$ which is also shown in Figure 4.5. For $k = 0$ the measure V_0 does not correspond to a variance since $z_0(t) = d(t)$. V_0 therefore depends only on the size of the disturbance $d(t)$. The simulation is however done to show what happens with the variance of the output $z_k(t)$ for $k \geq 1$.

A rapid decrease of V_k can be seen in the first iterations. After 10 iterations, for example, the measure is reduced to 0.1. To reduce the last 0.1 units down to 0, however, takes infinitely many iterations. The conclusion from this simulation is that the use of the proposed ILC algorithm gives an increased performance in the case when the system is completely known but the disturbance is unknown. In the next section the properties of the method will be examined when the system is not completely known.

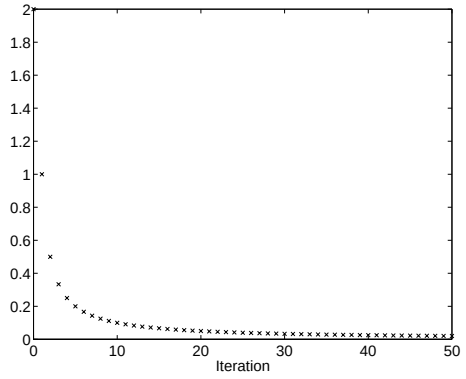


Figure 4.5 *Evaluation of V_k from* (4.45) *in a simulation.*

4.3.3 Some notes on the asymptotic and transient behavior

In practice it is clear that a model of the true system has to be used in the ILC algorithm. In this section some results based on simulations will be discussed. The transient behavior of the proposed algorithm is highlighted and compared with another algorithm. In applications it is often required that the algorithm should give a small error after, perhaps, the first 10 iterations. If it takes 100 iterations

or more to get the desired level of the errors the method will probably not be considered useful at all.

Consider the following ILC updating scheme,

$$u_{k+1}(t) = u_k(t) - L_k(q)y_k(t) \tag{4.46}$$

applied to the system in (4.38) with the following choices of the filter $L_k(q)$,

$$L_k(q) = (G^0(q))^{-1} \tag{4.47a}$$
$$L_k(q) = (G(q))^{-1} \tag{4.47b}$$
$$L_k(q) = \frac{1}{k+1}(G^0(q))^{-1} \tag{4.47c}$$
$$L_k(q) = \frac{1}{k+1}(G(q))^{-1} \tag{4.47d}$$

Assume that the ILC updating scheme in (4.46) gives a stable ILC system for all the different L_k-filters in (4.47). The system G^0 is given by

$$G^0(q) = \frac{0.07q^{-1}}{1 - 0.93q^{-1}} \tag{4.48}$$

and the model G by

$$G(q) = \frac{0.15q^{-1}}{1 - 0.9q^{-1}} \tag{4.49}$$

To compare the transient behavior of the four ILC schemes created by using the updating scheme from (4.46) and the filters from (4.47) a simulation is performed. The system used in the simulation is given by (4.38) and the actual system description by (4.48). The model of the system, available for the ILC control scheme, is given by (4.49). The variance of the additive noise, $n_k(t)$, is set to 10^{-3}.

To evaluate the result from the simulations the following measure is used

$$V(z_k) = \frac{1}{n-1}\sum_{t=1}^{n} z_k^2(t) \tag{4.50}$$

which is an estimate of the variance if z_k is a random variable with zero mean. In Figure 4.6 the results from the simulations are shown. In the first iteration $V(z_0)$ contains only the value of $V(d)$ and for the d used in the simulations $V(d) = 0.182$. Obviously the ILC schemes, (4.47a) and (4.47c), give similar results in iteration 1 since both use the inverse of the true system to find the next control input. The pair, (4.47b) and (4.47d), give for the same reason similar results after one iteration.

Figure 4.6 shows the general behavior that can be expected from the different ILC approaches covered by (4.46) and (4.47). It is clear that among the methods

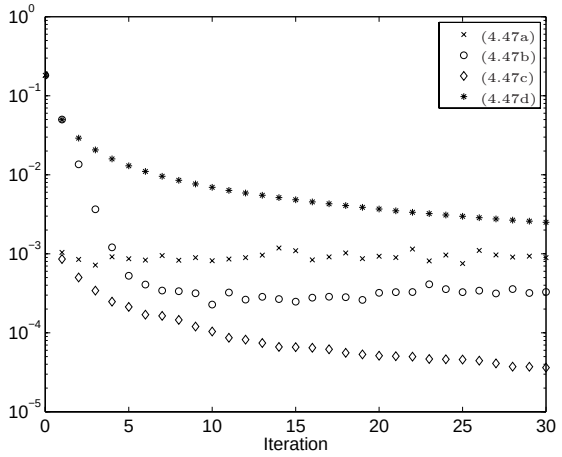


Figure 4.6 *The transient behavior of V_k for the 4 different ILC schemes given by* (4.46) *and* (4.47).

described here the approach given by (4.47c) is the best choice, although this method requires that the system description is completely known. If the system is not known as in (4.47d) the result may be not so good, cf. Figure 4.6.

For the asymptotic analysis the case when the L_k-filter is chosen according to (4.47a) is first considered. Since $u_0(t) = 0$, this means that $z_0(t) = d(t)$. From (4.38) it now follows that

$$y_0(t) = d(t) + n_k(t)$$

and therefore

$$u_1(t) = -(G^0)^{-1}(d(t) + n_0(t))$$

As was shown in (4.42), the corresponding $z_1(t)$ becomes $z_1(t) = -n_0(t)$. This means that

$$u_2(t) = -(G^0)^{-1}(d(t) + n_1(t))$$

and $z_2(t) = -n_1(t)$. The asymptotic value of $V(z_k)$ for $k > 0$ therefore becomes equal to r_n since with this approach $z_k(t) = -n_{k-1}(t),\ k > 0$.

A more general case is when a model of the system G^0 is used in the L-filter. This corresponds to the choice of filter for the ILC according to (4.47b). Obviously it is true that $z_0(t) = d(t)$ again. Now assume that the relation between the true system and the model can be described according to

$$G^0(q) = (1 + \Delta_G(q))G(q) \tag{4.51}$$

where $\Delta_G(q)$ is a relative model uncertainty. Using the ILC updating scheme in (4.46) and the filter in (4.47b), it is straightforward to arrive at

$$z_1(t) = -\Delta_G(q)d(t) - (1 + \Delta_G(q))n_0(t) = -\Delta_G(q)z_0(t) - (1 + \Delta_G(q))n_0(t)$$

and in the general case

$$z_k(t) = -\Delta_G(q)z_{k-1}(t) - (1 + \Delta_G(q))n_{k-1}(t) \tag{4.52}$$

which can be expanded into the following finite sum

$$z_k(t) = -(1 + \Delta_G(q)) \sum_{j=1}^{k} (-\Delta_G(q))^{j-1} n_{k-j}(t) + (-\Delta_G(q))^k d(t) \tag{4.53}$$

Clearly (4.52) and (4.53) are valid also for the case when there is no model error, i.e., $\Delta_G(q) = 0$.

To understand why, in this case, using a model of the system gives better asymptotic performance compared to using the true system, consider (4.52) and (4.53). If $\|\Delta_G\| < 1$ then, for a large enough k, the influence of $d(t)$ can be neglected, since $\|\Delta_G\|^k$ becomes small. Now assume that the model uncertainty is mainly a scaling error, i.e., the dynamics are captured by the model. This means that $\Delta_G(q) = \delta$ for some δ, with $|\delta| < 1$. Since the effect of $d(t)$ in $z_k(t)$ is neglected the expected value of $z_k(t)$ becomes equal to 0. The variance expression is found using, e.g., (4.52) and

$$r_{z,k} = E\{z_k^2(t)\} \approx \delta^2 r_{z,k-1} + (1 + \delta)^2 r_n \tag{4.54}$$

Asymptotically this means that

$$r_{z,\infty} \approx r_n \cdot \frac{1 + \delta}{1 - \delta} \tag{4.55}$$

In the example when $L_k(q)$ from (4.47b) is used, $\delta \approx -\frac{1}{2}$ and using the result in (4.55) it follows that $r_{z,\infty} \approx \frac{r_n}{3}$, i.e., $r_{z,\infty} \approx 0.3 \cdot 10^{-3}$. In fact this is also what is shown in Figure 4.6. The conclusion from this is that it is possible to get a lower value of $V(z_k)$ asymptotically by choosing a model such that $G^0(q) = \kappa G(q)$ for some $0 < \kappa < 1$ and let $L(q) = (G(q))^{-1}$. Clearly this is the answer why it is, in this case, better to use a model of the system.

When the true model is known and used in the filter according to (4.47c), then the value of $V(z_k)$ becomes equal to $\frac{r_n}{k}$ for $k > 0$. For $k = 0$, $V(z_0)$ is equal to $V(d)$ since $z_0 = d$. If the true system is not known and the model based approach in (4.47d) is used, then the equation corresponding to (4.52) becomes

$$z_{k+1}(t) = \frac{k - \Delta_G(q)}{k + 1} z_k(t) - \frac{1 + \Delta_G(q)}{k + 1} n_k(t) \tag{4.56}$$

with $z_0(t) = d(t)$. To prove stability and find the asymptotic value of $V(z)$ for (4.56) is left for future work. From Figure 4.6 is however clear that this method does not always give a good transient behavior. This depends on the fact that the disturbance $d(t)$ is not completely compensated for in the first iteration. Since the gain is decreased at every iteration the amount of the disturbance, $d(t)$, that will be compensated for will decrease in every iteration. This means that instead of being dominated by the random disturbance the measure $V(z)$ will instead be dominated by a term depending on the disturbance $d(t)$. In Part V a method that adaptively compensate for this will be presented.

Appendix

4.A The Jordan canonical form

The Jordan canonical representation for matrices is a standard result from linear algebra given here for readability.

Theorem 4.A.10 (Jordan canonical representation)
For any square matrix $\boldsymbol{F} \in \mathbb{C}^{n\times n}$,
there exists a nonsingular matrix $\boldsymbol{V}$ *such that*

$$\boldsymbol{F} = \boldsymbol{V}\boldsymbol{J}\boldsymbol{V}^{-1} \tag{4.A.57}$$

where

$$\begin{aligned}
\boldsymbol{J} &= diag(\boldsymbol{J}_1, \boldsymbol{J}_2, \dots, \boldsymbol{J}_l) \\
\boldsymbol{J}_i &= diag(\boldsymbol{J}_{i1}, \boldsymbol{J}_{i2}, \dots, \boldsymbol{J}_{im_i}) \\
\boldsymbol{J}_{ij} &= \begin{bmatrix} \lambda_i & 1 & & & \\ & \lambda_i & 1 & & \\ & & \ddots & \ddots & \\ & & & \lambda_i & 1 \\ & & & & \lambda_i \end{bmatrix} \in \mathbb{C}^{n_{ij}\times n_{ij}}
\end{aligned} \tag{4.A.58}$$

with zeros outside the diagonals, $\sum_{i=1}^{l}\sum_{j=1}^{m_i} n_{ij} = n$, *and with* λ_i $i = 1, \dots, l$ *as the distinct eigenvalues of* $\boldsymbol{F}$.

The transformation $\boldsymbol{V}$ can be expressed in the following way,

$$\begin{aligned}
\boldsymbol{V} &= \begin{bmatrix} \boldsymbol{V}_1 & \boldsymbol{V}_2 & \dots & \boldsymbol{V}_l \end{bmatrix} \\
\boldsymbol{V}_i &= \begin{bmatrix} \boldsymbol{V}_{i1} & \boldsymbol{V}_{i2} & \dots & \boldsymbol{V}_{im_i} \end{bmatrix} \\
\boldsymbol{V}_{ij} &= \begin{bmatrix} \boldsymbol{v}_{ij1} & \boldsymbol{v}_{ij2} & \dots & \boldsymbol{v}_{ijn_{ij}} \end{bmatrix}
\end{aligned} \tag{4.A.59}$$

where $\boldsymbol{v}_{ij1}$ are the eigenvectors of $\boldsymbol{F}$, $\boldsymbol{F}\boldsymbol{v}_{ij1} = \lambda_i \boldsymbol{v}_{ij1}$. For $\boldsymbol{v}_{ijk}$ with $k \geq 2$ the following relation holds

$$\boldsymbol{F}\boldsymbol{v}_{ijk} = \lambda_i \boldsymbol{v}_{ijk} + \boldsymbol{v}_{ij(k-1)} \tag{4.A.60}$$

and these vectors are called generalized eigenvectors of $\boldsymbol{F}$.

4.B Proof of Lemma 4.1

The following lemma will prove useful in showing the result.

Lemma 4.B.5
If $0 < \rho < 1$, then it is true that there exists a constant C and $\bar{\rho}$ such that

$$k\rho^k < C\bar{\rho}^k$$

with $\rho < \bar{\rho} < 1$.

Proof Using that $\ln k < C_0 k$ for some $k > k_0$ and by choosing $C_0 = -\frac{1}{2}\ln\rho$ it is easy to see that

$$k\rho^k = e^{k\ln\rho + \ln k} < e^{k\frac{1}{2}\ln\rho} = \sqrt{\rho}^k$$

for $k > k_0$. Let $\bar{\rho}$ be chosen as $\sqrt{\rho} < \bar{\rho} < 1$ and C such that

$$k\rho^k < C\bar{\rho}^k$$

also for $k = 1, \ldots, k_0$. This concludes the proof. ■

To get an upper bound of $|\boldsymbol{F}_M \boldsymbol{F}_{M-1} \ldots \boldsymbol{F}_1 \boldsymbol{z}_0|$ the mapping of one general column of V, v_{ijk}, is considered. From (4.A.60) it follows that if $k \leq n_{ij}$, then

$$\boldsymbol{F}_M \boldsymbol{F}_{M-1} \ldots \boldsymbol{F}_1 \boldsymbol{v}_{ijk} = \sigma_{i,k}\boldsymbol{v}_{ijk} + \sigma_{i,k-1}\boldsymbol{v}_{ij(k-1)} + \ldots + \sigma_{i,1}\boldsymbol{v}_{ij1} \tag{4.B.61}$$

where each $\sigma_{i,s}$ is a sum of all combinations of products created by taking (without replacement) s terms from the set $\{\lambda_{i,M}, \lambda_{i,M-1}, \ldots, \lambda_{i,1}\}$ and $k-s$ terms from the set $\{c_M, c_{M-1}, \ldots, c_1\}$. This means that the number of terms in $\sigma_{i,s}$ becomes $\binom{M}{k-s}$. If $c_k = 1$ and λ_i is constant, such that F_k is constant for all k, the standard result from Norrlöf (2000b) follows. An upper bound for (4.B.61) can be formulated as

$$|\boldsymbol{F}_M \boldsymbol{F}_{M-1} \ldots \boldsymbol{F}_1 \boldsymbol{v}_{ijk}| \leq \max_{s=1,\ldots,M} |\lambda_{i,s}|^{M-k+1} \binom{M}{k-1} k \max_{p=1,\ldots,k} |\boldsymbol{v}_{ijp}| \tag{4.B.62}$$

where the triangular inequality is applied to the sum (4.B.61) and the coefficients of the vectors in the sum are taken as

$$\max_{s=1,\dots,M} |\lambda_{i,s}|^{M-k+1} \binom{M}{k-1}$$

which is an upper bound of the coefficients in (4.B.61). As an upper limit for the norm of the vectors in (4.B.61) the maximum value of the norm of all the vectors is used. The k in (4.B.62) comes from the fact that there are k terms in the sum in (4.B.61).

Now an upper limit for the mapping, independent of the choice of column in V, can be formulated based on the result in (4.B.62).

$$\begin{aligned} |\boldsymbol{F}_M \boldsymbol{F}_{M-1} \dots \boldsymbol{F}_1 \boldsymbol{v}_{ijk}| &\leq \rho_m^{M-n_m+1} \binom{M}{n_m-1} n_m \max_{\substack{i=1,\dots,l \\ j=1,\dots,m_i \\ p=1,\dots,n_{ij}}} |\boldsymbol{v}_{ijp}| \\ &= \rho_m^M \binom{M}{n_m-1} K_1 \end{aligned} \tag{4.B.63}$$

where ρ_m is defined according to (4.15) and

$$n_m = \max_{\substack{i=1,\dots,l \\ j=1,\dots,m_i}} n_{ij}$$

and $K_1 < \infty$ independent of M. The result in (4.B.63) gives the worst case upper bound and it is important to note that by choosing M big enough this limit can be made arbitrarily small. This follows form the fact that for fixed n_m the binomial coefficient is a polynomial in M which does not grow faster than the exponential term ρ_m^M.

Now, in the general case the following relation holds

$$\begin{aligned} |\boldsymbol{F}_M \boldsymbol{F}_{M-1} \dots \boldsymbol{F}_1 \boldsymbol{z}_0| &= |\boldsymbol{F}_M \boldsymbol{F}_{M-1} \dots \boldsymbol{F}_1 V V^{-1} \boldsymbol{z}_0| \\ &\leq n \rho_m^M \binom{M}{n_m-1} K_1 \max_{i=1,\dots,n} |(V^{-1}\boldsymbol{z}_0)_i| \\ &= \rho_m^M \binom{M}{n_m-1} K_2 \end{aligned} \tag{4.B.64}$$

where $K_2 < \infty$ and independent of M. The upper limit is found by replacing the vector $u = V^{-1}v$ with the vector $\tilde{u} = \mathbf{1} \max |u_i|$, i.e., a vector containing only the biggest component in u. This is why the n shows up after the first inequality. It is possible to move one step further and use the fact that the binomial term $\binom{M}{n_m-1}$ is a polynomial in M for fixed n_m. Each of the terms in the resulting polynomial is multiplied by ρ_m^M and using the result from Lemma 4.B.5 this can be written as

$$|\boldsymbol{F}_M \boldsymbol{F}_{M-1} \dots \boldsymbol{F}_1 \boldsymbol{z}_0| \leq \gamma \bar{\rho}^M |\boldsymbol{z}_0| \tag{4.B.65}$$

for some γ and $\rho_m < \bar{\rho} < 1$.

4.C Proof of Theorem 4.5

The proof is done in two steps. The first is to show uniform exponential stability and the second is to show that it exists a coordinate transformation such that the system with input is transformed into an exponentially stable system without input.

Step 1. From Definition 4.2 it follows that a linear iterative system,

$$\boldsymbol{z}_{k+1} = \boldsymbol{F}_k \boldsymbol{z}_k + \boldsymbol{F}_{r,k} \boldsymbol{r}$$

is uniformly exponentially stable if there exist a constant $\gamma > 0$ and a λ, $0 < \lambda \leq 1$ such that

$$\|\boldsymbol{z}_k\| \leq \gamma \lambda^{k-k_0} \|\boldsymbol{z}_0\|,\ k \geq k_0$$

Using Parseval's identity it follows that

$$\|\boldsymbol{z}_k\|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |Z_k(\omega)|^2 d\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} |F(e^{i\omega}) Z_0(\omega)|^2 d\omega \tag{4.C.66}$$

With

$$\bar{\rho} = \sup_{\omega \in [0,\pi]} \rho(F(e^{i\omega})$$

and Corollary 4.3 it is possible to find a bound for $|F(e^{i\omega})Z_0(\omega)|$,

$$|F(e^{i\omega}) Z_0(\omega)| \leq \gamma \bar{\rho}^k |Z_0(\omega)|, \quad \omega \in [0, \pi]$$

where γ is chosen such that the inequality holds for $\omega \in [0, \pi]$. From (4.C.66) it now follows that

$$\|\boldsymbol{z}_k\|^2 \leq \frac{1}{2\pi} \int_{-\pi}^{\pi} \gamma^2 \bar{\rho}^{2k} |Z_0(\omega)|^2 d\omega = \gamma^2 \bar{\rho}^{2k} \|\boldsymbol{z}_0\|^2$$

which exactly corresponds to Definition 4.2.

Step 2. Let,

$$\tilde{Z}_k(\omega) = Z_k(\omega) - (I - F(e^{i\omega}))^{-1} \bar{R}(\omega)$$

with $\bar{R}(\omega) = F_r(e^{i\omega}) R(\omega)$. Using (4.16) it follows that

$$\begin{aligned} \tilde{Z}_{k+1}(\omega) &= F(e^{i\omega})(\tilde{Z}_k(\omega) + (I - F(e^{i\omega}))^{-1} \bar{R}(\omega)) + \bar{R}(\omega) \\ &\quad - (I - F(e^{i\omega}))^{-1} \bar{R}(\omega) \\ &= F(e^{i\omega}) \tilde{Z}_k(\omega) \end{aligned}$$

From the result in step 1 above,

$$\|\tilde{\boldsymbol{z}}_k\| \leq \gamma \bar{\rho}^k \|\tilde{\boldsymbol{z}}_0\|$$

for all k and BIBO stability is shown.

Part II

The Application

5

Background

In this part of the thesis the application, industrial robots, chosen in the thesis for the evaluation of different ILC algorithms is described. Next chapter gives a general description of industrial robot modeling and control and in Chapter 7 the actual platform used in the experiments is presented more in detail. In this chapter a "soft" introduction to the two more technical chapters is given. This includes a motivation to why the robot application has been chosen and also a brief description on how the platform for the experiments has been developed in practice.

5.1 Why the Robot Application?

The most important reason to choose the robot application stems from the fact that this project has been carried out within NUTEK's competence center *ISIS (Information Systems for Industrial Control and Supervision)* at Linköping University. Within the competence center, research is performed in different areas, focusing on problems that are important for the industry. One of the companies that has joined as partner in ISIS is ABB Robotics and robotic applications are clearly the main interest for them. By using the industrial robots of ABB it was also possible to take advantage of the knowledge already gained from many years

of experience from robot design and robot control.

It should also be stressed that the most common applications for ILC have been, and still are, in the robotics area, see e.g., Arimoto et al. (1984a), Mita and Kato (1985), Arimoto (1985, 1991), Bondi et al. (1988), Poloni and Ulivi (1991), Burdet et al. (1997), Casalino and Bartolini (1984), Guglielmo and Sadegh (1996), Horowitz et al. (1991), Horowitz (1993), Jiang et al. (1999), and Lange and Hirzinger (1999). It is natural to view many of the production processes performed by industrial robots as iterative processes. An industrial robot often repeats the same task over and over again with a high repetition accuracy. This means that the same task is repeated in the same way with a very small deviation in the tracking error. If it was possible to compensate for systematic and repeatable errors this could be a great gain in many practical production problems. This leads naturally to the idea of applying ILC.

5.2 Using a Commercial System for Research

One of the great challenges of the work presented in the thesis is that it has been implemented in a commercial system. The modifications that have been done to the system can hence be made a part of the commercial software quite easily. For the company the main contribution, however, has probably been to show what actually can be done using ILC. The fact that the system is a commercial system puts also some limitations on what can be achieved, at least with a descent amount of work. In fact, the code used in the test platform has been developed over a period of more than three years with an effective time of production of about 3-5 months. Most of the work has been done by the author alone.

In order to implement the functions in the robot control system, needed to apply the ILC method, a complete understanding of the software that runs the control algorithms had to be achieved. This kind of knowledge is only found by discussing with people that have designed and written the code. If the people at the company had not had time for this it would have been impossible to complete the exercise and the project would most certainly have been a failure.

When using commercial code in a research program, a problem of security obviously arises. Sometimes the code in itself contains company secrets and it is therefore not allowed to be brought to the public. The solution that has been adopted in this project is that all the programming in the controller has been done at the site of the company. This has proved to be a very good solution and it has also shown to be a good way to transfer knowledge from the research community to the company as well as vice versa.

To reduce the amount of code needed in the commercial robot control system the ILC algorithms are implemented in MATLAB™ on a PC. This makes it also easy to try new ILC algorithms without rewriting the code of the controller.

6

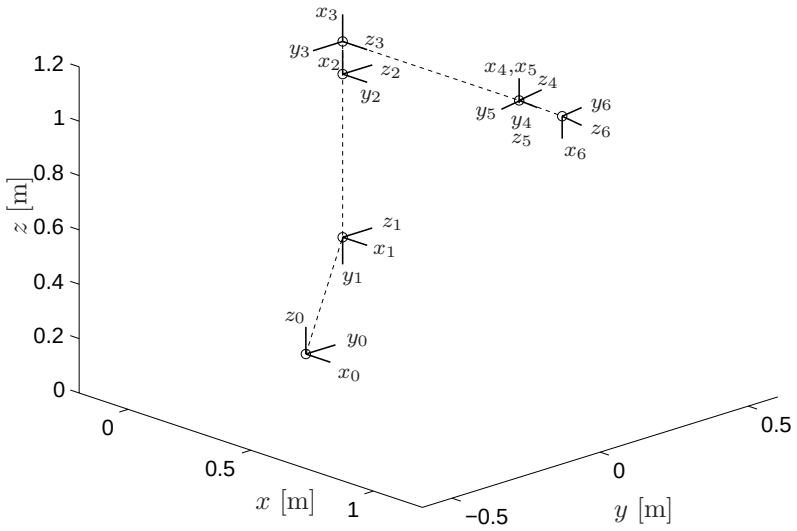
Industrial Robots

In this part of the thesis the application, industrial robots, used in the experiments will be presented. The presentation in this chapter includes robot modeling and control in general and a short description of the ABB industrial robot family. In the next chapter the robot used in the implementation of ILC is presented together with an overview of the implementation of the programs necessary to apply ILC to the commercial robot control system.

6.1 Introduction

In the thesis, the term *robot* refers to a mechanical arm, as shown in Figure 6.1(a). This is a classical industrial manipulator, in this case an ABB IRB 1400. In this chapter some of the important properties of this type of robot will be examined starting from the modeling aspect. Many survey articles and books have been written on the kinematics, dynamics and control of robots, see e.g., Spong and Vidyasagar (1989), Spong et al. (1992), and Craig (1988).

(a) A mechanical manipulator. In the thesis generally referred to as a robot.



(b) The frames associated with the joints of the IRB1400 using an algorithm from Spong and Vidyasagar (1989).

Figure 6.1 *Two different views of the robot.*

6.2 Modeling

The modeling of industrial robots is usually divided into kinematic and dynamic modeling. The first part that will be discussed here is the kinematics, i.e., the geometrical description of the manipulator. The material in this section is partly based on Norrlöf (1999) and Spong and Vidyasagar (1989).

6.2.1 Kinematics

Kinematics of a robot refers to the geometric relationship between the motion of the robot in joint space and the motion of the tool frame in the task space. In Figure 6.1(b) the two frames are represented by the coordinate systems $o_0x_0y_0z_0$ and $o_6x_6y_6z_6$ respectively.

The robot joint coordinates are given by a vector $\phi = (\phi_1, \phi_2 \dots, \phi_n)^T$. The joint angles, ϕ_i, are limited by the mechanical design of the robot. Cables mounted on the robot to serve for example a spot welding device can also give additional constraints.

A realization of the vector ϕ is called a *configuration* of the robot. Note that, for a *six degree-of-freedom (DOF)* robot, as the one in Figure 6.1, different configurations can give the same position and orientation of the tool frame with respect to the base frame. The position of the tool frame, the *tool center point (TCP)*, can be expressed as a coordinate point in the base frame coordinate system and it is represented by a vector $x \in \mathbb{R}^3$. The orientation of the tool frame is represented by an *orientation matrix*, $R \in \mathbb{R}^{3\times3}$. The TCP together with the orientation matrix is enough to describe the coordinate transformation from the base frame to the tool frame. The matrix R is orthogonal and satisfies $\det R = +1$. Even though R contains 9 elements, it describes only 3-DOF and can, accordingly, be represented with only 3 parameters, for example the Euler angles. The Euler angle representation will be denoted by $r \in \mathbb{R}^3$. The Euler angles specify the orientation of a coordinate frame, `frame1`, relative to another coordinate frame, `frame2`, by using three angles, (α, β, γ). The orientation of `frame2` relative to `frame1` is found by: First rotate about the z axis of `frame1` by the angle β. Next rotate about the current y axis by the angle α. Finally rotate about the current z axis by the angle γ.

Sometimes quaternions are used to represent orientation. A quaternion can be written as a mathematical object as,

$$q = q_1 + iq_2 + jq_3 + kq_4 \tag{6.1}$$

where $q_n \in \mathbb{R}$, $n \in [1, 4]$, and i, j, k are mutually orthogonal imaginary units having the property,

$$i^2 = j^2 = k^2 = ijk = -1, \quad ij = k, \quad jk = i, \quad ki = j \tag{6.2}$$

A quaternion can also, as suggested in Funda and Paul (1988), be written as a tuple, $q = [s, < x, y, z >] = [s, v]$. Where v is a vector with three elements. Obviously the quaternion is over-specified, having 4 elements. By putting the additional constraint that the size of the quaternion is equal to 1, this problem is however eliminated. More on quaternions and the computational aspects of quaternions can be found in, e.g., Funda and Paul (1988), Shoemake (1985), Funda et al. (1990), and Dobrovodsky (1994).

Position kinematics

A *kinematic description* refers to the geometric relationship between the motion of the robot in joint space and the motion of the tool frame in task space, usually defined in Cartesian coordinates. The description is without consideration of the forces needed to really perform the motion of the robot. The *forward kinematic problem* is to determine the mapping

$$X_0 = \begin{bmatrix} x(\phi) \\ r(\phi) \end{bmatrix} = f_0(\phi) \tag{6.3}$$

from joint space to task space. The *inverse kinematic problem* is to determine the inverse of this mapping, i.e., given a position and rotation of the tool frame calculate the corresponding robot joint configuration. As noted before, the inverse kinematic problem has many solutions while, for a serial link robot as in Figure 6.1(b), the forward kinematic problem has a unique solution. A systematic way of building the forward kinematic model is the Denavit-Hartenberg representation. In, for example Norrlöf (1999), an algorithm is presented to build forward kinematics model using the D-H representation. An example is also used to show how it actually works on the ABB IRB1400 robot. In Norrlöf (1999) the inverse kinematics problem is also discussed, using the ABB IRB1400 as an example.

Velocity kinematics

The *velocity kinematics* define the relationship between the joint velocities, $\dot{\phi}$, and the translational and the angular velocities of the tool frame. Similar to (6.3) the velocity kinematics can be written as

$$V = \begin{bmatrix} v \\ \omega \end{bmatrix} = J_0(\phi)\dot{\phi} \tag{6.4}$$

where $J_0(\phi) \in \mathbb{R}^{6\times n}$ is defined as the *manipulator Jacobian*, and V represents the linear and angular velocities of the tool frame. The vector $v \in \mathbb{R}^3$ is just the derivative with respect to time of the position vector $x(\phi)$ in (6.3) and the angular velocity is given by $\omega = (\omega_x, \omega_y, \omega_z)^T \in \mathbb{R}^3$. One way to calculate the velocity kinematics is to use the forward kinematic description in (6.3) and differentiate

with respect to time,

$$\dot{X}_1 = \frac{\partial f_0(\phi)}{\partial \phi}\dot{\phi} = J_1(\phi)\dot{\phi} \tag{6.5}$$

where the Jacobian is given by $J_1(\phi) = \frac{\partial}{\partial \phi} f_0(\phi)$. The points called *singular points* where the Jacobian looses rank are important because they can be interpreted as the points in the work space where a serial type robot looses one or more degrees of freedom. When planning a trajectory it is important to try to avoid passing through singular points.

6.2.2 Dynamics

The Euler-Lagrange equations are a tool from analytical mechanics that can be used to derive the equations of motion for a mechanical system. In this approach the joint variables, ϕ, are considered as generalized coordinates. The *kinetic energy* of the manipulator can be calculated as,

$$K(\phi, \dot{\phi}) = \frac{1}{2}\dot{\phi}^T D(\phi)\dot{\phi} \tag{6.6}$$

where $D(\phi) > 0$ is the *inertia matrix*. Let $P : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function, called the *potential energy*. For a rigid robot, the potential energy is due to gravity only. For a flexible robot the potential energy also stems from the elasticity. Now, define the *Lagrangian* function according to

$$\mathcal{L}(\phi, \dot{\phi}) = K(\phi, \dot{\phi}) - P(\phi) \tag{6.7}$$

The dynamics of the manipulator are described by Lagrange's equations

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\phi}_k} - \frac{\partial \mathcal{L}}{\partial \phi_k} = \tau_k,\ k = 1, \dots, n \tag{6.8}$$

where $\tau_1, \dots, \tau_n$ represent generalized input forces. Inserting the kinetic energy and the potential energy for the Lagrangian $\mathcal{L}$ above leads to the matrix description,

$$D(\phi)\ddot{\phi} + C(\phi, \dot{\phi})\dot{\phi} + g(\phi) = \tau \tag{6.9}$$

where $D(\phi) > 0$, $D(\phi) = D^T(\phi)$ is the inertia matrix, $C(\phi, \dot{\phi})\dot{\phi}$ is generally referred to as the velocity dependent term, containing the centrifugal and Coriolis effects, and $g(\phi)$ is the gravitational term.

There are some important properties of the Lagrangian dynamics of (6.9) that are helpful in the analysis and design of the manipulator control system. Among these properties are (from Spong and Vidyasagar (1989)):

1. The inertia matrix $D(\phi)$ is positive definite and symmetric and there exist scalars such that

   $$\mu_1(\phi)I \leq D(\phi) \leq \mu_2(\phi)I \tag{6.10}$$

   If all joints are revolute, then μ_1 and μ_2 are constants.

2. The matrix $W(\phi, \dot{\phi}) = \dot{D}(\phi) - 2C(\phi, \dot{\phi})$ is skew symmetric.

3. The mapping $\tau \rightarrow \dot{\phi}$ is passive, i.e., there exists $\beta \geq 0$ such that

$$\int_0^T \dot{\phi}^T(u)\tau(u)dt \geq -\beta \tag{6.11}$$

4. Rigid robot manipulators are fully actuated. This means that there is an independent control input for each degree-of-freedom. Robots that have joint or link flexibilities are no longer fully actuated and the control problem is in general more difficult.

5. The equations of motion given in (6.9) are linear in the inertia parameters.

All these properties have been used in different proofs concerning, for example, stability and convergence of adaptive and robust controllers for robots.

6.2.3 High level control

In the early days of robot development the manipulators were mainly used for set-point tracking. This means that the robots were programmed to move to a certain point but that the trajectory they followed in order to do so were not well defined. The controllers that were used in these robots were of PD-type or sometimes PID-type. This remarkably simple controller structure shows good results for the set-point tracking problem and it can also be shown that the PD controller actually makes the system globally asymptotically stable. A proof is given in, e.g., Slotine and Li (1991).

High level planning and control

In general, the motion control problem of manipulators is divided into three stages,

- motion planning,
- trajectory generation, and
- trajectory tracking.

These steps are now going to be described briefly.

Motion planning

The motion planning level includes activities such as checking if the robot can move from one point to another without hitting an obstacle, or determining how

the work should be performed on a specific work object. In industrial applications this part is mainly done by a system separate from the robot control system. The motion planning is performed by the operator or by a computer program, e.g., a CAD/CAM tool. In the car industry, CAD programs that can simulate all the different levels of the motion control of the robots are used. The simulations also give very accurate predictions for the cycle time. The car manufacturers use such tools to, off-line, optimize the motion of the robot in order to reduce the cycle time and the production time. When the production rate is several units per minute, a reduction of the cycle time of only a 10th of a second will imply a considerable gain in production.

Trajectory generation

The trajectory generation problem is the problem of generating trajectories with position, speed, and acceleration given as functions of time. The problem also includes the consideration of the actual robot dynamics and kinematics since the trajectories must be feasible, i.e., the manipulator must be able to follow the trajectories that are generated. In many cases it is also a question of finding the optimal paths where the maximum speed and acceleration is used. The goal in many applications is to perform a given task in the least possible time and in this sense the trajectory planning problem is very important. Of course the motion planning gives some restrictions on what can be achieved.

The trajectory generation is often made in a chain of steps where, in a first step, a few points are planned on the trajectory and in the next steps the trajectory is refined and more points are created. Finally, the trajectory is well enough defined to be used for control, i.e., the trajectory tracking algorithms. The motion planning and also the trajectory generation are often made in the task space. To use the reference for control the trajectory must be transformed into configuration space, i.e., joint space.

Trajectory tracking

Since in most applications it is not possible to measure the actual joint positions, the motor positions are used to control the robot. This implies that the coordinates in the joint space must be scaled in order to compensate for the gear ratios. If the gear incorporates known flexibilities, those can also be compensated for in this step. The trajectory tracking problem can be defined as the problem of controlling the robot arms in such a way that the tool frame follows the trajectory calculated by the trajectory generator. The trajectory is defined both by the tool frame position and the tool frame orientation relative to the base frame.

A typical architecture for solving the robot control problem is shown in Figure 6.2, for more details on the planning and trajectory generation problems see e.g., Spong and Vidyasagar (1989).

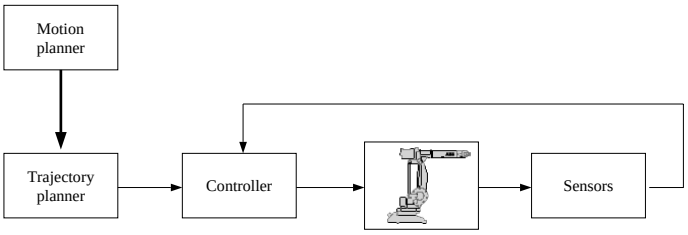


Figure 6.2 *Block diagram showing the components in the robot control problem.*

6.2.4 Control design

Next some methods to design the controller in a robot control system will be discussed.

Feedback linearization

Feedback linearization is a relatively recent tool in nonlinear control design. Because of the rapid development of microprocessor technology it has also been possible to apply this technique to real control systems. The idea of feedback linearization is to algebraically transform a nonlinear dynamic system into a linear one. In practice this is done by using nonlinear coordinate transformation and nonlinear feedback. In the robotics context the feedback linearization technique is known as *inverse dynamics*. The idea is,

1. to compensate all the coupling nonlinearities in the Lagrangian dynamics and
2. to design a linear compensator based on a linear decoupled plant.

The feedback linearization can be made in the joint space coordinates or in the task space coordinates. An example of the first type will now be presented.

Given a plant model

$$M(\phi)\ddot{\phi} + C(\phi, \dot{\phi})\dot{\phi} + g(\phi) = \tau \tag{6.12}$$

as shown in (6.9). Here the inertia of the motor and gear has been included in the inertia matrix,

$$M(\phi) = D(\phi) + A(\phi) \tag{6.13}$$

$A(\phi)$ is a diagonal matrix with elements $a_i r_i^2$; a_i is the i^{th} actuator inertia and r_i is the gear ratio. The following control law is used

$$\tau = M(\phi)a_\phi + C(\phi, \dot{\phi})\dot{q} + g(\phi) \tag{6.14}$$

where $a_\phi \in \mathbb{R}^n$ is an intermediate control input. Since the inertia matrix, $M(\phi)$, is positive definite and therefore invertible for all ϕ, the closed loop system reduces to the decoupled double integrator

$$\ddot{\phi} = a_\phi \tag{6.15}$$

If a reference trajectory, $r(t) = \phi_d(t)$, is assumed to be given, then one possible choice of a_ϕ is

$$a_\phi = \ddot{\phi}_d + K_d(\dot{\phi}_d - \dot{\phi}) + K_p(\phi_d - \phi) \tag{6.16}$$

i.e., a PD controller with a feedforward of the reference acceleration. Combining (6.15) and (6.16), using

$$\tilde{\phi} = \phi_d - \phi \tag{6.17}$$

the result becomes

$$\ddot{\tilde{\phi}} + K_d\dot{\tilde{\phi}} + K_p\tilde{\phi} = 0 \tag{6.18}$$

In fact, by choosing the parameters K_d and K_p it is possible to place the poles of the error characteristic equation, (6.18), arbitrarily. In the real system, restrictions will of course be posed on the control signal. This will limit also the possible bandwidth of the system and, hence, put more restrictions on the parameters K_d and K_p.

This method of controlling the robot makes it possible to have two different levels of control. For example, the inner loop controller can take care of the linearization, creating a closed loop system that gives the impression of being a linear system from the reference to the output. The outer loop controller, giving the overall control system the desired properties, can therefore be designed using linear techniques. For ILC this is also an advantage since there are methods to design the ILC algorithms for the linear case but it is more difficult in the nonlinear case. To be able to do the linearization it is necessary to have full knowledge of the system dynamics. In practice this is, of course, not plausible. Possible solutions to this problem are to introduce adaptive or robust controllers (Craig, 1988; Slotine and Li, 1991) which will be discussed next, or to use ILC.

Robust and adaptive control

As was discussed previously, using the feedback linearization method gives raise to new problems. The model of the system must be very accurate and in many applications there are parameters that can not be specified in advance, e.g., the load parameters. Robust and adaptive methods have the advantage that they can incorporate this kind of uncertainty in the design process and give good results also for this case. Robust and adaptive controllers differ on one important point. The adaptive algorithm uses some kind of online parameter estimation method

to cope with the changes in the system parameters. The robust methods, on the other hand, take care of the parameter uncertainties already in the controller design process, i.e., before the controller is actually applied to the system.

Robust feedback linearization

Many different techniques from linear and nonlinear control theory have been applied to the problem of robust feedback linearization for manipulators. Among these are

- sliding modes,
- Lyapunov's second method,
- method of stable factorization.

Consider again the dynamic equations for the n-link manipulator,

$$M_0(\phi)\ddot{\phi} + C_0(\phi,\dot{\phi})\dot{\phi} + g_0(\phi) = \tau \tag{6.19}$$

with the control input given by

$$\tau = M(\phi)a_\phi + C(\phi,\dot{\phi})\dot{\phi} + g(\phi) \tag{6.20}$$

where M, C, and g represent the nominal values of the true system M_0, C_0, and g_0. Now a model error, $\tilde{(\cdot)} = (\cdot)_0 - (\cdot)$, is introduced, indicating that exact feedback linearization can not be achieved in reality. The term a_ϕ may be used to compensate for the resulting perturbation terms. Let

$$a_\phi = \ddot{\phi}_d + K_d(\dot{\phi}_d - \dot{\phi}) + K_p(\phi_d - \phi) - \delta_a \tag{6.21}$$

where δ_a is an extra compensation to be chosen, and substitute (6.20) and (6.21) into (6.19). After some algebra the following expression is obtained,

$$\ddot{\tilde{\phi}} + K_d\dot{\tilde{\phi}} + K_p\tilde{\phi} = \delta_a + \eta(\phi,\dot{\phi},\delta_a,t) \tag{6.22}$$

where

$$\eta = M_0^{-1}\big(\tilde{M}(\ddot{\phi}_d + K_d\dot{\tilde{\phi}} + K_p\tilde{\phi} - \delta_a) + \tilde{C}\dot{\phi} + \tilde{g}\big) \tag{6.23}$$

The result of (6.22) can be formulated as a linear state-space description

$$\dot{x} = Ax + B(\delta_a + \eta) \tag{6.24}$$

where

$$x = \begin{bmatrix}\tilde{\phi}\\ \dot{\tilde{\phi}}\end{bmatrix},\ A = \begin{bmatrix}0 & I\\ -K_p & -K_d\end{bmatrix},\ B = \begin{bmatrix}0\\ I\end{bmatrix} \tag{6.25}$$

The goal is now to find a time-varying scalar bound, $\rho(x,t) \geq 0$, on the uncertainty η,

$$\|\eta\| \leq \rho(x,t) \tag{6.26}$$

and to design the additional input term, δ_a, such that the state trajectory in (6.24) is bounded or, if possible, converges to zero. It is in general difficult to calculate ρ in (6.26) since the term η is a complex expression involving also the additional control term δ_a. One approach that has been tried is to design δ_a using sliding mode theory (Spong and Vidyasagar, 1989). The simplest sliding mode controller results from choosing $\delta_{a,i}$ according to

$$\delta_{a,i} = \rho_i(x,t)\text{sign}(s_i),\ i = 1, \ldots, n \tag{6.27}$$

where ρ_i is the bound on the i-th component of η, $s_i = \dot{\tilde{q}} + \lambda_i \tilde{q}_i$ represents a sliding surface in the state-space and sign$(\cdot)$ is the sign function.

An alternative approach is the so-called theory of guaranteed stability of uncertain systems, based on Lyapunov's second method. The matrix A in (6.24) is Hurwitz, which means that $\forall Q > 0,\ Q = Q^T \implies \exists P > 0$ with $P = P^T$, satisfying the Lyapunov equation,

$$A^T P + PA = -Q \tag{6.28}$$

This follows from the Lyapunov stability theory for LTI systems, see e.g., Slotine and Li (1991). Using the matrix P, the term δ_a can be chosen as

$$\delta_a = \begin{cases} -\rho(x,t)\frac{B^T P x}{\|B^T P x\|} & \text{if } \|B^T P x\| \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.29}$$

With the Lyapunov function $V = x^T P x$ it is now possible to show that $\dot{V}$ is negative definite along solution trajectories of the system given by (6.24).

In practice these two approaches will lead to chattering, i.e., small but fast changes in the state x. This is due to the fact that the switching can not be made with nonzero delay. This chattering is undesirable because it involves high control activity and it may also excite high frequency dynamics neglected in the modeling of the system. Many refinements and extensions to the two approaches to robust feedback linearization have been proposed. Mainly the idea is to simplify the calculations of the uncertainty bounds, $\rho(x,t)$, and to smooth the chattering in the control signal (Spong et al., 1992).

The method of *stable factorizations* has also been applied to the robust feedback linearization problem. This approach will not be covered here, the reader is referred to Spong et al. (1992) with references.

Adaptive Feedback Linearization

The work on adaptive control of robot manipulators can be divided into two phases according to Spong et al. (1992), the *approximation* phase (1979 - 1985) and the

linear parameterization phase (1986-present). In the approximation phase the assumptions are that the robot dynamics can be linearized, that decoupling is possible for the joints, and that the inertia matrix varies slowly. The breakthrough for adaptive feedback linearization came about 1985 when it became widely known that the manipulator dynamics could be written as a linear parameterization. Consider again the dynamics, described by (6.19), but suppose that the parameters in (6.20) are not fixed. Instead, assume that they are time varying estimates of the true parameters. Let

$$a_\phi = \ddot{\phi}_d + K_d(\dot{\phi}_d - \dot{\phi}) + K_p(\phi_d - \phi) \tag{6.30}$$

Substitute (6.20) and (6.30) into (6.19). After some algebra the following expression is obtained,

$$\ddot{\tilde{\phi}} + K_d\dot{\tilde{\phi}} + K_p\tilde{\phi} = \widehat{M}^{-1}\varphi(\phi, \dot{\phi}, \ddot{\phi})\tilde{\theta} \tag{6.31}$$

where φ is a regressor, and $\tilde{\theta} = \hat{\theta} - \theta$, where $\hat{\theta}$ is a parameter vector estimate. The system in (6.31) can now be written as

$$\dot{x} = Ax + B\Phi\tilde{\theta} \tag{6.32}$$

where

$$x = \begin{bmatrix} \tilde{\phi} \\ \dot{\tilde{\phi}} \end{bmatrix},\ A = \begin{bmatrix} 0 & I \\ -K_p & -K_d \end{bmatrix},\ B = \begin{bmatrix} 0 \\ I \end{bmatrix},\ \Phi = \widehat{M}^{-1}\varphi(\phi, \dot{\phi}, \ddot{\phi}) \tag{6.33}$$

The PD controller parameters, K_p and K_d, are chosen such that the matrix A is Hurwitz. Suppose that the output, $y = Cx$, of (6.32), is such that the transfer function $C(pI - A)^{-1}B$ is strictly positive real (SPR). It then follows from the Kalman-Yakubovich lemma that there exist positive definite matrices P and Q such that

$$A^T P + PA = -Q \tag{6.34a}$$

$$B^T P = C \tag{6.34b}$$

If the parameter updating law is chosen as

$$\dot{\hat{\theta}} = -\Gamma^{-1}\Phi^T C x \tag{6.35}$$

where Γ is symmetric and positive definite, the global convergence to zero of the tracking error with all internal signals remaining bounded can be shown using the Lyapunov function

$$V = x^T P x + \frac{1}{2}\tilde{\theta}^T\Gamma\tilde{\theta} \tag{6.36}$$

This approach has some drawbacks.

- The parameter updating law uses the acceleration, $\ddot{\phi}$, as a known parameter, which is usually very noisy.

- The estimated inertia matrix, $\widehat{M}$, must be invertible. This can, however, be made possible by using projection in the parameter space.

Later work has been devoted to overcome these drawbacks and it has been proved possible by using so-called *indirect* approaches based on a filtered prediction errors Slotine and Li (1991).

Aspects on robot design and robot control

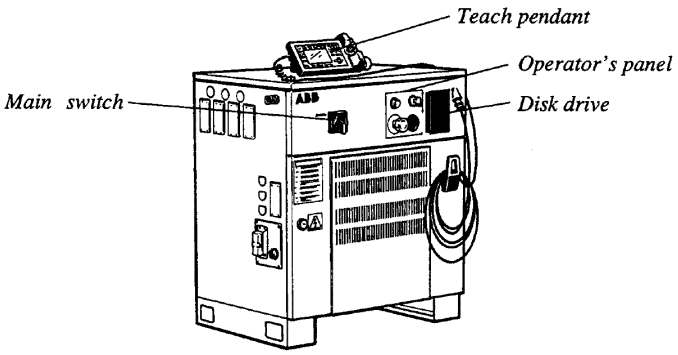
The structure of the manipulator has a great impact on the difficulty of the control problem. By using gears, for example, it is possible to reduce the nonlinear coupling between the axes of the robot. This, of course, to the price of friction, backlash, and flexibilities. If the position is measured on the motor side, i.e., before the gears, the static measurement accuracy will be increased r times by the gears, where r is the gear ratio. Because of the flexibilities introduced by the gears it is not so sure, however, that the dynamic accuracy is increased in the same way.

Today the gears contribute to a significant part of the cost when producing a robot. Cutting this cost by using less expensive gears would be preferable. A goal for the control designers should be to reach the same or better performance with cheaper gears having more flexibilities and more friction. ILC can be one concept that makes it possible to use cheaper robot components. More advanced control strategies, including for example adaptive control, could be another possibility to reach the same goal.

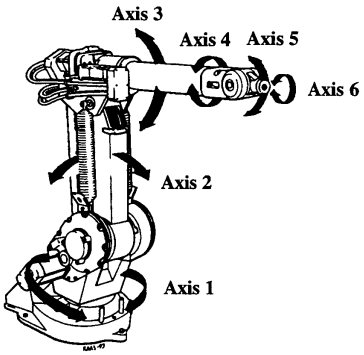
6.3 The ABB IRB Family

After the general description of robot manipulators and robot control the focus will now be on the type of robot that is used in the experiments. The ABB robot system used in this work consists of a controller, version *S4C*, and the manipulator, an IRB1400. In Figure 6.3 the two components are depicted.

Figure 6.3(b) shows that the manipulator has 6-DOF. Note also that axes 1, 2 and 3 are all controlled by motors placed on the lower part of the robot. Motors for axes 4, 5 and 6 are mounted on the rear part of the upper arm, next to joint 3. This makes the robot lighter and more balanced, and hence it can move faster with less motor torque. Before describing the robot in more detail some background on ABB robots in general is given.



(a) The controller, ABB S4C.



(b) The manipulator, ABB IRB 1400.

Figure 6.3 *The two main components in the robot system.*

6.3.1 Background

The first generation of robots from Asea/ABB was presented on the market in 1974 (Nilsson, 1996). The controller family was called S1 and the robots available were the IRB-6 and the IRB-60. The next generation, called S2, was presented in 1982 together with some new manipulators, e.g., the spot welding robot IRB-90. In 1986, S3, the third generation controllers and a new family of manipulators were presented. During 1994, S4, the fourth generation of the control system, was launched. The main contributions in S4 were the new programming language,

RAPID™, and a model-based motion control strategy. The language, RAPID™, is used by the operator when programming the robot.

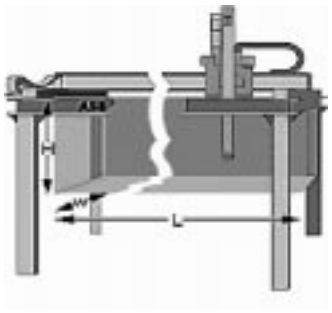


Figure 6.4 *An example of a gantry robot.*

Since 1994 a lot of new robots have been presented on the market and today ABB has a full range of traditional industrial robots for different applications, ranging in size from the IRB140, reaching about 0.8 m, to the IRB6400, reaching up to 3 m from the foot of the robot. ABB Robotics has also a family of gantry robots and some more specialized manipulators like the IRB640 flex palletizer having only 4-DOF. A gantry robot is a robot that is mounted on a track as displayed in Figure 6.4 making it possible for the robot to move in a plane parallel with the floor. A typical application for a gantry robot is a pick and place operation, e.g., moving a heavy object from one production line to another. ABB Robotics has also a high speed pick and place robot, the IRB340, using the DELTA robot structure (Burdet et al., 1997).

As the number of applications grows the demands on the controller grow. The controller is today the same for all the different robots and this is possible by using a highly parameterized system where almost all the functions can be controlled by parameters in a database implemented in the robot system.

Each individual robot has its unique database file. This makes it possible to adjust and compensate for small differences, caused by, e.g., the inaccuracy in the production of the mechanical manipulator.

6.3.2 The controller, S4C

The controller, depicted in Figure 6.3(a), consists of the cabinet and the teach pendant. Inside the cabinet, the hardware that runs the control program is found. The main computer that takes care of the high level control is a Motorola 68060 processor in the generation of the cabinet used in the experiments. The low level control is performed using a DSP from Texas Instrument. In the cabinet the two computers are physically separated on two different boards, called the main

computer board and the robot computer board. There is also a separate board for the memory used by the computers. The controller used in the experiments has 16Mb of RAM. An optional board for the Ethernet communication is also installed in the system. It is also possible to connect devices for digital and analogue I/O to the robot control system and the cabinet is prepared for standard bus communication with other equipment such as PLCs.

In the standard configuration the cabinet is equipped with 4 or 6 drive units, depending upon the number of DOF of the robot. Sometimes a cell is equipped with external axes, controlled by the cabinet. For this reason it is possible to control up to a total of 12 axes from inside the cabinet. This could be for example a 6-DOF robot plus three 2-DOF robots, moving work objects. A common configuration in arc welding applications is to use one extra 2-DOF robot, giving a total of 8-DOF in the cell.

Figure 6.5 *The teach pendant.*

In Figure 6.5 the device that the operator uses when programming the robot is depicted. The device is called the *teach pendant* and is equipped with a joystick having 3-DOF. Using the joystick it is possible to control

- the position of the tool in a Cartesian coordinate system in, e.g., the base frame, or
- the orientation of the tool, or
- the individual axes of the robot.

It is also possible to control external axes using the teach pendant. On the teach pendant there is a display and a keyboard that makes it possible for the operator to program and run the robot while being in the working cell close to the manipulator.

6.3.3 The manipulator, IRB1400

The manipulator used in the experiments is an IRB1400. It is a 6-DOF manipulator with the structure of the joints according to Figure 6.3(b). The motors are of AC type and all the motors that drive the axes 1 to 3 are placed on the base of the robot. The motors for the wrist, axes 4 to 6, are placed on the back of the upper arm and the torques are transmitted via a transmission in the upper arm to joints 4, 5 and 6. In Figure 6.6 the measures of the arms are depicted to give an idea of the size of the manipulator. Note the springs that are mounted in parallel with

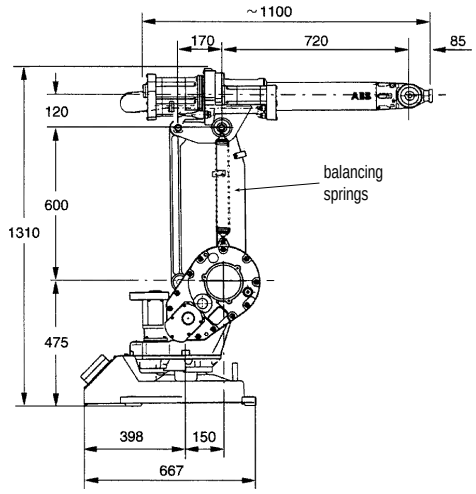


Figure 6.6 *To give an idea of the size of the manipulator.*

the lower arm, rotating around joint 2 (cf. Figure 6.3(b)). The springs are used to balance the robot and to decrease the static load of the motor for joint 2.

The resolvers measuring the joint angles of the robot are mounted on the motor axis and, hence, the position measured is the motor position. The goal is of course to control the arm position and ultimately the tool position and orientation. On the IRB1400 the flexibilities are not as notable as they are on the larger robots but it is still a problem that has to be dealt with. Using a dynamical model and the TrueMove™ function the ABB robots are very accurate. For the IRB1400 the positional repeatability is ± 0.05 mm, maximum speed for the TCP is 2.1 m/s and the maximum acceleration is 15 m/s^2. It is obvious that these measures are not valid for the robot in the whole working range. Changing the robot configuration will mean changing the performance of the robot considerably.

7

Platform for the Experiments

After making a general discussion of industrial robot modeling and control as well as an overview of the ABB IRB family in the previous chapter, it is now time to describe the platform used in the experiments. This includes the different components that are necessary in order to apply ILC to a commercial system. A short note on the possible future extensions to the software is also given. The chapter is concluded with a discussion on the methods used in the modeling process of the system.

7.1 The Implementation

The implementation is now going to be presented from a system theoretic point of view. In Figure 7.1 an abstract view of the implementation is depicted. The two functions *data-logger* and *data-input* are shown as dashed arrows going in to and out of the system, respectively. The system inside the bold rectangle is the robot control system which, in this case, is not possible to change. The data-input function makes it however possible to add a control signal $\{u_k(t)\}_0^{t_f}$ to the system. This makes it possible to evaluate ILC on the commercial control system. Next the components that make it possible to apply ILC to the commercial robot control system are described in more detail.

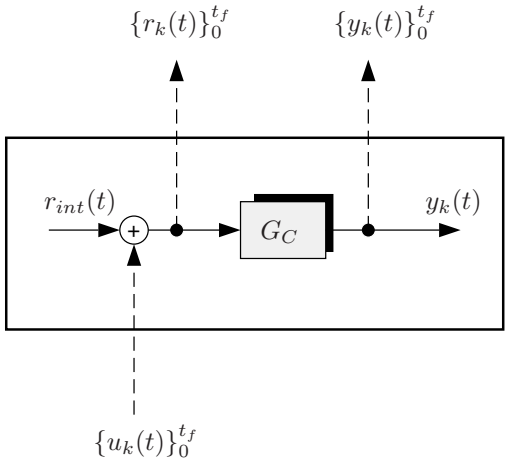


Figure 7.1 *Abstract view of the implementation.*

7.1.1 The interface

The interface means here the connection between MATLAB™ and the robot control system. This includes the high level synchronization of the ILC and the robot control system. The data exchange between the robot controller and the PC, the dashed arrows in Figure 7.1, is made using the file system. Since the two systems run completely in parallel some kind of synchronization is necessary. A standard solution when dealing with concurrent systems is to use *semaphores* (Stallings, 1992).

The two systems can be represented as two processes that have to be synchronized. The first one is the execution of the iteration or movement of the robot, called *process A*. The second is the calculation of the compensation signal in the PC using MATLAB™, this process is called *process B*. Since the data are not available until the file is completely written to the disk, process B has to wait until process A is completely finished before the processing can start, as shown in Figure 7.2.

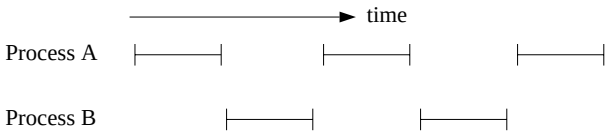


Figure 7.2 *The execution of the processes A and B.*

To show how the problem is solved using semaphores two primitives are introduced, `wait(semaphore)`, and `signal(semaphore)`. The semaphore can be seen as an

```
Process A:                          Process B:
  repeat                              repeat
    wait(semA);                         wait(semB);
    perform_programmed_motion;          load_data;
    log_data;                           process_data;
    save_logged_data;                   save_processed_data;
    signal(semB);                       signal(semA);
  until readyA                        until readyB
```

Figure 7.3 *The resulting processes A and B, using the semaphore for mutual exclusion.*

integer value that can be initialized to an arbitrary value. When doing a `wait` the semaphore value is decreased by one and if the value becomes negative the process that executed the `wait` statement is blocked. The `signal` function increases the value of the semaphore by one. If the value becomes positive when applying the `signal` statement the process waiting is unblocked. Using the semaphores to synchronize the processes A and B will result in the code shown in Figure 7.3.

A problem, also with only two processes, is to decide which one of the two processes that will start running. This problem is taken care of by an initializing procedure that creates the semaphores in a way that makes the execution of the two processes deterministic. In the case with the processes A and B, the initialization procedure runs in the PC and executes the following steps,

1. create `semA` and `semB` having the value 0,
2. write an empty control signal, u_0, to the disk, and
3. execute `signal(semA)`.

In this way the robot will first move, i.e., process A will execute before process B. After process A has finished the new correction term can be calculated, based on the error when u_0 was applied.

In the test environment, the implementation of the semaphores is made using files. The two primitives above correspond to creation and deletion of the files `semA` and `semB`. The expression `wait(semA)` means, wait until there is a file named `semA`, delete this file, then continue. The primitive `signal(semA)` creates the file `semA` and then the execution continues. This implementation uses a *binary semaphore* because the semaphore can only take the values 0 or 1 (Stallings, 1992), file does not exist and file exists, respectively. For this application, however, this is enough and when the interface changes to another kind of communication media, e.g., a direct link between the processes, the synchronization can easily be replaced by a

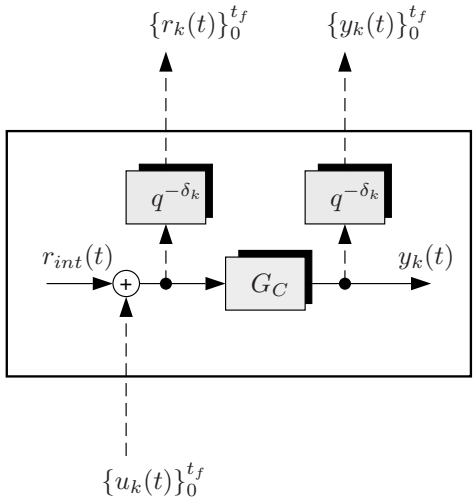


Figure 7.4 *The actual function of the implementation.*

new implementation. This makes the chosen solution of the synchronization and mutual exclusion very general.

7.1.2 The robot controller

In the previous section the synchronization of the terminal and the robot control system was discussed. Now, consider again the system depicted in Figure 7.1. In practice it is necessary to consider the internal synchronization of the data-input and the data-logger function. In the system depicted in Figure 7.4 this problem is shown. The synchronization problem can be seen as the problem of identifying the delay, δ_k, that is present in the data-logger.

In practice this is solved by logging an additional synchronization signal that can be used in the identification of the delay δ_k. Comparing the synchronization signal from different experiments makes is possible to keep track of how the delay δ_k evolves. The difficult part is to get an accurate estimate of δ_0. For this a "back door" in the system is used which enables the possibility to, simultaneously, add the compensation signal $u(t)$ and log the signal $r_{int}(t)$. If $u(t)$ is chosen as zero, r_{int} and r (logged by the data-logger) will coincide when compensated for the delay δ_k. This technique is used in the first iteration only and then the other technique, mentioned above, with additional synchronization signals is used.

7.1.3 The terminal

The program execution of the robot is controlled from the terminal. By altering the programs or creating new programs using, e.g., MATLAB™, it is possible to control the motion of the robot from a process running on the terminal. The synchronization of the processes on the terminal and the robot controller is made using semaphores, as described in Section 7.1.1.

The data-logger and the data-input function on the robot uses the hard-disk of the terminal for data storage. In MATLAB™ two functions, `readlog` and `writelog`, are implemented for reading/writing data to the data-logger and the data-input function respectively.

7.1.4 Summary

By using the different components described in this section, ILC can be evaluated on the robot. The interface makes it possible to communicate and synchronize the robot control system and the PC. This is a condition for the ILC system to work. The data-logger and the data-input function are, of course, also vital for the ILC method. In order to evaluate ILC on the robot system two new functions have been implemented, the data-input function and the synchronization between the data-logger and the data-input function. The ILC algorithm is implemented in MATLAB™. Using these functions, an environment is created where it is possible to try and compare different ILC updating formulas.

7.2 Future extensions to the software

As has been understood from the discussion in the previous sections there are some crucial points in the implementation that can be improved. These points will now be summarized and a pointer towards what should be done in the implementation to significantly improve the functionality is also given.

7.2.1 The interface

Today the file system is used as the interface between the robot system and the terminal. It would be a major improvement to connect the two systems directly by using a link with the TCP/IP protocol. This kind of interface will make the communication between the robot and the terminal much faster. In the current implementation it takes a couple of seconds before the system reacts after that a signal is sent to the semaphore. The transfer time of data from the data-logger to the terminal and to the data-input function from the terminal should also be much decreased.

7.2.2 The robot controller

There are some problems with the software in the robot controller that have to be understood and solved in order to make the ILC method completely reliable in the system. The, sometimes random, delays should be removed and a better solution for the synchronization of the data-input function and the data-logger has to be found.

7.2.3 The terminal

The major extension and improvement needed for the software in the terminal is a more user friendly man-machine interface. In the current system the test environment is run using a terminal program on the PC. A direct link from MATLAB™ to the robot controller would make it possible to control the robot directly from MATLAB™. The graphical user interface support in MATLAB™ would make it easier, also for beginners, to use the advanced functions that are available in the data-input and the data-logger functions. It should, of course, also be possible to use other functions in the test environment.

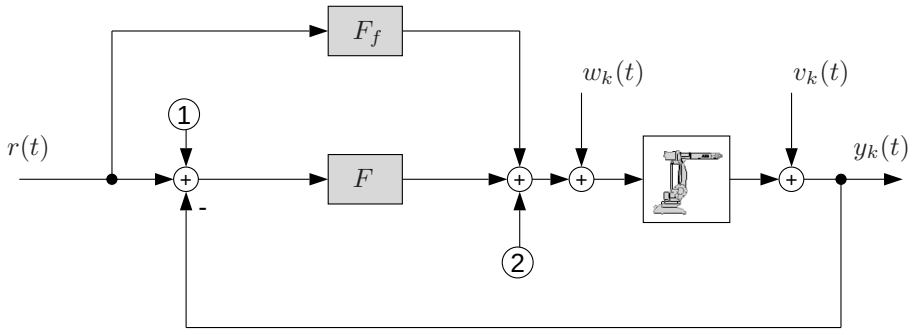


Figure 7.5 *A simplified view of the robot system used in the experiment.*

7.3 Modeling and Identification

Now the identification experiments that have been performed on the robot will be discussed. The system model that is used is shown in Figure 7.5. Next this model will be presented and a parameterized version will be identified.

7.3.1 A simplified view of the robot control system

In Figure 7.5 a simplified view of the robot control system is shown, with the blocks representing the robot and the controllers F_f and F, respectively. The robot is a non-linear system, having 6 inputs and 6 outputs. The feedforward controller, F_f, takes care of the decoupling and linearization of the system and also most of the actual control of the system. Since there are always model errors a feedback controller, F, is used to take care of the resulting control error. The actual implementation of the controllers in the robot is confidential but in principle the control scheme of Figure 7.5 gives an idea of how it works.

The input to the system in Figure 7.5 is the motor reference position $r(t)$, this signal is calculated from the arm reference position using a model based approach. How this works in principle was discussed in the previous chapter. Using the calculated motor position reference a feedforward torque signal is calculated and the resulting control error is taken care of by the feedback controller.

The disturbances $d_k(t)$ and $n_k(t)$ are load disturbances and measurement disturbances respectively. In the robot system the two types of disturbances are caused by, e.g., ripple in the torque applied to the motors and ripple of the resolver signals.

7.3.2 Applying ILC

Using the data-input function, implemented in the robot control system, the ILC input signal can only be applied at the points marked ① and ② in Figure 7.5, i.e., at the position reference and at the torque reference to the robot. In the thesis the ILC input signal is applied at ① using the error defined by

$$e_k(t) = r(t) - y_k(t) \tag{7.1}$$

With the data-logger it is possible to sample the reference signal, $r(t)$, and the motor position signal, $y_k(t)$, cf. Figure 7.5. With the definition of the error signal as in (7.1) the ILC input signal can be calculated using, e.g., the updating formula

$$u_{k+1}(t) = Q(q)(u_k(t) + L(q)e_k(t)) \tag{7.2}$$

cf. Chapter 3. Of course it is possible to implement more general ILC schemes (as will be shown in the following chapters of the thesis).

If it is assumed that the closed loop system is completely decoupled and also linear, it is enough to consider only the SISO case. Since the ILC algorithms covered here only consider the control errors on the motor side the pre-filter of the reference signal will not have to be considered in the analysis. Using Theorem 4.8 with

$$T_u(q) = G_{yu_k}(q) = \frac{F(q)G(q)}{1 + F(q)G(q)} \tag{7.3a}$$

$$T_r(q) = (F_f(q)F^{-1}(q) + 1)G_{yu_k}(q) \tag{7.3b}$$

where $G(q)$ is assumed to be a linearized model of a robot joint, a sufficient condition for stability becomes

$$|1 - L(e^{i\omega})T_u(e^{i\omega})| < |Q^{-1}(e^{i\omega})|, \ \forall\, \omega \tag{7.4}$$

Note that this criterion only contains the closed loop system, which is designed to be well behaved.

7.3.3 Identification

In order to identify T_u a signal $u_k(t)$ is applied at the point ①. From the schematic view of the robot control system in Figure 7.5 it is clear that feedforward of the torque is used in the system. To identify the closed loop system seen from point ① it is necessary to do an experiment where the effect of the feedforward is removed.

The experiment described here is performed as an identification experiment on axis 1 of the IRB 1400 industrial robot. To see only the effect of the input, $u(t)$, on the output it is necessary to run two experiments. The first is with the input $u_1(t)$ equal to zero and the second with $u_2(t)$ as shown in Figure 7.6(a). The input $u_2(t)$ is taken from an ILC experiment. There are systematic ways to create input signals for identification experiments (Ljung, 1987), but the discussion here is focused on showing how the identification process can be performed on the system considered in the thesis.

By taking the difference of the output in the second and the first experiment it is possible to find the resulting output from applying the input $u_2(t)$ alone (since $u_1(t) = 0$). This means that

$$\Delta y(t) = y_2(t) - y_1(t) = T_u(q)u_2(t) \tag{7.5}$$

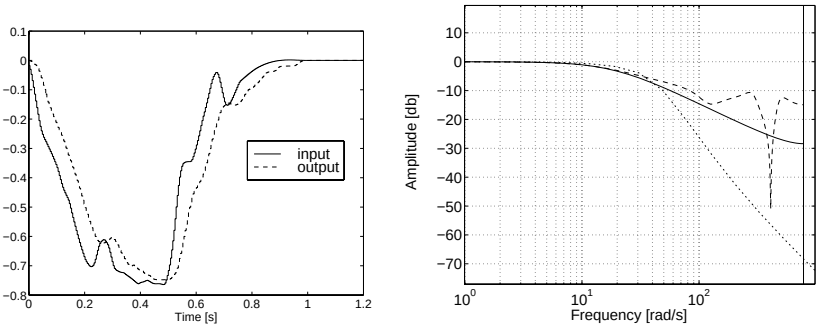
where it is now left to choose the model structure of the system $T_u(q)$. The tool used in the modeling is the *System Identification Toolbox* (Ljung, 1995) for MATLAB™ which provides a model from the data shown in Figure 7.6(a).

The identification procedure is not presented in detail here, the interested reader is referred to e.g., Ljung (1987). A few words about the model structure will however be given. The model structure used here is an ARX model structure,

$$\begin{aligned} y(t) + a_1 y(t-1) + \ldots + a_{n_a} y(t-n_a) \\ = b_1 u(t-n_k) + \ldots + b_{n_b} u(t-n_b-n_k+1) + e(t) \end{aligned} \tag{7.6}$$

In the identification process, one design choice is the number of parameters that should be estimated, n_a and n_b. There is also a parameter n_k indicating the time delay in the system. From the set of possible models we have chosen two. One with many parameters and one with just a few parameters. The first model is an ARX model having, $n_a = 1$, $n_b = 1$, and $n_k = 1$ and the model is given by

$$\hat{T}_u(q) = \frac{0.07q^{-1}}{1 - 0.93q^{-1}} \tag{7.7}$$



(a) Input signal, $u_2(t)$. and the output, $\Delta y(t)$.

(b) Amplitude plots of the models obtained. ARX na=1, nb=1, and nk=1 (solid). ARX na=8, nb=8, and nk=1 (dashed). Model based on physical knowledge of the system (dotted).

Figure 7.6 *The data used to build the model of the closed loop system from reference position to motor position and the resulting models shown in the frequency domain.*

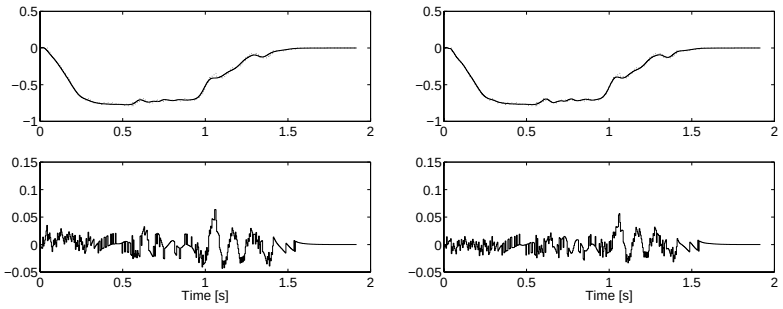
In Figure 7.6(b) the amplitude plot of the model is shown together with a model calculated based on a stiff robot model G and the controller parameters. A high order ARX model with $n_a = 8$, $n_b = 8$, and $n_k = 1$, is also estimated and in Figure 7.6(b) the amplitude curve for this model is shown. The actual parameters for the high order model are not given here.

7.3.4 Model validation

The amplitude plots of the different models in Figure 7.6(b) give a validation of the models in the frequency domain. The calculated physical model is based on nominal parameters and a rigid body model for the joint of the robot. This is of course a simplification. The conclusion from the validation in the frequency domain is that the bandwidth of the estimated closed loop system corresponds to the one achieved using the continuous time model and the knowledge of the controller parameters.

A validation of the model is also carried out using another set of data from the system. The actual output from the system is compared to the output that is simulated by the ARX models. In Figure 7.7 the result from this validation is

shown and it can be seen that the simulation of the models nearly exactly gives the same behavior of the output as the real system. The very fast switching that can be seen in the prediction error is caused by quantization of the measured motor angles.



(a) Model 1. *(b)* Model 2.

Figure 7.7 *Validation of the two models using validation data from the system. Above: predicted value (solid), measured value (dotted). Below: prediction error.*

Part III

Classical ILC

8

Design Strategies with Examples

The interpretation of *classical ILC* might differ among researchers but the interpretation that will be adopted here is that classical ILC is first order ILC with iteration independent filters. This means that the updating equation for the ILC can be written according to (4.22), i.e.,

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}(\boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{e}_k)$$

with the updating equation in (3.13) as a special case.

In this chapter some design strategies for the classical ILC will be presented. Examples will also be given where the applicability of the design schemes are shown in simulations as well as in experiments performed on the industrial robot described in Part II.

8.1 Introduction

The ILC synthesis problem is to find the ILC algorithm given a system to control and a specification of the wanted behavior. It is a fact that in the ILC literature the synthesis of ILC updating formulas has not been addressed very much. The focus has instead been on the analysis of stability.

Among the works that actually have been done on the design, an important part has been on synthesis based on linear quadratic optimal ILC. Within this framework Professor Owens' group has made contributions in e.g., Amann and Owens (1994) and Amann et al. (1995b, 1997), while other contributions are for example Lee and Lee (1998) and Lee et al. (2000). An early contribution in this direction is also Togai and Yamano (1985). This kind of approach will be discussed later on in this chapter. ILC synthesis based on H_∞ methods has been covered in contributions by, e.g., Park and Hesketh (1993), Liang and Looze (1993), and de Roover (1996a,b). The work in Park and Hesketh (1993) does not involve the Q filter in the ILC, but this is included in the work of de Roover (1996a,b), and also to some extent in Liang and Looze (1993). In the next section some explicit formulations of design schemes for ILC will be presented.

8.2 Algorithms for ILC Synthesis

To design a stable and efficient ILC algorithm it is necessary to have a model of the system that will be controlled. The level of detail of the model will differ between the different design algorithms but it is always true that it is necessary to have some knowledge of the system to make the design. The knowledge might be replaced by experiments where the ILC algorithm is adjusted according to the result from the experiments. Obviously the data from the experiments can be used also for modeling of the process and therefore this kind of ILC method will not be considered here.

8.2.1 A heuristic approach

The first design algorithm uses a system model to check that the stability criterion is fulfilled. The knowledge about the system could also be reduced to only the time delay of the system and, to be sure of the stability, the size of the first Markov parameter of the controlled system. This might however give very poor performance. The algorithm is here called "heuristic" but sometimes it has also been referred to as P-type in the literature.

Algorithm 8.1 (A heuristic design procedure)

1. Choose the Q filter as a low-pass filter with cut-off frequency such that the band-width of the learning algorithm is sufficient.
2. Let $L(q) = \kappa q^\delta$. Choose κ and δ such that the stability criterion, formulated in the frequency domain $|1 - L(e^{i\omega})T_u(e^{i\omega})| < |Q^{-1}(e^{i\omega})|$, is fulfilled. Normally it is sufficient to choose δ as the time delay and $0 < \kappa \leq 1$ to get a stable ILC system.

If the necessary and sufficient condition for stability is used instead, i.e., δ is chosen as the time delay and κ is chosen such that κ times the first Markov parameter is less than 1, then the algorithm will result in a stable ILC scheme. In the next example it is shown that the performance might not be so good if there is an uncertainty in the time delay of the system.

Example 8.1 Example 2.1 revisited

Assume that the system $T_u(q) = G_C(q)$ is given by (2.7) and the filter $L(q)$ is chosen according to (2.8) and (2.9). The first choice corresponds to the true system while the second choice comes as a result when the system model is incorrect and it is assumed that there is no delay in the system. In Example 2.1 the bandwidth of the ILC algorithm is not limited since $Q(q) = 1$. In order to fulfill the stability condition for the given system, κ has to be chosen such that

$$|1 - \kappa 0.09516| < 1$$

in the first case. This means that κ has to be chosen in the interval

$$0 < \kappa < 21$$

In the second case the necessary and sufficient stability condition in Corollary 4.6 is not fulfilled since $\rho(I - \boldsymbol{T}_u) = 1$. This implies that convergence will not be achieved for any κ. ❑

If the delay in the system is over-estimated instead of under-estimated as in the example above, then it is not true that with $Q = 1$ the system will be guaranteed to be stable. In Figure 8.1 the absolute value of the eigenvalues of $I - \boldsymbol{L}\boldsymbol{T}_u$ are shown for different $\boldsymbol{L}$. When $L(q) = q$ the matrix $I - \boldsymbol{L}\boldsymbol{T}_u$ is lower triangular and the diagonal elements are constant and equal to $1 - 0.09516$. For $L(q) = q^2$ the diagonal elements are still constant but the matrix is no longer lower triangular and therefore the eigenvalues are no longer constant. Moreover, it is clear from Figure 8.1 that the stability condition is not met since $\rho(I - \boldsymbol{L}\boldsymbol{T}_u) > 1$. This shows how important it is to have a correct estimate of the delay of the system.

To have stability in the example discussed above, when $\kappa = 1$ and $Q = 1$, it is necessary to have an exact value of the delay. By using the Q-filter it is however possible to get a convergent ILC scheme also for $L(q) = q^2$. In Figure 8.1 the dotted curve corresponds to the eigenvalues of $\boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_u)$ when $Q(q)$ is a zero phase low-pass filter with cut-off frequency 0.3 of the Nyquist frequency. Obviously the convergence criterion is now satisfied. The price paid for this is that the convergence will no longer be guaranteed to be to a zero error, cf. Theorem 4.9.

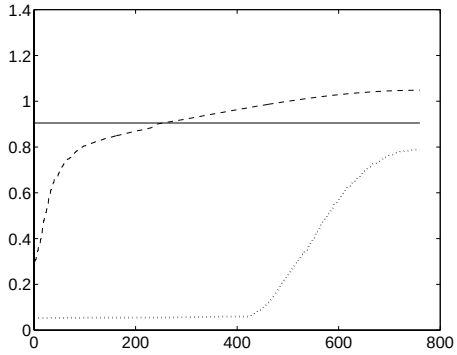


Figure 8.1 *Absolute value of the eigenvalues (number of eigenvalues on the x-axis) of* $\boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_u)$ *when* $Q(q) = 1,\ L(q) = q$ *(solid),* $Q(q) = 1,\ L(q) = q^2$ *(dashed), and when* $\boldsymbol{Q}$ *is a zero phase low-pass filter and* $L(q) = q^2$ *(dotted).*

8.2.2 A model-based approach

The design procedure presented in this section has also been discussed in Gunnarsson and Norrlöf (1997a,c) and Norrlöf (1998). The idea is similar to the approach by de Roover (1996a,b) but he uses a model matching approach based on H_∞ methods while here an algebraic approach is used. The transfer function $T_u(e^{i\omega})$ is the transfer function from the ILC input, $u_k(t)$, to the output of the system, $y_k(t)$.

Algorithm 8.2 (A model-based design procedure)

1. Build a model of the relation between the ILC input and the resulting correction on the output, i.e., find a model $\widehat{T}_u$ of T_u.
2. Choose a filter $H_B(q)$ such that it represents the desired convergence rate for each frequency. Normally this means a high-pass filter.
3. Calculate L by $L(q) = \widehat{T}_u^{-1}(q)(1 - H_B(q))$.
4. Choose the filter $Q(e^{i\omega})$ as a low-pass filter with cut-off frequency such that the band-width of the resulting ILC is high enough and the desired robustness is achieved.

To explain the use of the filter $H_B(q)$ in step 2, consider Lemma 4.4. When $Q(q) = 1$ and the disturbances are neglected the updating equation for the error becomes,

$$\epsilon_{k+1}(t) = (1 - T_u(q)L(q))\epsilon_k(t)$$

Clearly the choice of $H_B(q)$ will decide the nominal convergence rate for the error. In the frequency domain the filter H_B can be adjusted to give, e.g., a slower but more robust convergence for some frequencies. The choice of H_B must be realizable. It is clearly not possible to choose H_B small for frequencies where the model is very uncertain since this will most likely lead to a divergent behavior of the resulting ILC. The choice of H_B has, therefore, also to include robustness considerations, although robustness is also achieved with the Q-filter.

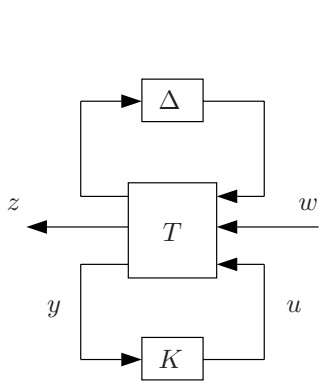
The resulting L-filter might have an unnecessary high degree, therefore it can be possible to make a model reduction of L using some model reduction technique, e.g., balanced truncation.

Next a method that uses H_∞ control theory to systematically design the filters Q and L is reviewed.

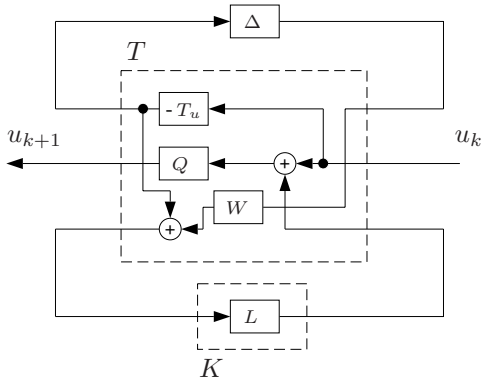
8.2.3 An approach based on μ-synthesis

The design process for the filters Q and L presented here is a review of the algorithm in de Roover (1996a,b). The notation is however changed to fit the framework used in the thesis. The algorithm is based on optimal H_∞ control theory (Zhou et al., 1996).

Before formulating the algorithm some prerequisites will be discussed. The standard plant format, depicted in Figure 8.2(a) is used in the discussion. Within this framework, tools (for example in MATLAB™ (Balas et al., 1994)) are available for



(a) Standard plant.



(b) Standard plant interpretation for the ILC synthesis problem.

Figure 8.2 *Standard plant description for μ-synthesis.*

computing a stabilizing K that minimizes $\|T_{zw}\|_\infty$. It is assumed that the real system can be described by a nominal model T_u^0, having an output multiplicative uncertainty. The uncertainty is described in the frequency domain by a stable and inversely stable weighting function $W(e^{i\omega})$. The true system is found in the following set of systems,

$$T_u(q) = \{(I + W(q)\Delta(q))T_u^0(q) \mid \|\Delta(q)\|_\infty < 1\} \tag{8.1}$$

Now the ILC design problem can be formulated in the μ-synthesis framework using the representation shown in Figure 8.2(b). Given

$$K = L(q) \quad \text{and} \quad T = \begin{bmatrix} 0 & -T_u(q) & 0 \\ 0 & Q(q) & Q(q) \\ W(q) & -T_u(q) & 0 \end{bmatrix} \tag{8.2}$$

the optimization problem can be solved using μ-synthesis. The solution is based on the *D-K iteration* technique, the reader is referred to e.g., Zhou et al. (1996), for a thorough discussion. It is important to note that the D-K iterations do not necessarily need to converge to the optimum. Many applications have, although, shown that the technique works (Balas et al., 1994). The proposed algorithm is now presented.

Algorithm 8.3 (A μ-synthesis based approach)

1. Build a model $\widehat{T}_u$ of the transfer function T_u together with an upper bound on the model uncertainty, according to (8.1).
2. Choose the Q-filter as a low-pass weighting filter with cut-off frequency ω_c.
3. For given $\widehat{T}_u$ and Q, find a filter L that minimizes $\|T_{zw}\|_\infty$ using μ-synthesis (Balas et al., 1994).
4. If a filter L can be found such that $\|T_{zw}\| < 1$, increase the bandwidth of the filter Q and perform step 3 again; else decrease ω_c. Perform step 2 and 3 repeatedly until the maximum of ω_c is reached.

The algorithm has been tested in de Roover (1996a) on an $xy\phi$-stage, which is a type of high accuracy positioning mechanism. The model with uncertainty, according to (8.1), is identified using data from the plant and two different ILC updating formulas are evaluated. One based on the nominal model, designed according to Algorithm 8.2, and one using the robust approach in Algorithm 8.3. Not surprisingly, the performance is better with the ILC algorithm based on the nominal model compared to the robust algorithm. The reader is referred to de Roover (1996a,b) and the references there for more details.

8.2.4 A DFT based approach

In Manabe and Miyazaki (1994), a method to update the ILC signal, u_k, in the frequency domain is presented. The idea is to use the DFT of the output and the input to the plant. Using the ETFE, i.e., the ratio between the DFT:s of the output and the input signals, a (local) model is identified in each iteration. The inverse of the ETFE is then used in the updating of the learning control signal. This corresponds to the ideas presented in Algorithm 8.2 were the L filter is chosen to be the inverse of the transfer function from u_k to y_k.

8.2.5 Design based on optimization

Previous contributions to the optimization based approach to ILC can be found in e.g., Gorinevsky et al. (1995), Phan (1998), Lee and Lee (1998), and Amann et al. (1995a,b). Some approaches based on ideas from unconstrained optimization and minimization techniques are presented by Togai and Yamano (1985). For a general discussion on unconstrained minimization see, e.g., the book by J.E. Dennis and Schnabel (1983). The material presented in this section is based on Gunnarsson and Norrlöf (1999a,b).

Algorithm derivation

Assume that the system is in matrix form as described by (3.2), i.e.,

$$\boldsymbol{z}_k = \boldsymbol{T}_r \boldsymbol{r} + \boldsymbol{T}_u \boldsymbol{u}_k + \boldsymbol{T}_w \boldsymbol{w}_k \tag{8.3a}$$

$$\boldsymbol{y}_k = \boldsymbol{z}_k + \boldsymbol{T}_v \boldsymbol{v}_k \tag{8.3b}$$

where it is also assumed that the system description is the same at each iteration (cf. (3.2)). Let the quadratic criterion be formulated according to

$$J_{k+1} = \boldsymbol{e}_{k+1}^T \boldsymbol{W}_e \boldsymbol{e}_{k+1} + \boldsymbol{u}_{k+1} \boldsymbol{W}_u \boldsymbol{u}_{k+1}$$

where $\boldsymbol{e}_{k+1} = \boldsymbol{r} - \boldsymbol{y}_{k+1}$. The idea is to determine $\boldsymbol{u}_{k+1}$ in such a way that the error $\boldsymbol{e}_{k+1}$ becomes as small as possible with respect to the criterion. The weighting matrices decide the trade off between performance and input energy and the matrices can be used for both frequency as well as time weighting. The criterion is minimized subject to the constraint

$$(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k)^T (\boldsymbol{u}_{k+1} - \boldsymbol{u}_k) \leq \delta$$

Introducing the Lagrange multiplier yields the criterion

$$\bar{J}_{k+1} = \boldsymbol{e}_{k+1}^T \boldsymbol{W}_e \boldsymbol{e}_{k+1} + \boldsymbol{u}_{k+1}^T \boldsymbol{W}_u \boldsymbol{u}_{k+1} + \lambda((\boldsymbol{u}_{k+1} - \boldsymbol{u}_k)^T (\boldsymbol{u}_{k+1} - \boldsymbol{u}_k) - \delta) \tag{8.4}$$

From (8.3) it follows that $\boldsymbol{e}_{k+1}$ is given by

$$\boldsymbol{e}_{k+1} = (I - \boldsymbol{T}_r)\boldsymbol{r} - \boldsymbol{T}_u \boldsymbol{u}_{k+1} - \boldsymbol{T}_w \boldsymbol{w}_{k+1}$$

Using this result together with (8.4) makes it possible to do a straightforward differentiation of $\bar{J}_{k+1}$ with respect to $\boldsymbol{u}_{k+1}$. This gives

$$-\boldsymbol{T}_u^T \boldsymbol{W}_e \boldsymbol{e}_{k+1} + \boldsymbol{W}_u \boldsymbol{u}_{k+1} + \lambda(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k) = 0 \tag{8.5}$$

where the optimum is achieved when the derivative equals zero. Using a statistical description of $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ i.e., the system and the measurement disturbances, it is possible to use a prediction of the error in the next iteration as in e.g., (Lee and Lee, 1998). This possibility will however not be utilized here and the disturbances are instead predicted using the corresponding mean values, which are assumed to be zero. The error $\boldsymbol{e}_{k+1}$ is hence represented as

$$\hat{\boldsymbol{e}}_{k+1} = (I - \widehat{\boldsymbol{T}}_r)\boldsymbol{r} - \widehat{\boldsymbol{T}}_u \boldsymbol{u}_{k+1} \tag{8.6}$$

where $\widehat{\boldsymbol{T}}_r$ and $\widehat{\boldsymbol{T}}_u$ denote a nominal model of the closed loop system and the transfer function from the ILC input to the output respectively. This implies that some a priori knowledge of the system to be controlled is available. Using (8.6) in (8.5) gives

$$-\widehat{\boldsymbol{T}}_u^T \boldsymbol{W}_e (I - \widehat{\boldsymbol{T}}_r)\boldsymbol{r} + \widehat{\boldsymbol{T}}_u^T \boldsymbol{W}_e \widehat{\boldsymbol{T}}_u \boldsymbol{u}_{k+1} + \boldsymbol{W}_u \boldsymbol{u}_{k+1} + \lambda(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k) = 0 \tag{8.7}$$

which implies

$$\boldsymbol{u}_{k+1} = (\boldsymbol{W}_u + \lambda \cdot I + \widehat{\boldsymbol{T}}_u^T \boldsymbol{W}_e \widehat{\boldsymbol{T}}_u)^{-1} (\lambda \boldsymbol{u}_k + \widehat{\boldsymbol{T}}_u^T \boldsymbol{W}_e (I - \widehat{\boldsymbol{T}}_r)\boldsymbol{r}) \tag{8.8}$$

This equation does not contain any measurements from the system but from (8.8) it is possible to establish the convergence criterion for the proposed method.

Theorem 8.1
If $\lambda > 0$ and the nominal model corresponds to the true system then the proposed optimization based ILC algorithm always gives a stable ILC system.

Proof A sufficient condition for stability, according to Theorem 4.8, is that the maximum singular value of

$$(\boldsymbol{W}_u + \lambda \cdot I + \widehat{\boldsymbol{T}}_u^T \boldsymbol{W}_e \widehat{\boldsymbol{T}}_u)^{-1} \lambda I \tag{8.9}$$

is less than one. Since $\boldsymbol{W}_u + \widehat{\boldsymbol{T}}_u^T \boldsymbol{W}_e \widehat{\boldsymbol{T}}_u$ is symmetric and positive definite it is possible to write (8.9) according to

$$(\boldsymbol{V}(\lambda I + \boldsymbol{\Sigma})\boldsymbol{V}^T)^{-1} = \boldsymbol{V}(\lambda I + \boldsymbol{\Sigma})^{-1}\boldsymbol{V}^T$$

where $\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is the SVD of $\boldsymbol{W}_u + \widehat{\boldsymbol{T}}_u^T \boldsymbol{W}_e \widehat{\boldsymbol{T}}_u$. Let σ_i be the elements of the diagonal of Σ, i.e., the corresponding singular values. The singular values of (8.9) become

$$\frac{\lambda}{\lambda + \sigma_i} < 1$$

since $\sigma_i > 0$ from the positive definiteness mentioned above. ■

From (8.6) it follows that

$$(I - \widehat{\boldsymbol{T}}_r)\boldsymbol{r} = \hat{\boldsymbol{e}}_k + \widehat{\boldsymbol{T}}_u \boldsymbol{u}_k$$

This means that the result in (8.8) can be reformulated into

$$\boldsymbol{u}_{k+1} = (\boldsymbol{W}_u + \lambda \cdot I + \widehat{\boldsymbol{T}}_u^T \boldsymbol{W}_e \widehat{\boldsymbol{T}}_u)^{-1}((\lambda \cdot I + \widehat{\boldsymbol{T}}_u^T \boldsymbol{W}_e \widehat{\boldsymbol{T}}_u)\boldsymbol{u}_k + \widehat{\boldsymbol{T}}_u^T \boldsymbol{W}_e \hat{\boldsymbol{e}}_k) \quad (8.10)$$

i.e.,

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}(\boldsymbol{u}_k + \boldsymbol{L}\hat{\boldsymbol{e}}_k) \quad (8.11)$$

where

$$\boldsymbol{Q} = (\boldsymbol{W}_u + \lambda \cdot I + \widehat{\boldsymbol{T}}_u^T \boldsymbol{W}_e \widehat{\boldsymbol{T}}_u)^{-1}(\lambda \cdot I + \widehat{\boldsymbol{T}}_u^T \boldsymbol{W}_e \widehat{\boldsymbol{T}}_u) \quad (8.12)$$

and

$$\boldsymbol{L} = (\lambda \cdot I + \widehat{\boldsymbol{T}}_u^T \boldsymbol{W}_e \widehat{\boldsymbol{T}}_u)^{-1} \widehat{\boldsymbol{T}}_u^T \boldsymbol{W}_e \quad (8.13)$$

The updating matrices $\boldsymbol{Q}$ and $\boldsymbol{L}$ hence depend on the nominal model $\widehat{\boldsymbol{T}}_u$ and the weighting matrices $\boldsymbol{W}_u$ and $\boldsymbol{W}_e$. The Lagrange multiplier λ is not computed explicitly but instead used as a design variable.

In equation (8.11) the error signal is formed using the nominal model of the system, while in real use the actual error signal from the system is used. The normal definition of the error $\boldsymbol{e}_k = \boldsymbol{r} - \boldsymbol{y}_k$ leads to

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}(\boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{e}_k) \quad (8.14)$$

where $\boldsymbol{Q}$ and $\boldsymbol{L}$ are given by (8.12) and (8.13).

Before discussing design aspects for the optimization based ILC design some remarks will be given to some trivial choices of weighting matrices and Lagrange multipliers.

First, if the penalty on the input signal in (8.12) is set to zero, $\boldsymbol{W}_u = 0$, this implies that $\boldsymbol{Q} = I$. The result is an updating equation without any weighting of the previous input signal. The same effect is achieved by minimizing a criterion of the type

$$J = \boldsymbol{e}_{k+1}^T \boldsymbol{e}_{k+1} + \rho(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k)^T(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k) \quad (8.15)$$

as studied in e.g., Amann and Owens (1994) and Amann et al. (1995b). As will be seen below, the use of $\boldsymbol{W}_u > 0$ i.e., $\boldsymbol{Q} \neq I$ is useful, both for robustness reasons as well as when dealing with non-minimum phase systems.

Second, by instead putting $\lambda = 0$, i.e., removing the constraint on the updating step, the computation of $\boldsymbol{u}_{k+1}$ can be expressed as

$$\boldsymbol{u}_{k+1} = (\boldsymbol{W}_u + \widehat{\boldsymbol{T}}_u^T \boldsymbol{W}_e \widehat{\boldsymbol{T}}_u)^{-1} \widehat{\boldsymbol{T}}_u^T \boldsymbol{W}_e (I - \widehat{\boldsymbol{T}}_r)\boldsymbol{r} \quad (8.16)$$

This means that the computation of the input vector becomes a one-step procedure which relates more to a feed-forward control law than to ILC.

The third and last remark is that choosing both $\boldsymbol{W}_u = 0$ and $\lambda = 0$ results in the choice (assuming that the inverse exists)

$$\boldsymbol{u}_{k+1} = \widehat{\boldsymbol{T}}_u^{-1}(I - \widehat{\boldsymbol{T}}_r)\boldsymbol{r}$$

This corresponds to the choice $L(q) = T_u^{-1}(q)$ in the classical ILC updating equation,

$$u_{k+1}(t) = Q(q)(u_k(t) + L(q)e_k(t))$$

which is also discussed in Section 4.2. Now the actual design algorithm can be formulated.

Algorithm 8.4 (Optimization based ILC design)

1. Build a model of the relation between the ILC input and the resulting correction on the output, i.e., find a model $\widehat{\boldsymbol{T}}_u$ of $\boldsymbol{T}_u$.
2. Choose the weights $\boldsymbol{W}_e$ and $\boldsymbol{W}_u$ and the Lagrange multiplier λ in the criterion.
3. Calculate the matrices $\boldsymbol{Q}$ and $\boldsymbol{L}$ according to (8.12) and (8.13).
4. Use the ILC updating equation according to (8.14) with $\boldsymbol{u}_0$ for example chosen as $\boldsymbol{u}_0 = 0$.

In the next sections it will be discussed more on the different design parameters and how different choices effect the resulting ILC system.

Non-minimum phase systems

The ultimate goal in ILC is, when it is possible, to generate a signal which is the reference signal filtered through the inverse of the system, it is obvious that there will be problems when the system has some zeros outside the unit circle. Now it will be shown how the optimization based ILC algorithm deals with this problem but first to a general discussion on non-minimum phase zeros.

In, e.g., Åström and Wittenmark (1984) the following result can be found. Consider an n:th order continuous time system $G(s)$ with m zeros $z_i, i = 1, \ldots, m$. Assuming that zero order hold is applied, the corresponding discrete time system $G_T(q)$ will have the following properties as T tends to zero:

- The discrete time system $G_T(q)$ will have $n - 1$ zeros.

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

This means that for a continuous time system with relative degree larger than one the discrete time system will, for short sampling intervals, have zeros on or outside the stability boundary.

It will now be shown how the design variables in the optimization approach can be used to handle this situation. Recall the updating equation (8.14)

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}(\boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{e}_k) = \boldsymbol{Q}\boldsymbol{u}_k + \boldsymbol{Q}\boldsymbol{L}\boldsymbol{e}_k$$

with $\boldsymbol{Q}$ and $\boldsymbol{L}$ given by equations (8.12) and (8.13) respectively. The asymptotic input vector (with respect to k) is given by

$$\boldsymbol{u}_\infty = (I - \boldsymbol{Q} + \boldsymbol{Q}\boldsymbol{L}\boldsymbol{T}_u)^{-1}\boldsymbol{Q}\boldsymbol{L}(I - \boldsymbol{T}_r)\boldsymbol{r} \tag{8.17}$$

in the general case, and

$$\boldsymbol{u}_\infty = (\boldsymbol{W}_u + \boldsymbol{T}_u^T\boldsymbol{W}_e\boldsymbol{T}_u)^{-1}\boldsymbol{T}_u^T\boldsymbol{W}_e(I - \boldsymbol{T}_r)\boldsymbol{r} \tag{8.18}$$

in the optimization case. In (8.18) it has been assumed that the nominal model equals the true system, i.e., $\widehat{\boldsymbol{T}}_u = \boldsymbol{T}_u$. The properties of the algorithm will be studied by investigating the 2-norm, i.e., the largest singular value, of the matrices $\boldsymbol{Q}$, $\boldsymbol{Q}\boldsymbol{L}$ and $(\boldsymbol{W}_u + \boldsymbol{T}_u^T\boldsymbol{W}_e\boldsymbol{T}_u)^{-1}\boldsymbol{T}_u^T\boldsymbol{W}_e$. The discussion is furthermore confined to the case $\boldsymbol{W}_e = I$ and $\boldsymbol{W}_u = \rho \cdot I$. The following result can then be stated.

Theorem 8.2
Consider the matrices $\boldsymbol{Q}$ and $\boldsymbol{Q}\boldsymbol{L}$ where $\boldsymbol{Q}$ and $\boldsymbol{L}$ are given by equations (8.12) *and* (8.13) *in the situation $\boldsymbol{W}_e = I$ and $\boldsymbol{W}_u = \rho \cdot I$, where $\rho > 0$. Then*

$$\|\boldsymbol{Q}\|_2 < 1$$

$$\|\boldsymbol{Q}\boldsymbol{L}\|_2 \leq \frac{1}{2 \cdot \sqrt{\rho + \lambda}}$$

and

$$\|(\rho \cdot I + \boldsymbol{T}_u^T\boldsymbol{T}_u)^{-1}\boldsymbol{T}_u^T\|_2 \leq \frac{1}{2 \cdot \sqrt{\rho}}$$

Proof Introduce the SVD of the matrix $\boldsymbol{T}_u$

$$\boldsymbol{T}_u = \bar{\boldsymbol{U}}\boldsymbol{\Sigma}\bar{\boldsymbol{V}}^T$$

where the singular values of $\boldsymbol{T}_u$, i.e., the diagonal element of $\boldsymbol{\Sigma}$, are denoted by σ_i. Introducing this representation in the definitions of $\boldsymbol{Q}$ and $\boldsymbol{L}$ together with the assumptions for $\boldsymbol{W}_u$ and $\boldsymbol{W}_e$ gives

$$\boldsymbol{Q} = \bar{\boldsymbol{V}}^T \boldsymbol{\Sigma}^Q \bar{\boldsymbol{V}}$$

where the diagonal elements of $\boldsymbol{\Sigma}^Q$ are given by

$$\sigma_i^Q = \frac{\sigma_i^2 + \lambda}{\sigma_i^2 + \lambda + \rho}$$

This is a monotonous function of σ_i asymptotically tending to one. Similar calculations give that

$$\boldsymbol{QL} = \bar{\boldsymbol{V}} \boldsymbol{\Sigma}^{QL} \bar{\boldsymbol{U}}^T$$

where the diagonal elements σ_i^{QL} are given by

$$\sigma_i^{QL} = \frac{\sigma_i}{\rho + \lambda + \sigma_i^2}$$

Maximizing the right hand side of this expression w.r.t. $\sigma_i \geq 0$ gives that

$$\sigma_i^{QL} \leq \frac{1}{2\sqrt{\rho + \lambda}} \quad \forall\ i$$

Finally similar calculations give

$$(\rho \cdot I + \boldsymbol{T}_u^T \boldsymbol{T}_u)^{-1} \boldsymbol{T}_u^T = \bar{\boldsymbol{V}} \boldsymbol{\Sigma}^\infty \bar{\boldsymbol{U}}^T$$

where the diagonal elements of $\boldsymbol{\Sigma}^\infty$ are given by

$$\sigma_i^\infty = \frac{\sigma_i}{\rho + \sigma_i^2}$$

Maximizing the right hand side gives that

$$\sigma_i^\infty \leq \frac{1}{2\sqrt{\rho}} \quad \forall\ i$$

■

The conclusions of the discussion above are the following. When dealing with computer control of real systems it is rather likely that the corresponding discrete time model will have some zero outside the stability region. It can easily be seen using simple examples that this corresponds to very small singular values of the corresponding matrix $\boldsymbol{T}_u$. The use of the input weighting in the criterion $\boldsymbol{W}_u > 0$, which implies $\boldsymbol{Q} \neq I$ then offers a method to control the "gain" from the error (or reference) signal to the applied input signal.

Frequency Domain Interpretation

The derivation and analysis of the optimization based ILC algorithm have so far been carried out in the time domain. In many cases however useful insights and interpretations of an ILC algorithm are achieved by considering the problem in the frequency domain. The aim in this section is to present a frequency domain interpretation of the ILC algorithm derived using optimization and illustrate some of its properties.

The starting point when deriving a frequency domain interpretation of the updating equation (8.14) is to use equation (8.3) in its simplest version, i.e.,

$$\boldsymbol{y} = \boldsymbol{T}_u \boldsymbol{u}$$

Consider now a discrete time system with impulse response $g_{T_u}(l), \quad l = 0, \ldots, \infty$ and transfer operator $T_u(q)$ defined by

$$T_u(q) = \sum_{l=0}^{\infty} g_{T_u}(l) q^{-l}$$

where q denotes the shift operator. Applying an input signal $u(t)$ which is zero for $t < 0$ the output can be written

$$y(t) = \sum_{l=0}^{t} g_{T_u}(l) u(t-l)$$

i.e.,

$$y(t) = T_u(q) u(t)$$

Provided that the inverse exists the relationship

$$\boldsymbol{y} = \boldsymbol{T}_u^{-1} \boldsymbol{u}$$

corresponds to the filtering

$$y(t) = \frac{1}{T_u(q)} u(t)$$

Furthermore the operation

$$\boldsymbol{y} = \boldsymbol{T}_u^{T} \boldsymbol{u}$$

corresponds to a non-causal filtering

$$\begin{aligned} y(t) &= g_{T_u}(0)u(t) + g_{T_u}(1)u(t+1) + \ldots + g_{T_u}(n-t)u(n) = \\ &= \sum_{l=0}^{n-t} g_{T_u}(l)u(t+l) = \sum_{l=0}^{n-t} g_{T_u}(l)(\frac{1}{q})^{-l}u(t) = T_u(\frac{1}{q})u(t) \end{aligned}$$

Hence the operation

$$\boldsymbol{y} = \boldsymbol{T}_u^T \boldsymbol{T}_u \boldsymbol{u}$$

corresponds to

$$y(t) = T_u(q)T_u(\frac{1}{q})u(t)$$

or, in the frequency domain,

$$Y(\omega) =\mid T_u(e^{i\omega}) \mid^2 U(\omega)$$

i.e., a filtering operation that gives zero phase shift. One implementation of this operation is provided by the MATLAB™ command `filtfilt`.

Now consider again (8.14)

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}(\boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{e}_k) \tag{8.19}$$

where putting $\boldsymbol{W}_e = I$ and $\boldsymbol{W}_u = \rho \cdot I$ the matrices $\boldsymbol{Q}$ and $\boldsymbol{L}$ are given by

$$\boldsymbol{Q} = (\rho \cdot I + \lambda \cdot I + \boldsymbol{T}_u^T \boldsymbol{T}_u)^{-1}(\lambda \cdot I + \boldsymbol{T}_u^T \boldsymbol{T}_u)$$

and

$$\boldsymbol{L} = (\lambda \cdot I + \boldsymbol{T}_u^T \boldsymbol{T}_u)^{-1} \boldsymbol{T}_u^T$$

It is now possible to express (8.19) as filter operators,

$$Q(q) = \frac{\lambda + T_u(q)T_u(q^{-1})}{\rho + \lambda + T_u(q)T_u(q^{-1})} \tag{8.20}$$

and

$$L(q) = \frac{T_u(q^{-1})}{\lambda + T_u(q)T_u(q^{-1})} \tag{8.21}$$

Equations (8.20) and (8.21) hence represent particular choices of the filters in equation (3.13). It is now possible to formulate a second, simplified, version of Algorithm 8.4.

Algorithm 8.5 (Optimization based ILC design, filter implementation)

1. Build a model of the relation between the ILC input and the resulting correction on the output, i.e., find a model $\widehat{T}_u(q)$ of $T_u(q)$.
2. Choose the weight $W_u = \rho$ (W_e is chosen as 1) and the Lagrange multiplier λ in the criterion.

3. Calculate $Q(q)$ and $L(q)$ according to (8.20) and (8.21).
4. Use the ILC updating equation,

$$u_{k+1}(t) = Q(q)(u_k(t) + L(q)e_k(t)) \tag{8.22}$$

with $\boldsymbol{u}_0$ for example chosen as $\boldsymbol{u}_0 = 0$.

It should be observed that all boundary effects in the beginning and the end of the data sequences are neglected in the derivation of the transfer function formulation. Hence the updating equations (8.22) and (8.14) do not give exactly the same results. If the length of the iterations, n, is large the frequency domain expressions give a useful interpretation of the optimization based ILC algorithm.

Assuming that the nominal model equals the true system the stability criterion

$$\mid Q(e^{i\omega}) \mid\mid 1 - L(e^{i\omega})T_u(e^{i\omega}) \mid < 1$$

from Theorem 4.8 becomes

$$\frac{\lambda}{\lambda + \rho + \mid T_u(e^{i\omega}) \mid^2} < 1 \tag{8.23}$$

This condition is satisfied for all ω provided that $\rho > 0$, cf. Theorem 8.1. Since the gain of the system tends to zero for high frequencies the left hand side of the inequality tends to $\lambda/(\lambda + \rho)$ for high frequencies. It is also of interest to compute the asymptotic error that is obtained provided that the ILC algorithm converges and this is given by

$$e_\infty(t) = \frac{\rho}{\rho + T_u(q)T_u(q^{-1})}e_0(t) \tag{8.24}$$

The value of ρ hence influences the magnitude of the error after it has converged. Choosing $\rho > 0$ guarantees that the condition in (8.23) holds, but the price is that there will still be a remaining error also asymptotically. This is a well known fact in ILC but here this trade off is concentrated in one scalar parameter.

Numerical Example

In this section an example will be given where the properties of the filters $L(q)$ and $Q(q)$ are investigated. The example will be based on the first order discrete-time system

$$T_u(q) = \frac{0.07q^{-1}}{1 - 0.93q^{-1}} \tag{8.25}$$

obtained by identification of an ABB IRB 1400, see Section 7.3.3. This model will also be used in the experiments in Section 8.3.1. The data used for the identification

were collected while the robot was controlled by the conventional control system. This means that $T_u(q)$ in (8.25) is a model of the closed loop system from the ILC input signal $u_k(t)$ to the output signal $y_k(t)$. See also Figure 7.5.

The filter $L(q)$ is determined by the nominal model $T_u(q)$ and the parameter λ. In (8.21) it can be seen that when λ decreases the filter tends towards the inverse of the nominal model. In Figure 8.3 the gain of $L(e^{i\omega})$ is shown for different values of λ. For small values of λ the gain of $L(e^{i\omega})$ is close to the gain of $T_u^{-1}(e^{i\omega})$ while for larger values it obtains more low-pass character. The phase of $L(e^{i\omega})$ is however not effected by λ and it is equal to the phase of $T_u^{-1}(e^{i\omega})$ for all values of ρ.

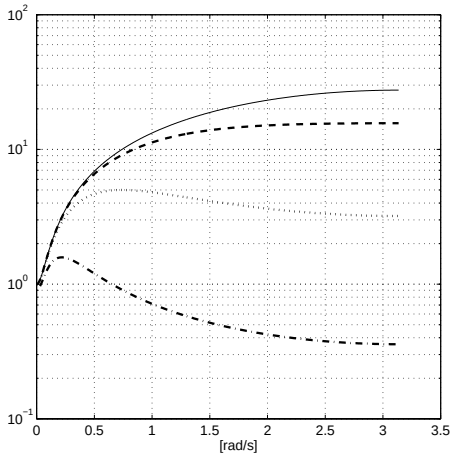


Figure 8.3 *Magnitude of $T_u^{-1}(z)$ (solid) and $L(z)$ for $\lambda = 10^{-3}$ (dashed), $\lambda = 10^{-2}$ (dotted), and $\lambda = 10^{-1}$ (dash-dotted).*

The properties of $Q(e^{i\omega})$ are determined by both λ and ρ and the shape of $Q(e^{i\omega})$ depends on both their values and their relative size. Provided that the gain of $T_u(e^{i\omega})$ decreases as ω increases, the gain of $Q(e^{i\omega})$ tends to $\lambda/(\lambda+\rho)$ as ω increases. This means that the value of ρ has to be of the same order of magnitude as λ in order to give a significant low pass behavior of $Q(e^{i\omega})$. When ρ is essentially smaller than λ the high frequency gain of $Q(e^{i\omega})$ is close to unity. This can be seen in Figure 8.5. The low frequency gain of $Q(e^{i\omega})$ also depends on the low frequency gain of the nominal model $T_u(e^{i\omega})$. Some examples are given in Figure 8.4 and Figure 8.5 respectively.

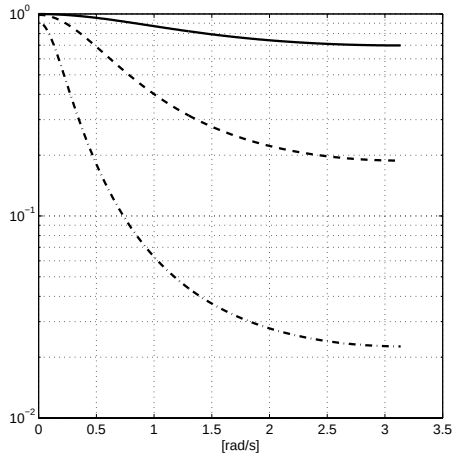


Figure 8.4 *Gain of $Q(z)$ for $\lambda = 10^{-3}$, and $\rho = 10^{-3}$ (solid), $\rho = 10^{-2}$ (dashed), $\rho = 10^{-1}$ (dash-dotted).*

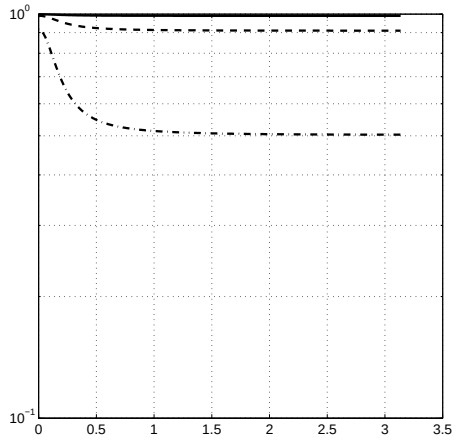


Figure 8.5 *Gain of $Q(z)$ for $\lambda = 10^{-1}$, and $\rho = 10^{-3}$ (solid), $\rho = 10^{-2}$ (dashed), $\rho = 10^{-1}$ (dash-dotted).*

8.3 Examples

Next, three of the presented algorithms are implemented and evaluated on the industrial robot described in Chapter 6. The algorithms that will be tested are, Algorithm 8.1, Algorithm 8.2, and Algorithm 8.4. The models that are used for the design are found according to the description in Section 7.3. Two different test cases have been used in the evaluation. The first is a 1 joint experiment, involving only axis one on the robot. The second experiment is an example of a multiple joint motion where ILC is applied to three of the joints of the robot.

8.3.1 One joint motion experiment

Description of the test case

The first experiment involves one of the joints of the robot, as shown in Figure 8.6 (cf. Figure 6.3(b)). The reference trajectory for the motor angle is depicted at the right of Figure 8.6. On the arm side the motion corresponds to a rotation, counter-clockwise, of 1 rad of axis 1. Since the motion on the arm side is 1 rad it follows that the gear ratio is about 120:1 because on the motor side the motion stops at, about, 120 rad. To give an idea of the distance that the tool has moved in this case, the length of the upper arm, link 3, is about 1 m. Moving 1 rad means therfore a trajectory length of 1 m. This motion is performed in about one second and this means a speed of about 1 m/s which also correspond to the programmed speed of the TCP.

The model of the system from the ILC input to the output is given by

$$\widehat{T}_u(q) = \frac{0.07q^{-1}}{1 - 0.93q^{-1}} \tag{8.26}$$

i.e., a system having low pass characteristics.

Design 1: Heuristic design, Algorithm 8.1

The design of the first ILC algorithm follows the design scheme in Algorithm 8.1.

1. The Q-filter is chosen as a zero-phase filter $Q(q) = \bar{Q}(q)\bar{Q}(\frac{1}{q})$ with $\bar{Q}(q)$ as a Butterworth filter of second order with cut-off frequency 0.2 of the Nyquist frequency. In MATLAB™ this filter can be calculated by using the following command,

   ```
   >> [Qb,Qa] = butter(2,0.2);
   ```

   and a zero-phase filter is achieved by using the function `filtfilt`. In Figure 8.7 the amplitude diagram for $\bar{Q}(e^{i\omega t_s})$ is depicted. The sampling time, t_s, is here 0.004 s.

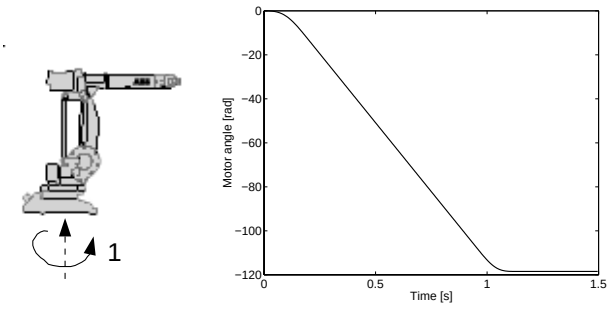


Figure 8.6 *Illustration of the motion in the one joint experiment. Axis 1 motion (left), reference for the motor angle (right).*

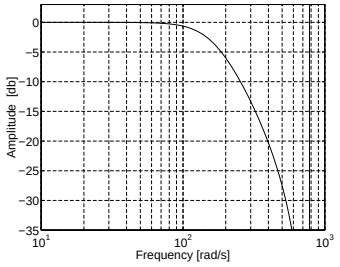


Figure 8.7 *The amplitude plot for the filter $\bar{Q}$.*

The bandwidth of the low-pass filter is chosen such that the bandwidth of the ILC algorithm is sufficiently high. A Butterworth filter is chosen, simply, because it is easy to find using MATLAB™.

2. The L-filter is chosen as

$$L(q) = 0.9q \tag{8.27}$$

i.e., $\kappa = 0.9$ and $\delta = 1$. This corresponds to what is referred to as "normal" choice in the algorithm since the static gain of the system equals 1 and the time delay also equals 1. The condition $|Q(e^{i\omega t_s})(1 - L(e^{i\omega t_s})T_u(e^{i\omega t_s}))| < 1$ is also fulfilled for the model, $\widehat{T}_u$ and this choice of L-filter.

Design 2: Model-based design, Algorithm 8.2

Following Algorithm 8.2 the result becomes:

1. The model $\widehat{T}_u$ is given by (8.26).
2. An aggressive approach is applied here where $H_B(q)$ is chosen as $H_B(q) = 0$. Robustness is achieved using the Q-filter.
3. The choice of H_B above gives $L(q) = \widehat{T}_u^{-1}(q)$, i.e., the inverse of the system model.
4. The filter Q is chosen as in the heuristic design above, i.e., a zero-phase filter $Q(q) = \bar{Q}(q)\bar{Q}(\frac{1}{q})$ with $\bar{Q}(q)$ as a Butterworth filter of second order with cut-off frequency 0.2 of the Nyquist frequency.

Design 3: Optimization based design, Algorithm 8.4

The optimization based design is implemented according to the description in Algorithm 8.4 using the matrix formulation. The choices of weight matrices are however inspired by the design in Algorithm 8.5.

1. The model $\widehat{T}_u$ is the same as above, i.e., given by (8.26). The matrix $\widehat{\boldsymbol{T}}_u$ is simply the lower triangular Toeplitz matrix created from the impulse response of (8.26).
2. The weight matrices are chosen as $\boldsymbol{W}_e = I$, and $\boldsymbol{W}_u = \rho \cdot I$, with $\rho = 0.01$. The Lagrange multiplier is chosen as $\lambda = 0.001$.
3. $\boldsymbol{Q}$ and $\boldsymbol{L}$ in the ILC algorithm are calculated according to

$$\boldsymbol{Q} = ((\rho + \lambda) \cdot I + \widehat{\boldsymbol{T}}_u^T \widehat{\boldsymbol{T}}_u)^{-1} (\lambda \cdot I + \widehat{\boldsymbol{T}}_u^T \widehat{\boldsymbol{T}}_u)$$

and

$$\boldsymbol{L} = (\lambda \cdot I + \widehat{\boldsymbol{T}}_u^T \widehat{\boldsymbol{T}}_u)^{-1} \widehat{\boldsymbol{T}}_u$$

The resulting ILC algorithms are next applied to the industrial robot in the test case described earlier.

Results from the one joint motion experiments

The three different ILC designs have run for a total of 11 iterations each, where in the first iteration the control signal $u_0 \equiv 0$ has been used. In Figure 8.8 the

resulting normalized maximum value,

$$V_{k,i}^{\infty} = \frac{\|e_{k,i}\|_{\infty}}{\max\limits_{j=1,2,3} \|e_{0,j}\|_{\infty}} \quad i = 1, 2, 3 \tag{8.28}$$

is shown together with the normalized 2-norm,

$$V_{k,i}^{2} = \frac{\|e_{k,i}\|_{2}}{\max\limits_{j=1,2,3} \|e_{0,j}\|_{2}} \quad i = 1, 2, 3 \tag{8.29}$$

which corresponds to a measure of the energy of the error. The normalization factor is chosen such that, after the normalization, $V_{0,i}^{\infty}$ and $V_{0,i}^{2}$ are less than or equal to 1 for all the algorithms ($i = 1, 2, 3$). From Figure 8.8 it is clear that the three different ILC algorithms give the same initial values of the two norms, since $V_{k,i}^{p} = 1$ for $p = 2, \infty$.

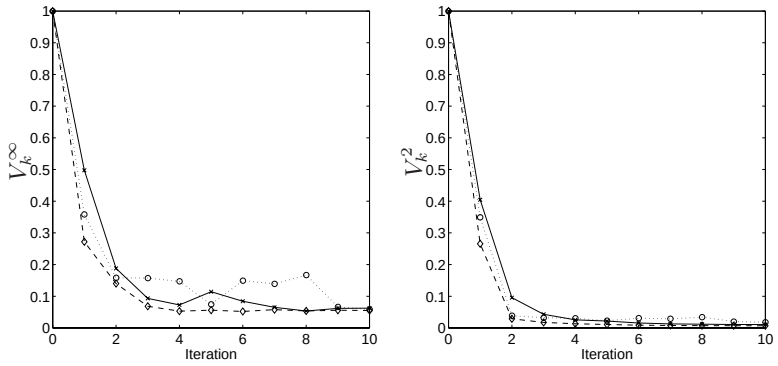


Figure 8.8 *Normalized maximum error (left) and normalized error energy (right) for design 1 (×), design 2 (◇), and design 3 (o).*

Figure 8.8 shows that the fastest convergence is achieved with the model based design, i.e., design 2. This approach gives the best behavior when considering either of the measures in Figure 8.8. The second best is the optimization based approach, design 3, and the difference is very small when evaluating the 2-norm while the difference in the maximum value is a bit greater. Common for the two best approaches is that they both use, explicitly, the system model in the ILC updating equation. The heuristic design can not compete with the two other ILC implementations in the rapid convergence but, clearly, after 4 iteration the level of the error is the same for all the different suggested algorithms and if it is acceptable to run a total of 5 iterations instead of 3 or 4 iterations, which is the case for the other ILC schemes, the heuristic design can be chosen.

In Figure 8.9 the resulting normalized errors for the different ILC schemes are shown in the time domain. It is clear that the low frequency components are

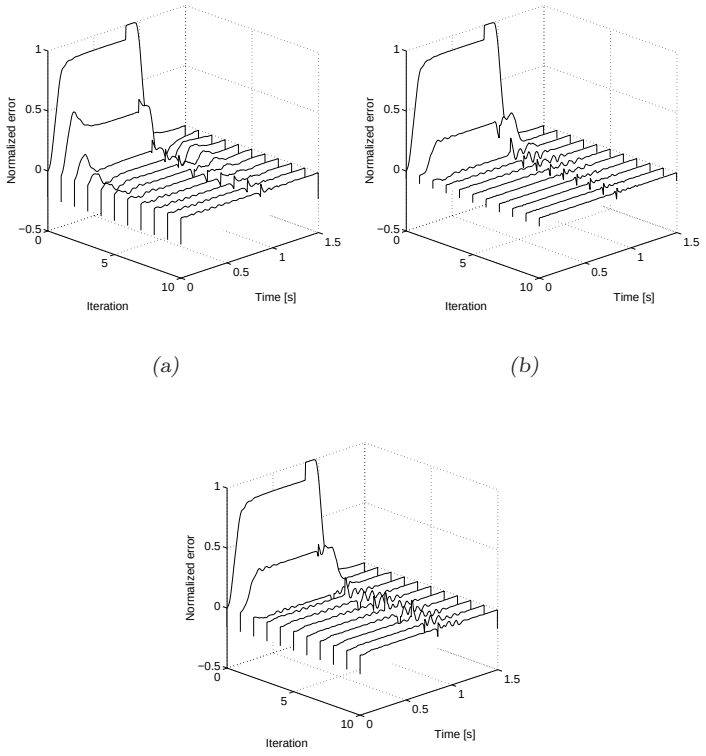


Figure 8.9 *The errors e_k in the time domain, from design 1 (a), design 2 (b) and design 3 (c).*

compensated for rapidly by all the three methods and that the damping rate for high frequencies is much lower. The small step in the error after about 1 second is not completely compensated for after 10 iterations using any of the three methods. It is also clear why only one of the two measures in Figure 8.8 can not, alone, give a good measure of the error. A large but very short peak in the error gives a high maximum absolute value of the error but it does not increase the 2-norm of the error very much. On the other hand, a small but constant error does not necessarily have to effect the maximum value of the error but it might give an increased energy of the error signal.

8.3.2 Multiple joint motion experiment

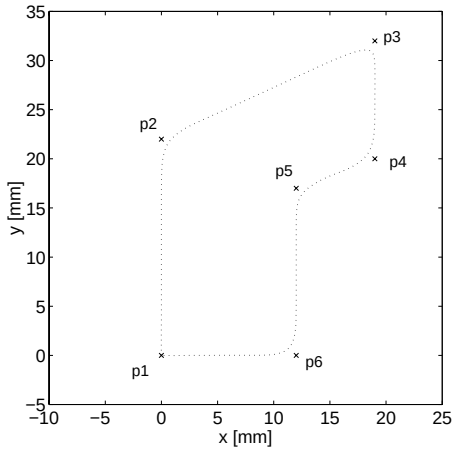
Description of the test case

In the second test case ILC is applied to 3 of the total 6 joints of the IRB 1400. Each of the 3 joints is modeled as a transfer function description from the ILC input to the measured motor position on the robot. The conventional feedback controller, implemented by ABB in the S4C control system, works in parallel with the ILC algorithms. Since the controller is working well, the closed loop from reference angular position to measured angular can be described using a low order linear discrete time model. The models are obtained by applying system identification to the closed loop system according to Section 7.3.

In Figure 8.11 the program used in the experiment is shown together with the desired trajectory on the arm-side of the robot. The program in Figure 8.11 is simplified, syntactically, compared to the true program. Semantically, however, the programs are the same. The instruction `moveL p2,v100,z1` refers to an instruction that produces a straight line on the arm-side of the robot. The line starts from the current position, not explicitly stated, and ends in `p2`. The speed along the path is in this case programmed to be 100 mm/s. The last parameter `z1` indicates that the point `p2` is a zone point. This means that the robot will pass in a neighborhood of the point with a distance not more than 1 mm. This can also be seen in Figure 8.11. The `moveL` instruction is a simplified version of the corresponding instruction `MoveL` in the RAPID programming language. The actual position of `p1` in the base coordinate system is x = 1300 mm, y = 100 mm, and z = 660 mm. The configuration of the robot is also shown in Figure 8.10.

As a first step in the design of the ILC schemes an identification experiment is

Figure 8.10 *The ABB IRB1400 manipulator.*




```
%% starting at p1
moveL p2,v100,z1;
moveL p3,v100,z1;
moveL p4,v100,z1;
moveL p5,v100,z1;
moveL p6,v100,z1;
moveL p1,v100,fine;
```

Figure 8.11 *The program used to produce the trajectory used in the example (left) and the resulting trajectory on the arm-side translated such that p1 is in the origin (right).*

performed. This step is simply done by using the ILC input u according to the discussion in Section 7.3. The result from this step is three models, one for each joint. The models are calculated using *System Identification Toolbox* (Ljung, 1995) and the models are of ARX type with $na = 1$, $nb = 1$, and $nk = 1$, The models used in the multiple joint motion experiment are

$$\widehat{T}_{u,1}(q) = \widehat{T}_{u,2}(q) = \frac{0.1q^{-1}}{1 - 0.9q^{-1}} \tag{8.30}$$

$$\widehat{T}_{u,3}(q) = \frac{0.13q^{-1}}{1 - 0.87q^{-1}} \tag{8.31}$$

The models are now utilized in order to design the different ILC algorithms.

Design 1: Heuristic design, Algorithm 8.1

The design follows the steps in the heuristic design, Algorithm 8.1.

1. The Q-filter is chosen as a zero-phase low-pass filter, $Q(q) = \bar{Q}(q)\bar{Q}(\frac{1}{q})$. $\bar{Q}(q)$ is a second order Butterworth filter with cut-off frequency 0.2 of the Nyquist frequency.

2. The L-filter has been chosen the same for the three joints,

$$L(q) = 0.9q^4$$

i.e., $\kappa = 0.9$ and $\delta = 4$. This choice can be explained by calculating

$$\sup_{\omega \in [0, \pi/t_s]} |Q(e^{i\omega t_s})(1 - L(e^{i\omega t_s})\widehat{T}_u(e^{i\omega t_s}))|$$

for different choices of δ. For robustness reasons this should be as low as possible and, as was also pointed out in Norrlöf (1998), it has a minimum for $\delta = 5$. The choice here is, however, $\delta = 4$ which gives nearly the same value of the expression above.

Design 2: Model-based design, Algorithm 8.2

1. The models are given by (8.30) and (8.31).
2. The same aggressive approach is employed as in the one joint motion experiment and H_B is chosen as $H_B(q) = 0$.
3. With the choice of H_B the L-filters become

$$L_i(q) = T_{u,i}^{-1}(q), \quad i = 1, 2, 3$$

4. The Q-filter is chosen as in design 1 above, i.e., as a zero-phase low-pass filter.

Design 3: Optimization based design, Algorithm 8.4

Here two different values of the parameters in the design of the optimization based ILC algorithm have been chosen. The resulting algorithms will be referred to as, design 3a, and design 3b, respectively.

1. The models of the three joints are given in (8.30) and (8.31) and the matrices $\widehat{\boldsymbol{T}}_{u,1}$, $\widehat{\boldsymbol{T}}_{u,2}$, and $\widehat{\boldsymbol{T}}_{u,3}$ are found in the same way as in design 3, Section 8.3.1.
2. The weight matrices are chosen as $\boldsymbol{W}_e = I$, and $\boldsymbol{W}_u = \rho \cdot I$ with $\rho_a = 0.01$ and $\rho_b = 0.1$ in design 3a and design 3b, respectively. The Lagrange multiplier is chosen as $\lambda = 0.1$.
3. The matrices in the ILC updating equations $\boldsymbol{Q}_i$ and $\boldsymbol{L}_i$, for $i = 1, 2, 3$, are calculated according to

$$\boldsymbol{Q}_i = ((\rho + \lambda) \cdot I + \widehat{\boldsymbol{T}}_{u,i}^T \widehat{\boldsymbol{T}}_{u,i})^{-1} (\lambda \cdot I + \widehat{\boldsymbol{T}}_{u,i}^T \widehat{\boldsymbol{T}}_{u,i})$$

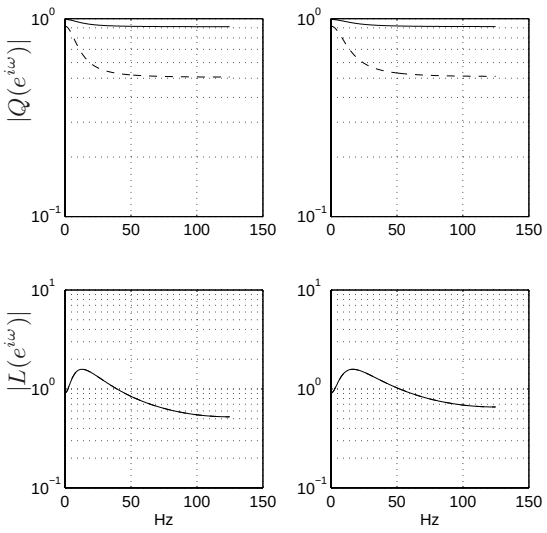


Figure 8.12 *The frequency interpretation of the Q-filter (upper) and the L-filter (lower) for joints 1 and 2 (left) and joint 3 (right), design 3a (solid) and design 3b (dashed).*

and

$$\boldsymbol{L}_i = (\lambda \cdot I + \widehat{\boldsymbol{T}}_{u,i}^T \widehat{\boldsymbol{T}}_{u,i})^{-1} \widehat{\boldsymbol{T}}_{u,i}$$

where ρ_a and ρ_b are used. In Figure 8.12 the frequency domain interpretation of the resulting Q and L filters are shown.

Note that the choice of ρ only has an effect on the Q-filter, i.e., the robustness of the ILC system. Clearly an increased ρ gives a more robust algorithm but also a lower bandwidth as shown in Figure 8.12.

Results from the multiple joint motion experiments

The four different ILC algorithms resulting from the three design algorithms have been running for a total of 11 iterations, as in the one joint motion experiments. The resulting normalized maximum errors for design 1 and 2 are shown in Figure 8.13 and the corresponding results for the designs 3a and 3b are shown in Figure 8.14. In Figure 8.13 and Figure 8.14 the normalized 2-norm of the error is also presented. The two measures are calculated similar to (8.28) and (8.29) but with the factor in the denominator being maximum also of the 3 joints.

From the upper row of diagrams in Figure 8.13 it can, for example, be seen that the maximum value of the error of joint 1 is about 50 % of what is achieved by joint 2. The initial value of the maximum error as well as the energy in the different experiments is not exactly the same but the behavior of the different algorithms can still be evaluated.

From Figure 8.13 and Figure 8.14 it is clear that the best result is achieved with the two (explicitly) model based ILC algorithms, i.e., design 2 and design 3. Clearly, by adjusting the design parameters in design 3 it is possible to get a slower and more robust scheme, as in design 3b, or a faster and less robust one, as in design 3a. Also in this more complicated motion the resulting behavior after 5-6 iterations is very similar for the different ILC algorithms. If it is acceptable to run 5-6 iterations then any of the ILC algorithms can be chosen. The algorithm in design 3b has, however, the disadvantage that there is a large steady state error. This is caused by the fact that the gain of the Q-filter is less than 1 for all frequencies, as can be seen in Figure 8.12.

Another way of evaluating the result of applying ILC to the robot is to transform the measured motor angles to the arm-side using the kinematic model of the robot, as was discussed in Chapter 6. In Figure 8.15 and Figure 8.16 the result from this operation is depicted for design 1 and 2, and design 3a and 3b, respectively. It is clear that the error when doing a transformation to the arm-side is not very big but it is important to stress that the transformation has been carried out under the assumption that the robot is stiff which is not true. In the next chapter the true path done by the robot will be evaluated and a possible solution including an additional sensor for measuring this path is also discussed.

8.4 Summary and Conclusions

Four different design strategies have been presented in detail and three of them have also been implemented on the industrial robot in different experiments. Three of the four design methods use explicitly a model of the system which can be considered a bit in contradiction with the original ILC idea. Often ILC is presented as a non model based approach which, as has been shown here, is not the case. Also the first approach, the *heuristic approach*, uses a model to secure stability of the ILC system.

To decide the best choice of design strategy is not so easy, but there are two general comments that can be made. First, *"try simple things first"*, which means that if Algorithm 8.1 gives sufficient performance this is the algorithm that should be used. If the performance is not enough a model based approach has to be chosen. The optimization based approach, as it is presented in Algorithm 8.5, has only two scalar design parameters to tune the performance/robustness of the algorithm. The deign based on Algorithm 8.2 is also straightforward to apply, at least as it has been done in the two experiments described in this chapter. What can be a

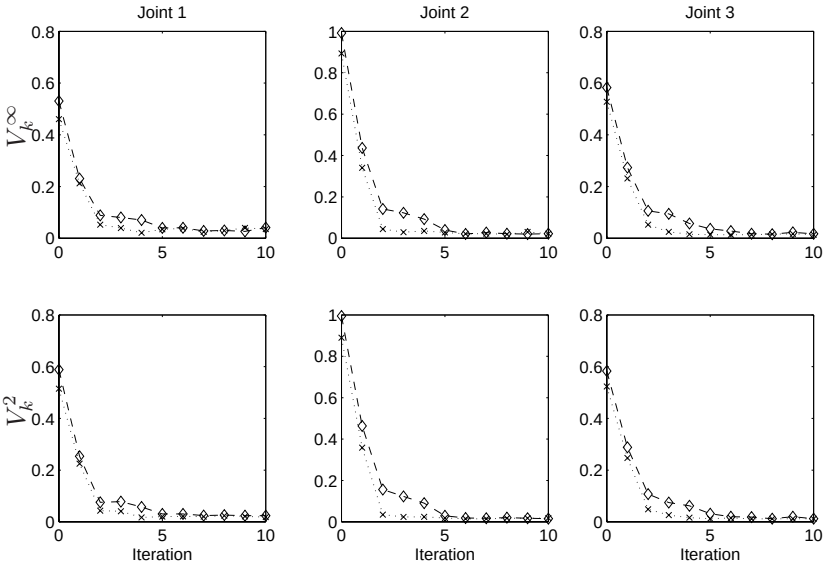


Figure 8.13 *Normalized maximum error (upper) and normalized energy of the error (lower) for joint 1 to joint 3 from left to right, design 1 ($\diamond$) and design 2 ($\times$).*

risk is, however, that the inverse system model solution is too aggressive and might lead to instability. The Q-filter makes the algorithm more robust but it also limits the bandwidth of the ILC system. The second, and final, comment is, *"use all information available"*. This means that if a good model is available this model should be used for the analysis and also for simulations to evaluate the resulting design.

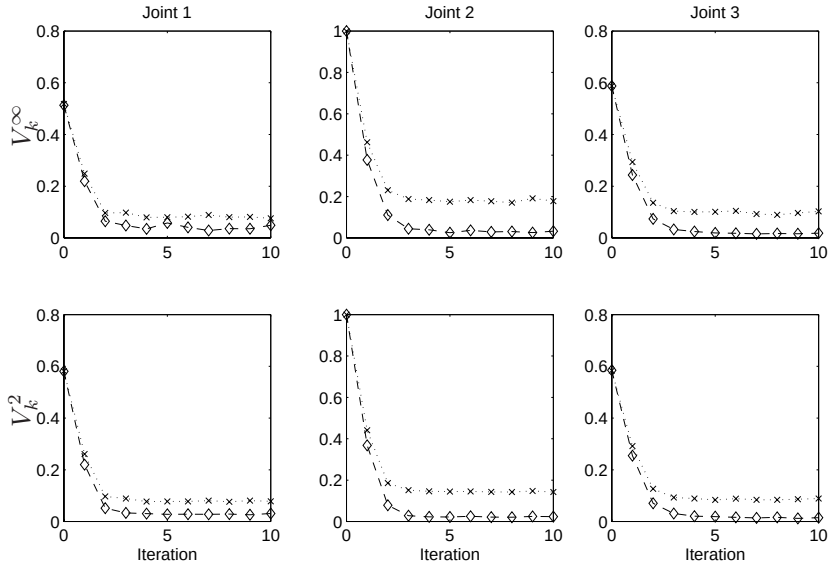


Figure 8.14 *Normalized maximum error (upper) and normalized energy of the error (lower) for joint 1 to joint 3 from left to right, design 3a ($\diamond$) and design 3b ($\times$).*

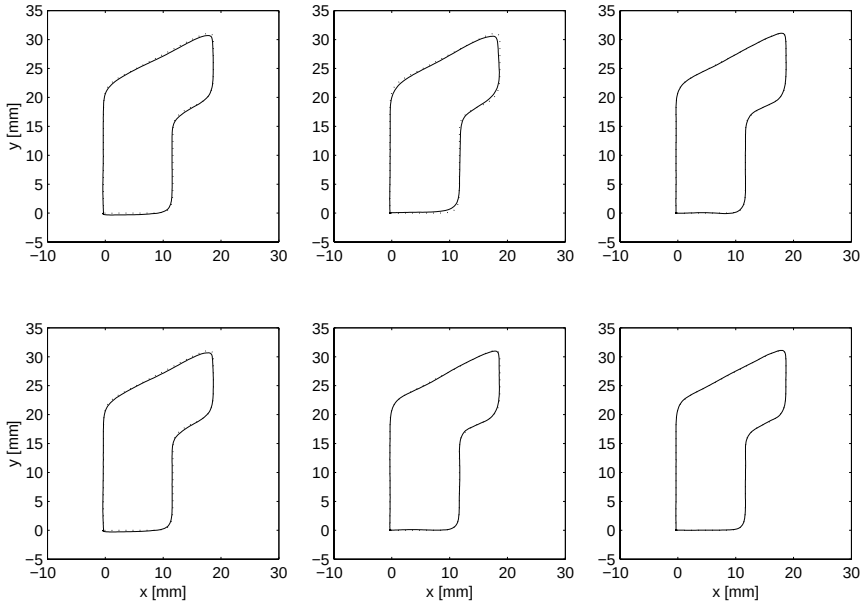


Figure 8.15 *Resulting trajectory tranformed to the arm-side using the forward kinematics (solid) and the reference trajectory (dotted). Upper row, design 1, and lower row, design 2, iteration 0, 1, and 5, from left to right.*

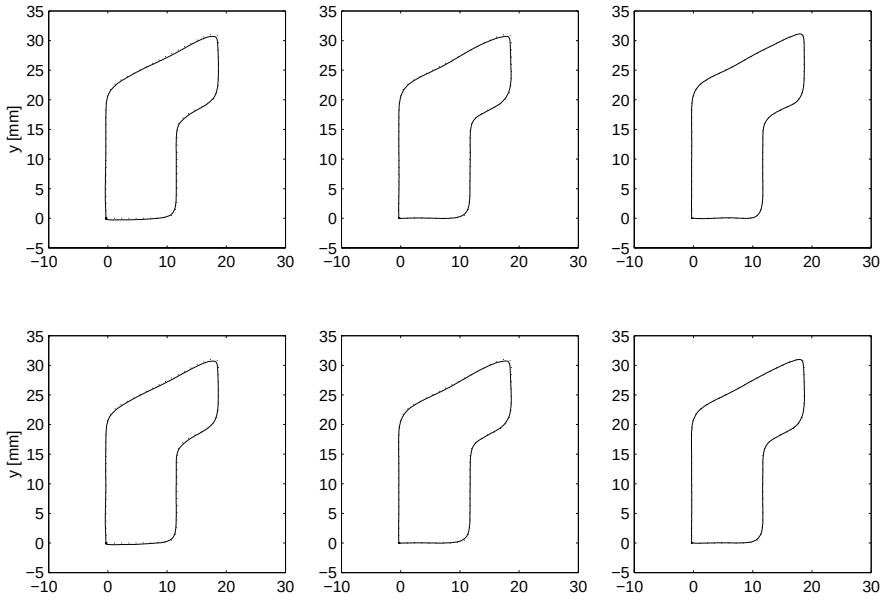


Figure 8.16 *Resulting trajectory tranformed to the arm-side using the forward kinematics (solid) and the reference trajectory (dotted). Upper row, design 3a, and lower row, design 3b, iteration 0, 1, and 5, from left to right.*

9

Limitations and Possibilities

In this chapter some basic limitations of ILC will be covered. The discussion will be done from the industrial robot application perspective and this will also be the example used throughout the chapter.

9.1 Trajectory Tracking

In the trajectory tracking formulation of ILC, presented in Chapter 3, there is an important implicit assumption that the error that should be minimized is also available as a measurement. In the presented ILC updating equations, both in the thesis as well as in the literature, this assumption shows up in practice in the fact that the error is constructed according to

$$e_k(t) = r(t) - y_k(t)$$

In general this measure is also what the ILC method tries to minimize.

Throughout the thesis it is assumed that the error $e_k(t)$ is also a measure of the true error $\epsilon_k(t)$, that the control algorithm actually *should* minimize. In the tracking formulation in Section 3.1, it is assumed that $y_k(t)$ is a direct measure of $z_k(t)$, except the measurement noise. In most cases this is also true in applications of

ILC but sometimes it is not possible to measure directly the controlled variable. It might, of course, be that another signal that depends indirectly on the actual error can be measured and in this case it is possible to calculate the controlled variable, $z_k(t)$, from the measured variable, $y_k(t)$.

For the application studied in the thesis, path following for industrial robots, it is actually true that the actual error is not measurable directly. In the examples presented in the thesis, the error on the motor side of the robot is studied and used in the ILC algorithm. Of course, the goal is to minimize the error on the arm side and it is possible to find cases where this is not so successful. Consider the program and the path shown in Figure 8.11. If the result on the arm-side is considered instead of the motor-side as done in the example in Section 8.3.2, then a result according to Figure 9.1 is achieved.

(a) Trajectory without ILC.

(b) Design 1, Section 8.3.2, iteration 5 (left) and iteration 10 (right).

Figure 9.1 *Results on the arm-side of the robot.*

Although the first path in Figure 9.1(a) does not exactly correspond to the programmed path, it does look much more like the reference path than what is achieved after 5 and 10 iterations using ILC, as shown in Figure 9.1(b).

This is because the ILC algorithm, as implemented here, assumes that if it is possible to make the measured output (the motor angle) follow the reference as well as possible, then this also implies that the correct path is followed on the arm-side. If the gearbox and the arm were completely rigid, without friction and backlash, etc., this is actually true. Moreover, if it were possible to model the dynamics and compensate by filtering the motor reference position through the inverse dynamics, it would be possible to make the robot perfectly track the reference path also on the arm-side. Of course, modeling all the involved dynamics is difficult and the final result depends very much on the accuracy of the model. Additional measurements from the tool position should make the problem much easier and a method using ILC together with an additional measurement system has been implemented by

ABB Robotics with very successful results in industrial applications. This work has also resulted in an application for a patent (Gunnarsson et al., 2000). Next a possible extension for ILC on the arm-side of the robot is presented.

9.2 ILC Using Additional Measurements

The main contribution in this section is to present some initial experiences from the use of accelerometers when ILC is applied to flexible mechanical systems. This will be done by presenting a simulation study where ILC is used to control the motion of the second mass in a system consisting of two masses connected via a spring and a damper. Different learning strategies applied to flexible mechanical systems have been studied previously in e.g., Panzieri and Ulivi (1995), Velthuis et al. (1996) and Lange and Hirzinger (1999). In these papers it is assumed that the position (angle) of the mass to be controlled can be measured. Here it will instead be assumed that only the acceleration of the mass to be controlled can be measured. In Miyazaki et al. (1986), which also deals with a flexible system, an acceleration signal is used in an ILC algorithm. The way it is used and the design methodology for the ILC algorithm differ however completely from the approach presented here.

The presentation will be restricted to discrete-time, linear SISO systems.

9.2.1 The ILC algorithm

The ILC design method used here is the optimization based approach presented in the previous chapter. This approach has previously been considered in e.g., Gorinevsky et al. (1995), Lee and Lee (1998) and Phan (1998). This algorithm can however be interpreted as a more conventional filter based algorithm, as described in Gunnarsson and Norrlöf (1999a,b) as well as the previous chapter.

9.2.2 A flexible system

The main interest here is the application of ILC to systems containing flexibilities. As an example of this situation a linear model of a flexible servo will be studied, as shown in Figure 9.2. Here θ_m and θ_a denote motor angle and load angle respectively and u is the applied torque, which is the input signal. This can be viewed as a simplified description of a one link robot arm with the flexibility concentrated in the joint. It is assumed that the system is working in a position where the gravitational effects can be neglected. Furthermore the gear ratio is assumed to be equal to 1.

Torque balances yield the following equations

$$J_m\ddot{\theta}_m + f_m\dot{\theta}_m + k(\theta_m - \theta_a) + d(\dot{\theta}_m - \dot{\theta}_a) = u \tag{9.1}$$

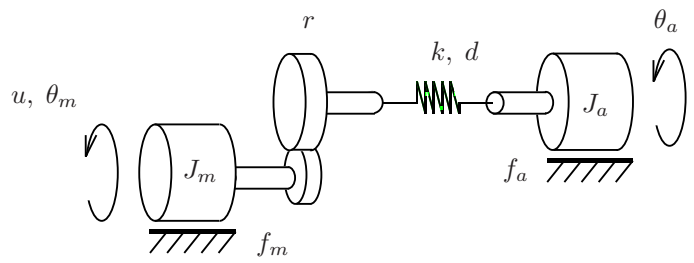


Figure 9.2 *Two-mass model.*

and

$$J_a\ddot{\theta}_a + f_a\dot{\theta}_a - k(\theta_m - \theta_a) - d(\dot{\theta}_m - \dot{\theta}_a) = 0 \tag{9.2}$$

where J_m and J_a denote the moment of inertia of each mass while f_m and f_a denote the viscous friction coefficient of each mass. Finally k and d denote the stiffness and damping respectively of the flexibility between the two masses. Using the state variables

$$x_1 = \theta_m \quad x_2 = \dot{\theta}_m \quad x_3 = \theta_a \quad x_4 = \dot{\theta}_a \tag{9.3}$$

the system can be described in state space form as

$$\dot{x} = Ax + Bu \qquad y = Cx = \begin{pmatrix} \theta_m \\ \ddot{\theta}_a \end{pmatrix} \tag{9.4}$$

since the available measurements are the position of the first mass and the acceleration of the second mass.

It is important to note that the main control goal is that the load angle θ_a follows a desired trajectory, like the one given by Figure 9.3. Note that this differs from the formulation used so far in the thesis. Before, it has always been assumed that the reference trajectory is given on the motor side. This is because the common case in many robot systems is that only the motor angle θ_m can be measured. Measurements of the load angle (position) are normally not available. The aim here is to investigate how the addition of an accelerometer measuring the load acceleration can be used in an ILC algorithm in order to improve the servo performance. In practice the accelerometer is mounted at the tool of the robot, measuring the acceleration of the tool in a Cartesian coordinate system. The problem of performing sensor-fusion with the measurements of the motor angle position and the Cartesian acceleration measurements will not be covered here but is left to future research. In Figure 9.4 a possible configuration of the accelerometer on the IRB1400 is shown.

The system is controlled using a conventional two-degrees of freedom controller structure giving the servo performance shown in Figure 9.3. The aim in the remaining part of this section is to investigate how ILC can be used to improve the

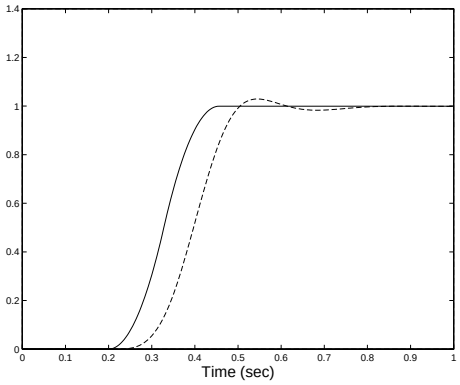


Figure 9.3 *Load angle reference (solid) and load angle (dashed) without ILC.*

Figure 9.4 *A possible configuration of the accelerometer on the IRB1400.*

servo properties both with respect to speed as well as to damping. In the simulations presented below the ILC signal is added to the original reference signal, which means that the ILC iterations generate a stepwise reshaping of the reference signal. It is also possible to add the ILC input to the control signal generated by the existing control system, which means that the ILC input will act as a torque feed-forward signal but this will not be covered here.

9.2.3 Estimating the load angle

The main idea here is to use an estimate of the load angle in the ILC algorithm. This estimate is formed as a combination of load acceleration and motor angle by using a state estimator based on the nominal relationship between the motor and

load angles. Recall equation (9.2)

$$J_a\ddot{\theta}_a + f_a\dot{\theta}_a - k(\theta_m - \theta_a) = 0 \tag{9.5}$$

where for simplicity the damping coefficient d is set to zero. Introducing the state variables $\bar{x}_1 = \theta_a$ and $\bar{x}_2 = \dot{\theta}_a$ and the measured signal $\bar{y}(t) = \ddot{\theta}_a$ equation (9.5) can be expressed in state space form as

$$\dot{\bar{x}} = \bar{A}\bar{x} + \bar{B}\theta_m \qquad \bar{y} = \bar{C}\bar{x} + \bar{D}\theta_m \tag{9.6}$$

A Kalman filter for this system is then given by

$$\dot{\hat{\bar{x}}} = \bar{A}\hat{\bar{x}} + \bar{B}\theta_m + K(\bar{y} - C\hat{\bar{x}} - \bar{D}\theta_m) \tag{9.7}$$

The estimated load angle can then be expressed using transfer functions as

$$\hat{\Theta}_a(s) = \hat{\bar{X}}_1(s) = F_{\bar{y}}(s)\bar{Y}(s) + F_{\theta_m}(s)\Theta_m(s) \tag{9.8}$$

where

$$F_{\bar{y}}(s) = \begin{pmatrix} 1 & 0 \end{pmatrix}(sI - \bar{A} + K\bar{C})^{-1}K \tag{9.9}$$

and

$$F_{\theta_m}(s) = \begin{pmatrix} 1 & 0 \end{pmatrix}(sI - \bar{A} + K\bar{C})^{-1}(\bar{B} - K\bar{D}) \tag{9.10}$$

When designing the Kalman filter the variance R_2 of the load acceleration signal can be used as a design variable that effects the balance between the measured load acceleration and the measured motor angle. Choosing a high value of R_2 implies that the load acceleration is almost neglected. The resulting filter from θ_m to $\hat{\theta}_a$ is then almost exactly the transfer function obtained from equation (9.5). The load angle estimate is then obtained by feeding the motor angle through the nominal transfer function from motor angle to load angle. In the nominal case this of course gives a good estimate of the load angle but the estimate is very sensitive to model errors. For example, a 20% error in J_a gives approximately 10% change in $\omega_a = \sqrt{k/J_a}$, determining the location of the peak in the transfer function. On the other hand, choosing a low value of R_2 implies that the load acceleration plays a large role in the estimation of load angle. The filters $F_{\bar{y}}$ and F_{θ_m} for this case are shown in Figure 9.5. In the considered frequency range the gain of the filters is almost constant. This is justified by (9.5) since a slight reformulation of gives

$$\theta_a = \theta_m - \frac{J_a}{k}\ddot{\theta}_a - \frac{f_a}{k}\dot{\theta}_a \tag{9.11}$$

If good measurements of $\ddot{\theta}_a$ are available an approximation of the load angle is given by

$$\hat{\theta}_a = \theta_m - \frac{\hat{J}_a}{\hat{k}}\ddot{\theta}_a \tag{9.12}$$

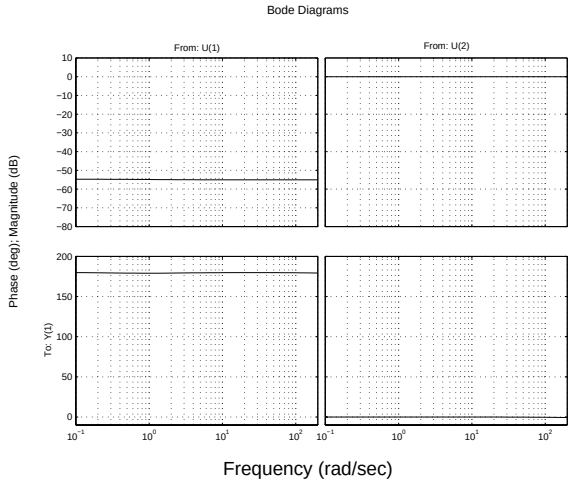


Figure 9.5 *Estimator filter for low values of* R_2. $\mid F_{\bar{y}}(i\omega) \mid$ *and* $\arg F_{\bar{y}}(i\omega)$ *(left column) and* $\mid F_{\theta_m}(i\omega) \mid$ *and* $\arg F_{\theta_m}(i\omega)$ *(right column).*

where the effect of the viscous friction has been neglected. Furthermore $\hat{k}$ and $\hat{J}_a$ denote estimates of the spring constant and the moment of inertia. The left part of Figure 9.5 hence corresponds to the constant $-\frac{\hat{J}_a}{\hat{k}}$ while the right part corresponds to the unit factor in front of θ_m. In order to analyze how sensitive the estimation procedure is the following calculations can be made. The relationship between the motor angle and the load angle is, using (9.2), given by

$$\Theta_a(s) = G_a(s)\Theta_m(s) \tag{9.13}$$

The measured load acceleration can therefore be expressed as

$$\bar{Y}(s) = s^2 G_a(s)\Theta_m(s) \tag{9.14}$$

Using (9.12) this gives

$$\hat{\Theta}_a(s) = (1 - \frac{\hat{J}_a}{\hat{k}} s^2 G_a(s))\Theta_m(s) \tag{9.15}$$

The transfer function

$$\hat{G}_a(s) = (1 - \frac{\hat{J}_a}{\hat{k}} s^2 G_a(s)) \tag{9.16}$$

giving the estimated load angle can now be compared with the transfer function $G_a(s)$ giving the true load angle. Figure 9.6 shows the Bode diagram of $G_a(s)$

and $\hat{G}_a(s)$ for some different values of $\hat{J}_a$. Around the resonance frequency the true $G_a(s)$ and the estimate $\hat{G}_a(s)$ are very close while they differ more for higher frequencies. It can also be seen that the gain of $\hat{G}_a(s)$ is systematically higher in the high frequency range than for $G_a(s)$. It therefore seems reasonable to low-pass filter the load angle estimate before using it in the ILC algorithm.

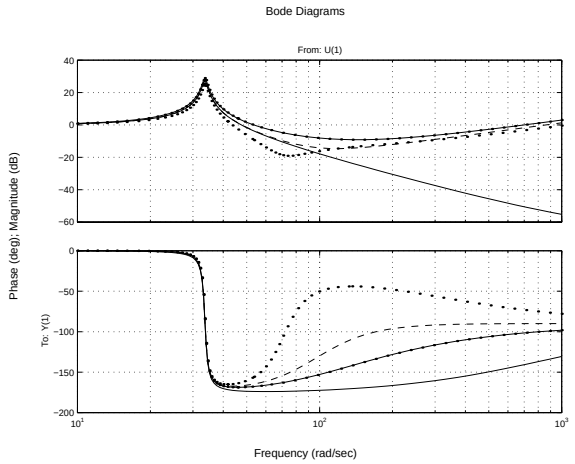


Figure 9.6 *$G_a(s)$ (solid) and $\hat{G}_a(s)$, $\hat{J}_a = J_a$ (dashed), $\hat{J}_a = 1.2J_a$ (dotted), and $\hat{J}_a = 0.8J_a$ (dash-dotted).*

9.2.4 ILC using estimated load angle error

The idea presented above is now evaluated in some simulations. The ILC input is computed as

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}(\boldsymbol{u}_k + \boldsymbol{L}\hat{\boldsymbol{e}}_k) \tag{9.17}$$

where $\hat{\boldsymbol{e}}_k$ is the error between load angle reference and estimated load angle.

In the first simulation example (the nominal case) it is assumed that the correct value of the ratio J_a/k is available and can be used in (9.12). The design variables are chosen as $\boldsymbol{W}_e = I, \boldsymbol{W}_u = 10^{-2} \cdot I$ and $\lambda = 10^{-2}$. Figure 9.7 shows the maximum absolute value of the error during 20 iterations. Figure 9.8 shows the reference load angle and achieved load angle after the last iteration.

In the next simulations the value of J_a differs from the nominal value used when designing the ILC algorithm. In the simulations the value of $\hat{J}_a$ is 20% larger/smaller than the true one. Figure 9.9 shows the evolution of the max error during the iterations in these two cases. The convergence is somewhat affected and the error settles at a slightly higher value.

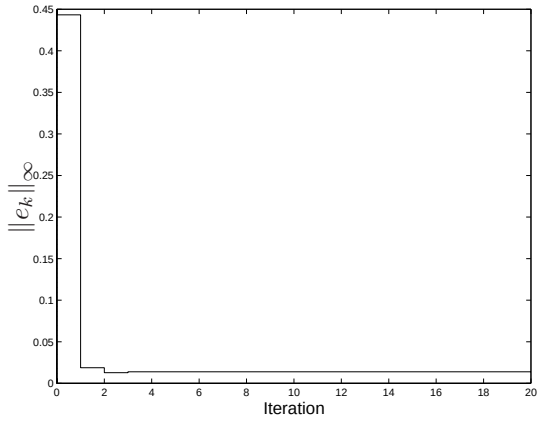


Figure 9.7 *Max error when* $\hat{J}_a = J_a$.

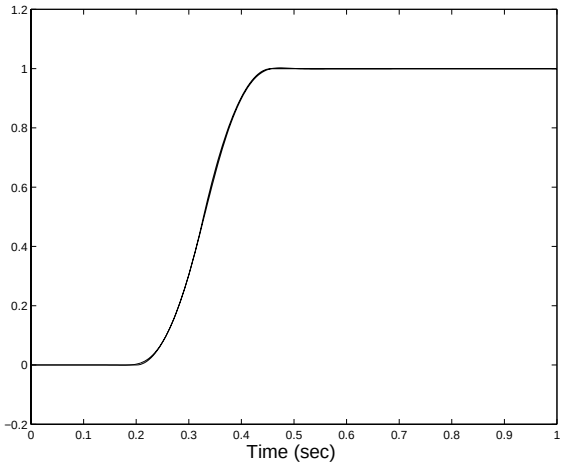


Figure 9.8 *Load reference angle and load angle after* 20 *iterations.*

Finally, the robustness against unmodeled dynamics is evaluated by applying the ILC algorithm to a system of higher order. The higher order system is obtained by splitting the second mass into two parts, each having moment of inertia $J_a/2$. The two masses are connected via a spring with spring constant $4.5 \cdot k$. Both masses are affected by viscous friction with friction constant f_a. It is assumed that the angle of interest is the angle of the third mass and that the corresponding acceleration is measured. Straightforward application of the ILC algorithm used above immediately leads to the well known behavior of ILC algorithms that the size of the error initially decreases when the low frequency components of the error

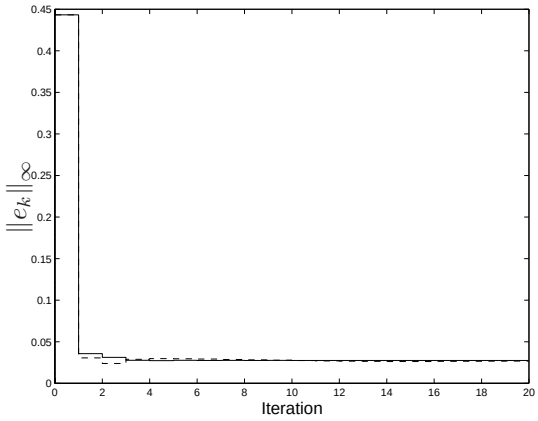


Figure 9.9 *Max error when $\hat{J}_a = 1.2J_a$ (solid) and $\hat{J}_a = 0.8J_a$ (dashed).*

dominate. while, after some iterations, it starts to grow when the high frequency components begin dominating the error (cf. Example 2.1). Since the performance requirements in the nominal case were high the instability was not so surprising. In order to improve the robustness of the ILC algorithm a zero-phase low pass filter is applied to the estimated load error. A low pass filtering can also be obtained by increasing λ in this case but the cut-off frequency and roll-off of the low pass filtering depend on the nominal model and can hence not be chosen freely. Instead a second order low pass-filter with cut-off frequency 6.25 Hz (0.05% of the Nyquist frequency) is applied to the estimated load angle error. Since the filtering is carried out off-line it can be done as a zero-phase filtering. The evolution of the error is shown in Figure 9.10.

9.2.5 Robustness considerations

The ILC algorithm using estimated load position has so far only been evaluated in one particular example, and it is therefore difficult to draw any general conclusions about the properties of the method. It can however still be of interest to point out some of the robustness aspects.

From Chapter 4 and Theorem 4.8 the following frequency domain sufficient convergence condition follows

$$| Q(e^{i\omega t_s}) | \quad | 1 - L(e^{i\omega t_s})T_u(e^{i\omega t_s}) | < 1 \tag{9.18}$$

where $Q(q)$ and $L(q)$ are the filters in the ILC updating equation. The matrices $\boldsymbol{Q}$ and $\boldsymbol{L}$ resulting from the optimization approach can be given a frequency domain interpretation as shown in Gunnarsson and Norrlöf (1999a,b) and the previous chapter.

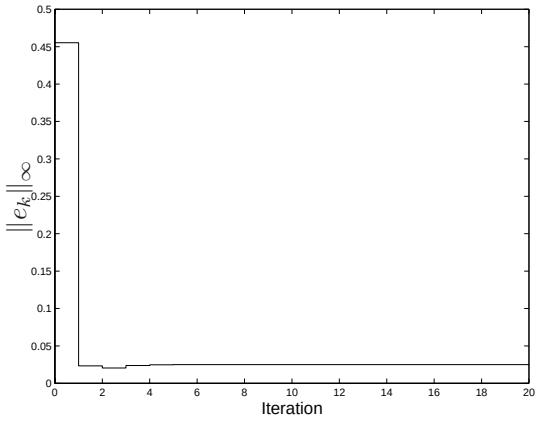


Figure 9.10 *Max error for sixth order system.*

Equation (9.18) can be obtained by considering the homogeneous part of a difference equation, in iteration index k, for the error $e_k(t)$. In the algorithm studied in this section there are two candidates for such an analysis. The most logical alternative would be to study $e_k(t) = r(t) - \theta_{a,k}(t)$, but it turns out to be simpler to study instead $\hat{e}_k(t) = r(t) - \hat{\theta}_{a,k}(t)$. If the load angle can be measured the transfer function $T_u(q)$ represents the *true* transfer function between ILC input and the load angle. This transfer function can then be written

$$T_u(q) = G_{cm}(q)G_a(q) \tag{9.19}$$

where $G_{cm}(q)$ is the transfer operator from ILC input to motor angle $\theta_m(t)$, and as before $G_a(q)$ is the transfer operator from motor angle to load angle. Correspondingly the transfer function from ILC input to estimated load angle can be written

$$\hat{T}_u(q) = G_{cm}(q)\hat{G}_a(q) \tag{9.20}$$

The estimated load angle can hence be written

$$\hat{\theta}_{a,k}(t) = G_{cm}(q)\hat{G}_a(q)u_k(t) \tag{9.21}$$

Then

$$\begin{aligned} & \hat{e}_{k+1}(t) = r(t) - \hat{T}_u(q)u_{k+1}(t) \\ = \;& r(t) - Q(q)\hat{T}_u(q)u_k(t) - Q(q)L(q)\hat{T}_u(q)\hat{e}_k(t) \\ = \;& (1 - Q(q))r(t) + Q(q)(1 - L(q)\hat{T}_u(q))\hat{e}_k(t) \end{aligned} \tag{9.22}$$

Hence the corresponding convergence criterion becomes

$$\mid Q(e^{i\omega t_s}) \mid\mid 1 - L(e^{i\omega t_s})\hat{T}_u(e^{i\omega t_s}) \mid < 1 \tag{9.23}$$

To illustrate this condition Figure 9.11 shows the left hand side of (9.18) together with the left hand side of (9.23) when $\hat{J}_a = J_a$, $\hat{J}_a = 1.2 \cdot J_a$ and when $\hat{J}_a = 0.8 \cdot J_a$. Further evaluation of the criterion predicts that $\hat{J}_a$ has to be chosen in the interval $0.75 \cdot J_a < \hat{J}_a < 1.4 \cdot J_a$ in order to satisfy equation (9.23). This prediction agrees very well with simulations. By reducing the performance requirements the stable interval for $\hat{J}_a$ is increased.

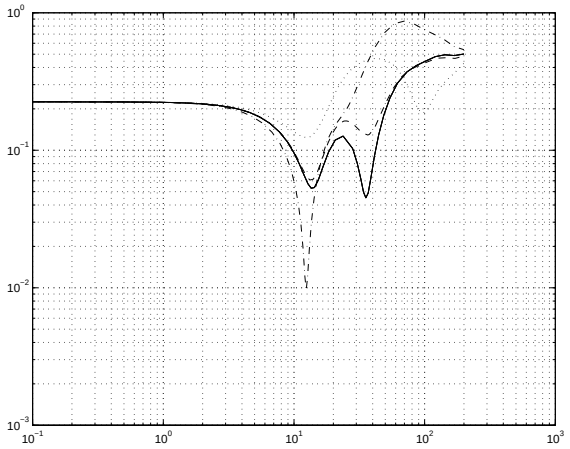


Figure 9.11 *Left hand side of equation* (9.18) *(solid). Left hand side of* (9.23) *with* $\hat{J}_a = J_a$ *(dashed),* $\hat{J}_a = 0.8J_a$ *(dash-dotted) and* $\hat{J}_a = 1.2J_a$ *(dotted).*

9.3 Conclusions and Comments

In the the first part of the chapter some limitations of the ILC approach presented in the thesis are discussed and it is shown in a practical example how the proposed ILC scheme actually can perform worse than not applying the method at all. In the second part of the chapter a possible solution to the problem is presented when ILC is applied to a flexible mechanical system. An ILC algorithm that uses estimated load angle shows promising properties in simulations. The load angle is estimated using a Kalman filter where the load acceleration is regarded as system output and motor angle is regarded as system input.

The results are based on an idealized description of the accelerometer and further work is needed in order to evaluate the sensitivity against measurement and load disturbances. Since the considered model is very simple it is of interest to investigate if the approach can be used on systems with higher order dynamics.

Part IV

Second Order ILC

10

Background

Much of the work on ILC in the research community has, so far, been devoted to first order ILC. In the first order ILC, knowledge from only the previous iteration is used explicitly in updating the control signal, see e.g., Mita and Kato (1985), Craig (1988), Horowitz (1993), de Roover (1996a), and Chapters 3 and 4. Higher order ILC, i.e., using explicitly information not only from the previous iteration but also from two or more of the previous iterations, has not been analyzed so much in literature, although there exist some examples in, e.g., Bien and Huh (1989), Chen et al. (1998) and Bien and Xu (1998). In Chen et al. (1998); Chen and Wen (1999) it is shown that using a higher order ILC can give better performance with respect to convergence speed and robustness when applied to a single robot arm. In this part of the thesis the properties of a second order ILC updating law applied to LTI systems are thoroughly investigated. The results presented here are based on Norrlöf and Gunnarsson (1999) and on Norrlöf (2000b,c).

10.1 Problem Formulation

The system description follows the one in the ILC tracking formulation in Section 3.1. Since only LTI systems will be considered, the system can be described by

$$y_k(t) = T_r(q)r(t) + T_u(q)u_k(t) \tag{10.1}$$

where $y(t)$, $r(t)$, and $u(t)$ are the output signal, the reference signal, and the input signal respectively. The transfer operators $T_r(q)$ and $T_u(q)$ are assumed to be rational functions in the delay operator q, representing the transfer operators from reference and control input respectively. The results presented here are for the disturbance free case which explains why the system disturbance and the measurement disturbance are omitted in (10.1).

The system is assumed to be of single input, single output (SISO) type, i.e., $y(t)$, $u(t)$, and $r(t)$ are all scalars. As usual the reference signal, $r(t)$, is defined over an interval $[0, t_f]$ and the system repeats the same exercise over and over again, i.e., starting from the same initial conditions and following the same reference signal repeatedly.

As was shown in Section 3.4, considering only linear operations a general high order ILC updating formula can be expressed as,

$$u_{k+1}(t) = \sum_{j=k-N+1}^{k} \Big(Q_{k-j+1}(q)\big(u_k(t) + L_{k-j+1}(q)e_j(t)\big)\Big) \tag{10.2}$$

where N is the order of the ILC. This general form of updating equation is often simplified to the case where u_{k+1} is a function of data from only the previous iteration, i.e., $r(t)$, $y_k(t)$ and $u_k(t)$, with $t \in [0, t_f]$. Let the error be defined as $e_k(t) = r(t) - y_k(t)$, from Section 3.4 the well known first order ILC updating equation is formulated as,

$$u_{k+1}(t) = Q(q)(u_k(t) + L(q)e_k(t))$$

which implies that in (10.2), $N = 1$, $Q_1(q) = Q(q)$ and $L_1(q) = L(q)$.

The second order ILC updating formula that will be considered here is given by,

$$u_{k+1}(t) = Q_1(q)\big(u_k(t) + L_1(q)e_k(t)\big) + Q_2(q)\big(u_{k-1}(t) + L_2(q)e_{k-1}(t)\big) \tag{10.3}$$

In the general formulation (10.2) this means that $N = 2$.

Using the ILC updating formulas presented above together with the system description in (10.1) it is possible to formulate a criterion for the stability of the ILC system. In Section 4.2 the result for the first order ILC applied to the system in (10.1) is given as Corollary 4.6 in the time domain, and Theorem 4.8 in the frequency domain. The stability for the second order ILC case will be discussed in the next section.

10.2 Stability Properties

As was pointed out in Chapter 4 the general results from Section 4.1 can be used for the analysis of second order ILC systems. The first result that will be explored here is the stability for higher order ILC using the notion of linear iterative systems from Section 4.1.

10.2.1 Stability results formulated in the frequency domain

Using the traditional first order ILC updating formula on the system described by (10.1) the well known stability criterion from Section 4.2 becomes,

$$|1 - L(e^{i\omega})T_u(e^{i\omega})| < |Q^{-1}(e^{i\omega})|, \quad \forall\omega \tag{10.4}$$

where the Q-filter can be used to increase the stability region. From the results in Section 4.2 it follows that when $Q \neq 1$ the input signal, $u_k(t)$, will no longer be guaranteed to converge to the input signal $u_d(t)$ that gives a zero error.

When using the second order ILC updating formula (10.3) on the system described by (10.1) the updating equation becomes,

$$\begin{aligned} u_{k+1}(t) =& Q_1(q)\big((1 - L_1(q)T_u(q))u_k(t)\big) + Q_2(q)\big(1 - L_2(q)T_u(q)u_{k-1}(t)\big) + \\ & \big(Q_1(q)L_1(q) + Q_2(q)L_2(q)\big)\big(1 - T_r(q)\big)r(t) \end{aligned} \tag{10.5}$$

and by choosing

$$\begin{aligned} z_k(t) &= \begin{bmatrix} u_k(t) \\ u_{k-1}(t) \end{bmatrix}, \quad F(q) = \begin{bmatrix} Q_1(q)(1 - L_1(q)T_u(q)) & Q_2(q)(1 - L_2(q)T_u(q)) \\ 1 & 0 \end{bmatrix} \\ F_r(q) &= \begin{bmatrix} (Q_1(q)L_1(q) + Q_2(q)L_2(q))(1 - T_r(q)) \\ 0 \end{bmatrix} \end{aligned} \tag{10.6}$$

as in Section 4.2, it is possible to apply the general stability result of Theorem 4.5 on the second order case. The stability criterion can be expressed as: the system is stable if the eigenvalues of $F(e^{i\omega})$ have an absolute value less than one. The special structure of the matrix F will be explored in the next section.

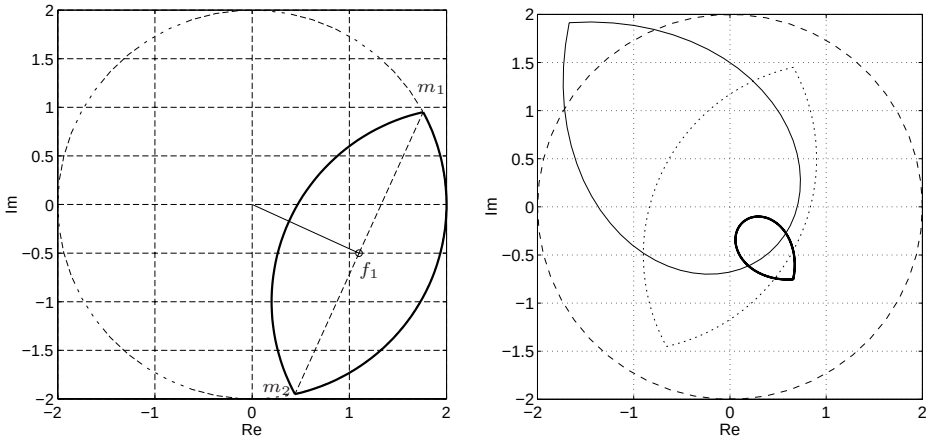
10.2.2 Analysis based on the F-matrix structure

The matrix F corresponding to the linear iterative system studied here has a companion matrix structure. This is now going to be explored.

Stability bounds on the elements in a companion matrix

In this section the special structure of the matrix F will be explored and by a geometrical discussion it will be possible to find bounds on the elements such that the eigenvalues of the matrix,

$$F = \begin{bmatrix} f_1 & -f_2 \\ 1 & 0 \end{bmatrix}, \quad f_1, f_2 \in \mathbb{C} \tag{10.7}$$



(a) Diagram showing $S_{\vee} + f_1$ (thick line) when $f_1 = 1.2 - 0.5i$.

(b) Diagram of $S_{\vee}$ (dotted), the bounds on $f_1^2 - 4f_2$ (thin), and the set S_{f_2} (thick line).

Figure 10.1 *An example showing some bounds related to the stability of the F-matrix (that will give $\rho(F) < 1$).*

have an absolute value less than one, i.e., the spectral radius $\rho(F) < 1$. The eigenvalues of the F-matrix can easily be calculated from the characteristic equation,

$$\lambda^2 - f_1\lambda + f_2 = 0 \Rightarrow 2\lambda = f_1 \pm \sqrt{f_1^2 - 4f_2} \tag{10.8}$$

The assumption that $|\lambda| < 1$ leads immediately to the following condition for stability,

$$\left|f_1 \pm \sqrt{f_1^2 - 4f_2}\right| < 2 \tag{10.9}$$

and the eigenvalues can be found using (10.8).

Assume now that f_1 is known and that a set, S_{f_2}, should be found such that when f_2 is chosen in this set the inequality in (10.9) is satisfied. Here, a geometrical approach will be used in order to find this set. Consider Figure 10.1(a) where an example of a possible f_1 is shown together with the bound on the square root expression in (10.9). The set is obviously determined by the circle segment reaching from m_1 to m_2, where m_1 and m_2 are calculated as

$$m_1 = f_1 + \frac{if_1}{|f_1|}\sqrt{4 - f_1^2}, \quad m_2 = f_1 - \frac{if_1}{|f_1|}\sqrt{4 - f_1^2} \tag{10.10}$$

The set that gives the bound on the square root expression in (10.9) can now be expressed as,

$$S_{\sqrt{\cdot}} = \{x \in \mathbb{C} \mid |x| < |2e^{iv} - f_1|,\ v \in [\arg m_2, \arg m_1]\} \tag{10.11}$$

It is also possible to write down a description of the set in which f_2 has to be chosen in order to have $\rho(F) < 1$. This set will be called the *stabilizing set* and it can be described by,

$$S_{f_2} = \{x \in \mathbb{C} \mid |x - \frac{f_1^2}{4}| < |e^{2iv} - f_1 e^{iv} + \frac{f_1^2}{4}|,\ v \in [\arg m_2, \arg m_1]\} \tag{10.12}$$

In Figure 10.1(b) the set $S_{\sqrt{\cdot}}$ in (10.11) is shown as the area closed by the dotted curve and the set,

$$S_{(\cdot)} = f(S_{\sqrt{\cdot}}),\ f(x) = x^2 \tag{10.13}$$

is depicted as the region bounded by the thin solid line. The stabilizing set that f_2 has to be chosen from, S_{f_2}, is bounded by the thick solid line shown in Figure 10.1(b). Note that in this particular case zero is not part of S_{f_2} which means that $|f_2| = 0$ will not give $\rho(F) < 1$.

Now some further results on the particular structure of F will be derived.

Eigenvectors of a 2×2 companion matrix

Consider the matrix in (10.7) again,

$$F = \begin{bmatrix} f_1 & -f_2 \\ 1 & 0 \end{bmatrix}$$

where f_1 and f_2 are complex numbers. According to the result in Theorem 4.A.10 the matrix can be made diagonal if and only if the eigenvalues are distinct. In this case the eigenvectors v_1 and v_2, corresponding to each of the eigenvalues λ_1 and λ_2, fulfill the following relation

$$Fv_i = \lambda_i v_i,\ i = 1, 2$$

By expanding this relation it is possible to find the eigenvectors as

$$\begin{bmatrix} f_1 & -f_2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} f_1 x_1 - f_2 x_2 \\ x_1 \end{bmatrix} = \lambda_i \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{10.14}$$

which implies that an eigenvector of F is given by

$$v_i = \kappa_i \begin{bmatrix} \lambda_i \\ 1 \end{bmatrix} \tag{10.15}$$

The other case that has to be considered is when $\lambda_1 = \lambda_2$. For the matrix F this happens when $f_2 = \frac{f_1^2}{4}$ and the eigenvector corresponding to the eigenvalue λ is the same as in (10.15). The generalized eigenvector can be calculated as

$$\begin{bmatrix} f_1 & -f_2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} f_1 x_1 - f_2 x_2 \\ x_1 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \kappa \begin{bmatrix} \lambda \\ 1 \end{bmatrix} \tag{10.16}$$

and therefore $x_1 = \lambda x_2 + \kappa$. By choosing, e.g., $x_2 = 0$ the generalized eigenvector becomes,

$$v_2 = \begin{bmatrix} \kappa \\ 0 \end{bmatrix} \tag{10.17}$$

The vector v_2 is therefore mapped by F according to $Fv_2 = \lambda v_2 + v_1$ which can have a bigger absolute value than v_2 (although $\lambda < 1$). For some M, however, $|F^M v_2| < |v_2|$ since Theorem 4.5 applies.

Note that, in the case of two distinct eigenvalues, a general vector in $\mathbb{C}^2$,

$$r = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} \tag{10.18}$$

can be written as a linear combination of the eigenvectors. To do this two constants κ_1 and κ_2 have to be chosen such that $r = \kappa_1 v_1 + \kappa_2 v_2$. If there is only one eigenvalue the vector r can be written as a linear combination of the eigenvector and the generalized eigenvector.

11

Convergence Properties

Using the results from the previous chapter it is possible to derive some results on the behavior of as well as the convergence speed for a second order ILC algorithm.

11.1 Background

First recall the description of the linear iterative system in the frequency domain from Section 4.1.3,

$$Z_{k+1}(\omega) = F(e^{i\omega})Z_k(\omega) + F_r(e^{i\omega})R(\omega) \tag{11.1}$$

From (10.6) it follows that

$$F(e^{i\omega}) = \begin{bmatrix} \times & \times \\ 1 & 0 \end{bmatrix},\ F_r(e^{i\omega}) = \begin{bmatrix} \times \\ 0 \end{bmatrix} \tag{11.2}$$

where the positions marked by $\times$ in the matrices might be different from zero. Instead of considering the system in (11.1) the following homogeneous system will be considered,

$$\tilde{Z}_{k+1,\omega} = F(e^{i\omega})\tilde{Z}_{k,\omega} \tag{11.3}$$

where $\tilde{Z}_{k,\omega}$ is defined as,

$$\tilde{Z}_{k,\omega} = Z_\infty(\omega) - Z_k(\omega) \tag{11.4}$$

In the above it is assumed that

$$\sup_\omega \rho(F(e^{i\omega})) < 1 \tag{11.5}$$

and that $Z_\infty(\omega)$ is defined as $\lim_{k\to\infty} Z_k(\omega)$, i.e., the limit value from Corollary 4.1 (since (11.5) guarantees convergence) transformed into the frequency domain.

11.2 Behavior of a Second Order ILC Updating Law

Using the definition of $Z_k(\omega)$ from (10.6) it is obvious that $\tilde{Z}_{k,\omega}$ becomes

$$\tilde{Z}_{k,\omega} = \begin{bmatrix} \tilde{U}_{k,\omega} \\ \tilde{U}_{k-1,\omega} \end{bmatrix} \tag{11.6}$$

with

$$\tilde{U}_{k,\omega} = U_\infty(\omega) - U_k(\omega) \tag{11.7}$$

The initial value, $\tilde{Z}_{0,\omega}$, is defined as

$$\tilde{Z}_{0,\omega} = \tilde{U}_{0,\omega} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{11.8}$$

where $\tilde{U}_{0,\omega} = U_\infty(\omega)$. This follows from the assumption that U_0 and U_{-1} both are equivalent to zero.

To study and understand the behavior of a second order ILC algorithm applied to an LTI system consider (11.3) for a fixed ω. Let $F_\omega = F(e^{i\omega})$ have the particular structure given by (11.2). According to the results in Section 10.2.2 F_ω has either one or two eigenvectors depending upon whether the matrix F_ω has one or two distinct eigenvalues.

The analysis is divided into the two cases, starting with the case when F_ω has two distinct eigenvalues.

11.2.1 F_ω has two distinct eigenvalues

When the matrix F_ω has two distinct eigenvalues, $\lambda_{1,\omega}$ and $\lambda_{2,\omega}$, it has also two eigenvectors,

$$e_{1,\omega} = \begin{bmatrix} \lambda_{1,\omega} \\ 1 \end{bmatrix},\ e_{2,\omega} = \begin{bmatrix} \lambda_{2,\omega} \\ 1 \end{bmatrix} \tag{11.9}$$

as was shown in Section 10.2.2. The initial value $\tilde{Z}_0(e^{i\omega})$ can now be expressed using the eigenvectors as

$$\tilde{Z}_{0,\omega} = \tilde{U}_{0,\omega}(\kappa_{1,\omega}e_{1,\omega} + \kappa_{2,\omega}e_{2,\omega}) \tag{11.10}$$

where $\tilde{U}_{0,\omega}, \kappa_{1,\omega}, \kappa_{2,\omega} \in \mathbb{C}$. From the definition of $\tilde{Z}_{0,\omega}$ in (11.8) it is possible to calculate $\kappa_{1,\omega}$ and $\kappa_{2,\omega}$ from

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \kappa_{1,\omega}e_{1,\omega} + \kappa_{2,\omega}e_{2,\omega} \tag{11.11}$$

Solving for $\kappa_{i,\omega}$ gives

$$\kappa_{1,\omega} = \frac{1 - \lambda_{2,\omega}}{\lambda_{1,\omega} - \lambda_{2,\omega}}, \quad \kappa_{2,\omega} = \frac{\lambda_{1,\omega} - 1}{\lambda_{1,\omega} - \lambda_{2,\omega}} \tag{11.12}$$

Using the fact that the eigenvectors of the matrix F_ω are mapped according to $F_\omega e_{i,\omega} = \lambda_{i,\omega}e_{i,\omega}$, and the definitions in (11.3) and (11.10) it follows that

$$\begin{aligned} \tilde{Z}_{k,\omega} = F_\omega^k \tilde{Z}_{0,\omega} &= \tilde{U}_{0,\omega}F_\omega^k(\kappa_{1,\omega}e_{1,\omega} + \kappa_{2,\omega}e_{2,\omega}) \\ &= \tilde{U}_{0,\omega}(\kappa_{1,\omega}\lambda_{1,\omega}^k e_{1,\omega} + \kappa_{2,\omega}\lambda_{2,\omega}^k e_{2,\omega}) \end{aligned} \tag{11.13}$$

The following useful result can now be formulated.

Theorem 11.1 (Eigenvalue decomposition of $\tilde{U}_{k,\omega}$)
It is possible to write $\tilde{U}_{k,\omega}$ in the following decomposed form,

$$\tilde{U}_{k,\omega} = \tilde{U}_{0,\omega}(\kappa_{1,\omega}\lambda_{1,\omega}^{k+1} + \kappa_{2,\omega}\lambda_{2,\omega}^{k+1}) \tag{11.14}$$

where $\tilde{U}_{0,\omega}$ is defined according to (11.7), *$\kappa_{1,\omega}$, $\kappa_{2,\omega}$ are given by* (11.12) *and, finally, $\lambda_{1,\omega}$, $\lambda_{2,\omega}$ are the eigenvalues of the matrix F_ω.*

Proof Follows from (11.13) and the definition in (11.6). ■

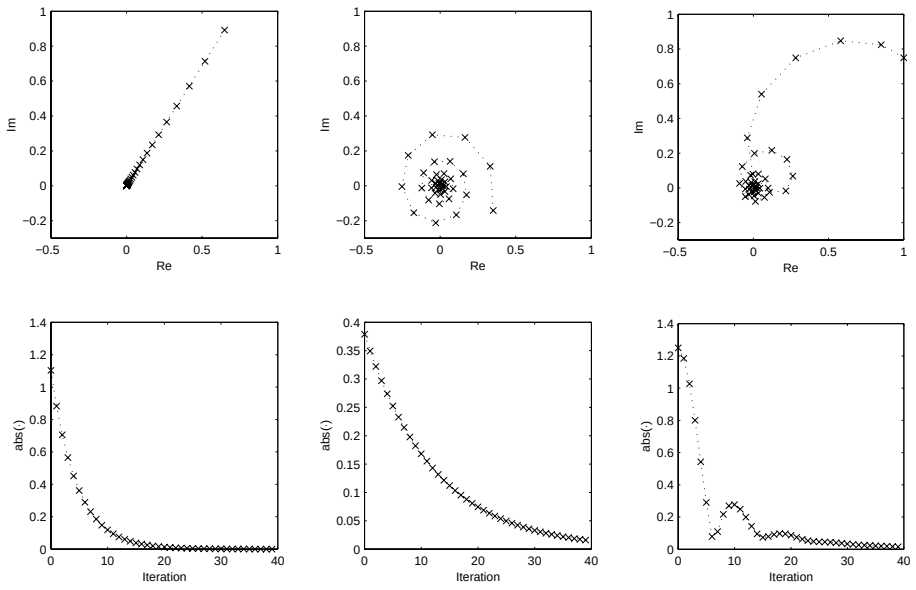
From Theorem 11.1 it is clear that the effect of the two eigenvalues can be considered separately. Each of the two eigenvalues contributes with a term,

$$\kappa_{i,\omega}\tilde{U}_{0,\omega}\lambda_{i,\omega}^{k+1} = |\kappa_{i,\omega}\tilde{U}_{0,\omega}||\lambda_{i,\omega}|^{k+1}e^{i((k+1)\arg\lambda_{i,\omega} + \arg\kappa_{i,\omega} + \arg\tilde{U}_{0,\omega})} \tag{11.15}$$

In the complex plane the different terms can be considered as functions of iteration, k, as will be shown in the following example.

Example 11.1 Behavior of a linear iterative system

Assume $\tilde{U}_{0,\omega} = 1+0.75i$, $\lambda_{1,\omega} = 0.8$, and $\lambda_{2,\omega} = 0.7+0.6i$. The resulting eigenvalue decomposition, according to Theorem 11.1, gives two terms. The evolution over the iterations is shown for each of the two components in Figure 11.1(a) and Figure 11.1(b), respectively. When $\lambda_{i,\omega}$ is positive and real the behavior becomes as shown in Figure 11.1(a) while for a complex eigenvalue the behavior will instead be as depicted in Figure 11.1(b). The resulting $\tilde{U}_{k,\omega}$ is shown in Figure 11.1(c). ❑



(a) Real eigenvalue, $\lambda = 0.8$.

(b) Complex eigenvalue, $\lambda = 0.7 + 0.6i$.

(c) Sum of the contribution from one real eigenvalue, $\lambda_{1,\omega} = 0.8$, and one complex eigenvalue, $\lambda_{2,\omega} = 0.7 + 0.6i$.

Figure 11.1 *Example of the behavior of $\tilde{U}_{0,\omega}\lambda_{i,\omega}^{k+1}$ for a real and a complex eigenvalue when $\tilde{U}_{0,\omega} = 1+0.75i$. The upper row shows the value of $\tilde{U}_{k,\omega}$ and the lower row the correspondent absolute value as a function of iteration.*

The dotted lines connecting the '×' in Figure 11.1 have been added for readability, the iterative system is only well defined for integer k.

An important remark from the example is that the absolute value of $\tilde{U}_{k,\omega}$ is not decreasing monotonously for this choice of eigenvalues. Here this is shown for a particular frequency but this can also be observed when considering the 2-norm or the energy of all the signal, see for example Chapters 6 and 18 in Bien and Xu (1998) or Chen et al. (1998).

11.2.2 F_ω has only one distinct eigenvalue

When the matrix F_ω has only one distinct eigenvalue a result corresponding to Theorem 11.1 can be formulated in the following way.

Corollary 11.1
When F_ω has only one distinct eigenvalue $\tilde{U}_{k,\omega}$ can be written as

$$\tilde{U}_{k,\omega} = \tilde{U}_{0,\omega}\lambda_\omega^k(k(1-\lambda_\omega)+1) \tag{11.16}$$

Proof The result follows from Theorem 11.1 by letting $\lambda_{1,\omega} = \lambda$ and $\lambda_{2,\omega} = \lambda+\epsilon$. Using (11.12), (11.14) and the definitions of $\lambda_{1,\omega}$ and $\lambda_{2,\omega}$ it is possible to write,

$$\frac{\tilde{U}_{k,\omega}}{\tilde{U}_{0,\omega}} = \frac{\epsilon+\lambda-1}{\epsilon}\lambda^{k+1} + \frac{1-\lambda}{\epsilon}(\lambda+\epsilon)^{k+1} = \lambda^{k+1} + \frac{\lambda-1}{\epsilon}(\lambda^{k+1} - (\lambda+\epsilon)^{k+1}) \tag{11.17}$$

The power series expansion of $(\lambda+\epsilon)^{k+1}$ is

$$\begin{aligned}(\lambda+\epsilon)^{k+1} = \lambda^{k+1}(1+\frac{\epsilon}{\lambda})^{k+1} = \lambda^{k+1}\Big(1 + (k+1)\frac{\epsilon}{\lambda} + \frac{(k+1)k}{2!}\Big(\frac{\epsilon}{\lambda}\Big)^2 + \dots \\ + \binom{k+1}{m}\Big(\frac{\epsilon}{\lambda}\Big)^m + \dots + \Big(\frac{\epsilon}{\lambda}\Big)^{k+1}\Big)\end{aligned} \tag{11.18}$$

where $0 \le m \le k+1$.

Now let $\epsilon \to 0$ in (11.17) while using the result from (11.18),

$$\begin{aligned}\lim_{\epsilon\to 0}\Big(\lambda^{k+1} + \frac{\lambda-1}{\epsilon}(\lambda^{k+1} - (\lambda+\epsilon)^{k+1})\Big) &= \lambda^k(\lambda - (\lambda-1)(k+1)) \\ &= \lambda^k(1+k(1-\lambda))\end{aligned} \tag{11.19}$$

■

It is possible to derive Corollary 11.1 using the eigenvector decomposition methodology, used in the previous section for the case where F_ω had two distinct eigenvalues. In this case there will be only one eigenvector, cf. Section 10.2.2. This single eigenvector corresponds to the eigenvalue, λ_ω, and becomes

$$e_\omega = \begin{bmatrix}\lambda_\omega \\ 1\end{bmatrix} \tag{11.20}$$

The generalized eigenvector, $e_{g,\omega}$, becomes

$$e_{g,\omega} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{11.21}$$

$\tilde{Z}_{0,\omega}$ can now be expressed in the basis consisting of the eigenvector and the generalized eigenvector

$$\tilde{Z}_{0,\omega} = \tilde{U}_{0,\omega}(\kappa_{1,\omega} e_\omega + \kappa_{2,\omega} e_{g,\omega}) \tag{11.22}$$

From (11.8) it follows that

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \kappa_{1,\omega} e_\omega + \kappa_{2,\omega} e_{g,\omega} \tag{11.23}$$

and using (11.20) and (11.21) the parameters $\kappa_{1,\omega}$ and $\kappa_{2,\omega}$ can easily be calculated,

$$\kappa_{1,\omega} = 1,\ \kappa_{2,\omega} = 1 - \lambda_\omega \tag{11.24}$$

The generalized eigenvector is mapped as $F_\omega e_{g,\omega} = \lambda_\omega e_{g,\omega} + e_\omega$ and the eigenvector is mapped according to $F_\omega e_\omega = \lambda_\omega e_\omega$. The definition in (11.3) now gives

$$\begin{aligned} \tilde{Z}_{k,\omega} &= F_\omega^k \tilde{Z}_{0,\omega} = \tilde{U}_{0,\omega} F_\omega^k (\kappa_{1,\omega} e_\omega + \kappa_{2,\omega} e_{g,\omega}) \\ &= \tilde{U}_{0,\omega} \lambda_\omega^{k-1} ((\kappa_{1,\omega}\lambda_\omega + k\kappa_{2,\omega}) e_\omega + \kappa_{2,\omega}\lambda_\omega e_{g,\omega}) \end{aligned} \tag{11.25}$$

From (11.25) and the definition in (11.6) it follows that,

$$\tilde{U}_{k,\omega} = \tilde{U}_{0,\omega}\lambda_\omega^k(\lambda_\omega + k(1-\lambda_\omega) + (1-\lambda_\omega)) = \tilde{U}_{0,\omega}\lambda_\omega^k(k(1-\lambda_\omega)+1) \tag{11.26}$$

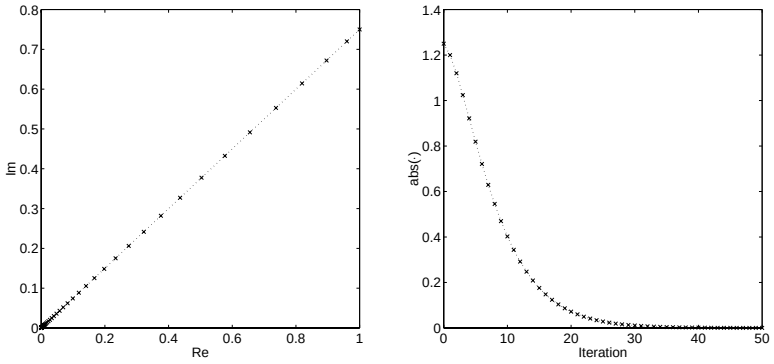
where also $\kappa_{i,\omega}$ from (11.24) is used. This is obviously the same result as given in Corollary 11.1.

To show two typical cases for the behavior of the iterative system when there is only one distinct eigenvalue consider the following example.

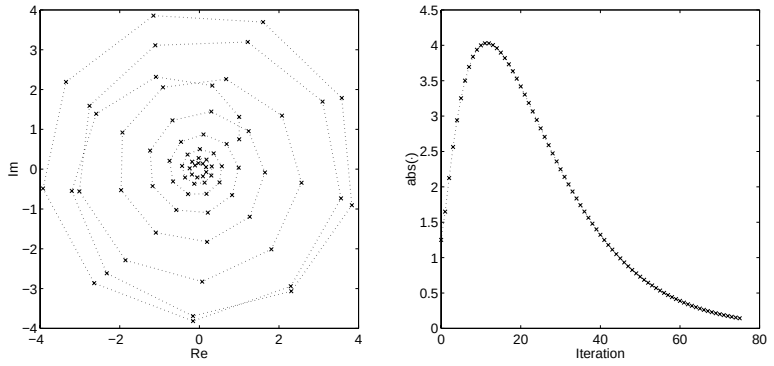
Example 11.2 Behavior of Linear Iterative Systems

Assume first that the single eigenvalue of the iterative system is real and positive, $\lambda = 0.8$. The behavior at the frequency ω for this choice of eigenvalue is shown in Figure 11.2(a). As a second case assume the eigenvalue to be $\lambda_\omega = 0.7 + 0.6i$. The behavior for this choice of eigenvalue is shown in Figure 11.2(b). ❑

Note that also here it is not sure that the absolute value of $\tilde{U}_{k,\omega}$ is monotonously decreasing, cf. Example 11.1. It is also worth to note that the behavior for the individual components in Example 11.1 is similar to the behavior in Example 11.2. The difference of behavior is explained by the term $k(1-\lambda_\omega)$. In the next section the absolute value of the linear iterative systems will be studied in more detail.



(a) Example when the eigenvalue is real, $\lambda_\omega = 0.8$. In the left plot $\tilde{U}_{k,\omega}$ is shown in the complex plane as a function of iteration k. The right plot shows $|\tilde{U}_{k,\omega}|$.



(b) Example when the eigenvalue is complex, $\lambda_\omega = 0.7 + 0.6i$. In the left plot $\tilde{U}_{k,\omega}$ is shown in the complex plane as a function of iteration k. The right plot shows $|\tilde{U}_{k,\omega}|$

Figure 11.2 *Different examples of the behavior of $\tilde{U}_{k,\omega}$ for different values of the eigenvalue, λ_ω when $\tilde{U}_{0,\omega} = 1 + 0.75i$.*

11.3 Transient Behavior for $|\tilde{U}_{k,\omega}|$

The most important property of a linear iterative system is stability. When stability is ensured the transient behavior becomes equally important. In the previous section some examples of the behavior of linear iterative systems are shown. They are all stable systems but the transient behavior of $|\tilde{U}_{k,\omega}|$ varies a lot.

For a typical application the value of $|\tilde{U}_{k,\omega}|$ is not allowed to have an overshoot as is the case in the example in Figure 11.2(b). In this section the transient behavior of $|\tilde{U}_{k,\omega}|$ is going to be addressed. This will lead to some basic rules on how to choose the eigenvalues of the matrix F_ω in order to achieve a good behavior of $|\tilde{U}_{k,\omega}|$.

11.3.1 Behavior of $|\tilde{U}_{k,\omega}|$ when F_ω has two distinct eigenvalues

From the analysis of the behavior of $\tilde{U}_{k,\omega}$ in the previous section it is obvious that at least for some choices of eigenvalues, cf. Example 11.1, the value $|\tilde{U}_{k,\omega}|$ does not converge monotonously to zero. Instead an oscillatory behavior can be recognized. To understand this consider the following way of exactly representing $\frac{|\tilde{U}_{k,\omega}|^2}{|\tilde{U}_{0,\omega}|^2}$,

$$\begin{aligned}
&\frac{|\tilde{U}_{k,\omega}|^2}{|\tilde{U}_{0,\omega}|^2} = \left|\kappa_{1,\omega}\lambda_{1,\omega}^{k+1} + \kappa_{2,\omega}\lambda_{2,\omega}^{k+1}\right|^2 = \\
&\quad \Big||\kappa_{1,\omega}||\lambda_{1,\omega}|^{k+1}(\cos((k+1)\arg\lambda_{1,\omega} + \arg\kappa_{1,\omega}) + i\sin((k+1)\arg\lambda_{1,\omega} + \arg\kappa_{1,\omega})) + \\
&\quad |\kappa_{2,\omega}||\lambda_{2,\omega}|^{k+1}(\cos((k+1)\arg\lambda_{2,\omega} + \arg\kappa_{2,\omega}) + i\sin((k+1)\arg\lambda_{2,\omega} + \arg\kappa_{2,\omega}))\Big|^2 \\
&= |\kappa_{1,\omega}|^2|\lambda_{1,\omega}|^{2(k+1)} + |\kappa_{2,\omega}|^2|\lambda_{2,\omega}|^{2(k+1)} + \\
&\quad 2|\kappa_{1,\omega}\kappa_{2,\omega}||\lambda_{1,\omega}\lambda_{2,\omega}|^{k+1}\cos((k+1)(\arg\lambda_{1,\omega} - \arg\lambda_{2,\omega}) + \arg\kappa_{1,\omega} - \arg\kappa_{2,\omega})
\end{aligned} \tag{11.27}$$

where the cause of the oscillation becomes evident. The value of the cos-factor depends on $\arg\kappa_{i,\omega}$ for $i = 1, 2$. From the definition of $\kappa_{i,\omega}$ we get

$$\begin{aligned}
\arg\kappa_{1,\omega} - \arg\kappa_{2,\omega} &= \arg\frac{1-\lambda_{2,\omega}}{\lambda_{1,\omega}-\lambda_{2,\omega}} - \arg\frac{\lambda_{1,\omega}-1}{\lambda_{1,\omega}-\lambda_{2,\omega}} \\
&= \arg(1-\lambda_{2,\omega}) - \arg(\lambda_{1,\omega}-1)
\end{aligned} \tag{11.28}$$

which shows the actual dependence on the eigenvalues. The last expression in (11.27) therefore consists of three terms where the two first are exponentially decreasing (since $\lambda_{i,\omega} < 1$) and the last term is oscillating with exponentially decreasing amplitude.

The following result can now be formulated.

Theorem 11.2
When F_ω has two distinct eigenvalues

$$\left| |\kappa_{1,\omega}||\lambda_{1,\omega}|^{k+1} - |\kappa_{2,\omega}||\lambda_{2,\omega}|^{k+1} \right| \leq \frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|} \leq |\kappa_{1,\omega}||\lambda_{1,\omega}|^{k+1} + |\kappa_{2,\omega}||\lambda_{2,\omega}|^{k+1}$$

where the expression on the lefthand side and the right hand side of the inequalities will be referred to as the lower bound *and the* upper bound *respectively.*

Proof Using (11.27) the bounds follow immediately from the fact that $-1 \leq \cos(\cdot) \leq 1$. ■

Now an example to show how this result can be applied.

Example 11.3 **A lower and an upper bound for $|\tilde{U}_{k,\omega}|$**

Consider the same choice of eigenvalues as in Example 11.1, i.e., $\lambda_{1,\omega} = 0.8$ and $\lambda_{2,\omega} = 0.7 + 0.6i$. In Figure 11.3 the normalized value of $|\tilde{U}_{k,\omega}|$ is shown together with the upper and the lower bounds. Note that the upper bound is a monotonous decreasing function of iteration while the lower bound is not. ❑

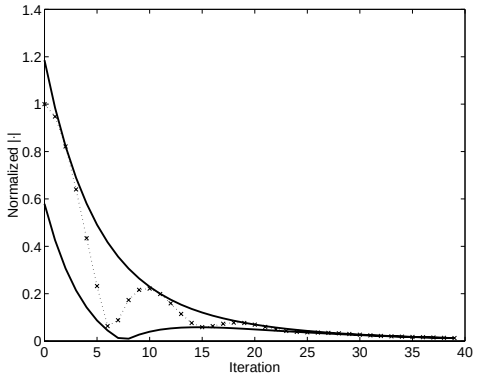


Figure 11.3 *Example of the lower and upper bound for $|\tilde{U}_{k,\omega}|$ when the eigenvalues are chosen as $\lambda_{1,\omega} = 0.8$ and $\lambda_{2,\omega} = 0.7 + 0.6i$.*

It is important to stress that, although the upper bound in Theorem 11.2 is monotonously decreasing this does not imply that the function $|\tilde{U}_{k,\omega}|$ decreases monotonously. The initial value of the upper bound contains the sum of the absolute values of $\kappa_{1,\omega}$ and $\kappa_{2,\omega}$. From (11.12) it follows that $\kappa_{2,\omega} = 1 - \kappa_{1,\omega}$ and

that

$$\kappa_{1,\omega} = \frac{1 - \lambda_{2,\omega}}{\lambda_{1,\omega} - \lambda_{2,\omega}} \tag{11.29}$$

Clearly the numerator of (11.29) is bounded according to $0 < |1 - \lambda_{2,\omega}| < 2$. Also the absolute value of the denominator is bounded by $0 < |\lambda_{1,\omega} - \lambda_{2,\omega}| < 2$, but when $\lambda_{1,\omega} \to \lambda_{2,\omega}$, or vice versa, the value of $\kappa_{1,\omega}$ and $\kappa_{2,\omega}$ will tend to infinity. This, in turn, imply that the upper bound is no longer very useful. At least for $k = 0$ where it is clear that $\frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|} = 1$.

The upper and the lower bounds for $|\tilde{U}_{k,\omega}|$ give however in most cases a great help in the analysis of the behavior of $|\tilde{U}_{k,\omega}|$. It is important from a design perspective to find conditions that the eigenvalues have to fulfill in order to give a specified behavior. The first requirement that will be formulated here is that $|\tilde{U}_{k,\omega}|$ should never be greater or equal to $|\tilde{U}_{0,\omega}|$. The following sufficient condition for $|\tilde{U}_{k,\omega}|$ to be less than $|\tilde{U}_{0,\omega}|$ is easy to find.

Corollary 11.2 (Sufficient condition for $\frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|} < 1$ for $k \in \mathbb{Z}^+$)

In the case when the matrix F_ω has two distinct eigenvalues, $\lambda_{1,\omega}$ and $\lambda_{2,\omega}$, a sufficient condition for

$$\frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|} < 1 \tag{11.30}$$

for all $k \in \mathbb{Z}^+$ is that

$$|\lambda_{1,\omega}|^2|1 - \lambda_{2,\omega}| + |\lambda_{2,\omega}|^2|\lambda_{1,\omega} - 1| < |\lambda_{1,\omega} - \lambda_{2,\omega}| \tag{11.31}$$

Proof Follows from the fact that the upper bound in Theorem 11.2 is monotonously decreasing and therefore if it is less than 1 for $k = 1$ it will be less than 1 for all $k \geq 1$. ■

Now an example showing how the result in Corollary 11.2 can be used in practice.

Example 11.4

Consider again the choice of eigenvalues from Example 11.1, i.e., $\lambda_{1,\omega} = 0.8$, and $\lambda_{2,\omega} = 0.7 + 0.6i$. In this case

$$|\lambda_{1,\omega}|^2|1 - \lambda_{2,\omega}| + |\lambda_{2,\omega}|^2|\lambda_{1,\omega} - 1| = 0.5993 < |\lambda_{1,\omega} - \lambda_{2,\omega}| = 0.6083$$

which means that it is guaranteed that

$$\frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|} < 1$$

for all $k \in \mathbb{Z}^+$. This result can also be verified in Figure 11.3. ❑

A problem with the result in Corollary 11.2 is that the condition is easily violated when $\lambda_{1,\omega}$ and $\lambda_{2,\omega}$ are close to each other, i.e., when the right hand side of the inequality in the condition is close to zero. As the condition is only sufficient the violation will, however, not guarantee that $\exists k \in \mathbb{Z}^+$ such that $|\tilde{U}_{k,\omega}| \geq |\tilde{U}_{0,\omega}|$.

11.3.2 Behavior of $|\tilde{U}_{k,\omega}|$ when F_ω has two real eigenvalues

It is possible to draw general conclusions about the behavior of the linear iterative system when it is known that the eigenvalues are real. First the case when $\operatorname{sign}\lambda_{1,\omega} \neq \operatorname{sign}\lambda_{2,\omega}$. Note that the case when there is only one distinct real eigenvalue is not covered in this section, this is instead included in the discussion in the next section.

Lemma 11.1
When $\lambda_{1,\omega},\ \lambda_{2,\omega} \in \mathbb{R}$ *and* $\operatorname{sign}\lambda_{1,\omega} \neq \operatorname{sign}\lambda_{2,\omega}$ *it is true that*

$$\frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|} < 1$$

for all $k \in \mathbb{Z}^+$.

Proof Follows from (11.27) using the fact that if $\arg\lambda_{1,\omega} = 0$ then $\arg\lambda_{2,\omega} = \pi$ and if $\arg\lambda_{1,\omega} = \pi$ then $\arg\lambda_{2,\omega} = 0$ and, accordingly, that $\arg(1-\lambda_{2,\omega}) = 0$, and $\arg(\lambda_{1,\omega}-1) = \pi$. The result becomes that

$$\begin{aligned}\frac{|\tilde{U}_{k,\omega}|^2}{|\tilde{U}_{0,\omega}|^2} = |\kappa_{1,\omega}|^2|\lambda_{1,\omega}|^{2(k+1)} + |\kappa_{2,\omega}|^2|\lambda_{2,\omega}|^{2(k+1)} \\ + 2|\kappa_{1,\omega}\kappa_{2,\omega}||\lambda_{1,\omega}\lambda_{2,\omega}|^{k+1}\cos(\sigma(k+1)\pi - \pi)\end{aligned} \tag{11.32}$$

where $\sigma = \operatorname{sign}(\lambda_{2,\omega})$. For $k = 0$ the cos-term will equal $+1$ and the expression in (11.32) will coincide with the upper bound in Theorem 11.2. Since the upper bound is a monotonously decreasing function of k it is sure that Lemma 11.1 is fulfilled for all $k \in \mathbb{Z}^+$. ■

Note that the cos-term in (11.32) when $k = 1$ will be equal to -1 which, in fact, makes the value of (11.32) equal to the lower bound in Theorem 11.2. This means that for real eigenvalues that fulfill $\operatorname{sign}\lambda_{1,\omega} \neq \operatorname{sign}\lambda_{2,\omega}$, the value of $\frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|}$ will actually oscillate between the upper bound, for even k, and the lower bound, for odd k.

Lemma 11.2
When $\lambda_{1,\omega},\ \lambda_{2,\omega} \in \mathbb{R}$ *and* $\operatorname{sign}\lambda_{1,\omega} = \operatorname{sign}\lambda_{2,\omega}$ *it is true that*

$$\frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|} = \Big||\kappa_{1,\omega}||\lambda_{1,\omega}|^{k+1} - |\kappa_{2,\omega}||\lambda_{2,\omega}|^{k+1}\Big|$$

i.e., the lower bound in Theorem 11.2.

Proof Follows from (11.27) using the fact that if $\arg\lambda_{1,\omega} = 0$ then $\arg\lambda_{2,\omega} = 0$ and if $\arg\lambda_{1,\omega} = \pi$ then $\arg\lambda_{2,\omega} = \pi$ and for all real eigenvalues $\arg(1-\lambda_{2,\omega}) = 0$, and $\arg(\lambda_{1,\omega} - 1) = \pi$. Clearly, the result becomes that

$$\begin{aligned}\frac{|\tilde{U}_{k,\omega}|^2}{|\tilde{U}_{0,\omega}|^2} &= |\kappa_{1,\omega}|^2|\lambda_{1,\omega}|^{2(k+1)} + |\kappa_{2,\omega}|^2|\lambda_{2,\omega}|^{2(k+1)} \\ &\quad + 2|\kappa_{1,\omega}\kappa_{2,\omega}||\lambda_{1,\omega}\lambda_{2,\omega}|^{k+1}\cos\pi\end{aligned} \tag{11.33}$$

and since $\cos\pi = -1$ the equality follows. ■

Note that the result in Lemma 11.2 does not imply that $\frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|} < 1$. Try for example $\lambda_{1,\omega} = -0.9$ and $\lambda_{2,\omega} = -0.1$.

The following result is a positive result showing how to choose the eigenvalues in order to be sure that the lower bound is always less than one.

Lemma 11.3
If $\lambda_{i,\omega} \in \mathbb{R}$ *and* $|\lambda_{i,\omega}| < 1$, *then it is true that*

$$\Big||\kappa_{1,\omega}||\lambda_{1,\omega}|^{k+1} - |\kappa_{2,\omega}||\lambda_{2,\omega}|^{k+1}\Big| < 1$$

for all $k \in \mathbb{Z}^+$ *iff*

$$\max\Big(\frac{|\lambda_{1,\omega}|-1}{|\lambda_{1,\omega}|+1}, -1\Big) < \lambda_{2,\omega} < 1 \tag{11.34}$$

Proof For the proof see Appendix 11.A. ■

Using Lemma 11.1 to Lemma 11.3 it is possible to formulate the following concluding result on the case when $\lambda_{1,\omega}, \lambda_{2,\omega} \in \mathbb{R}$.

Theorem 11.3 (Condition for $|\tilde{U}_{k,\omega}| < |\tilde{U}_{0,\omega}|$***)***
A necessary and sufficient condition for

$$|\tilde{U}_{k,\omega}| < |\tilde{U}_{0,\omega}|, \forall k \in \mathbb{Z}^+$$

when $\lambda_{1,\omega}, \lambda_{2,\omega} \in \mathbb{R}$, $\lambda_{1,\omega} \neq \lambda_{2,\omega}$ *is that*

$$-1 < \lambda_{1,\omega} < 1 \quad \textit{and} \quad \max\Big(\frac{|\lambda_{1,\omega}|-1}{|\lambda_{1,\omega}|+1}, -1\Big) < \lambda_{2,\omega} < 1$$

Proof Follows from Lemma 11.1, Lemma 11.2, and Lemma 11.3. ■

The regions given by Theorem 11.3 are shown in Figure 11.4.

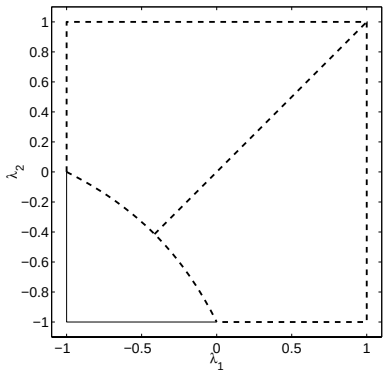


Figure 11.4 *The regions where to choose real (distinct) eigenvalues, $\lambda_{1,\omega}$ and $\lambda_{2,\omega}$, in order to have $|\tilde{U}_{k,\omega}| < |\tilde{U}_{0,\omega}|$ for all $k \in \mathbb{Z}^+$.*

11.3.3 Behavior of $|\tilde{U}_{k,\omega}|$ when F_ω has one distinct eigenvalue

When the two eigenvalues in the F_ω matrix become one, i.e., when $\lambda_{1,\omega} \to \lambda_{2,\omega}$ or vice versa, the bounds given by Theorem 11.2 become useless since

$$0 \leq \frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|} \leq \infty$$

The exact value of $|\tilde{U}_{k,\omega}|$ is easily found from Corollary 11.1,

$$|\tilde{U}_{k,\omega}| = |\tilde{U}_{0,\omega}||\lambda_\omega|^k|k(1-\lambda_\omega)+1| \tag{11.35}$$

A possible choice for the upper and lower bounds is instead given by the following theorem.

Theorem 11.4
When F_ω has only one distinct eigenvalue, then

$$|\lambda_\omega|^k \leq \frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|} \leq |\lambda_\omega|^k(k|1-\lambda_\omega|+1) \tag{11.36}$$

Proof The lower bound is found by minimizing the second factor of (11.35) by assuming $\lambda_\omega = 1$. The upper bound is simply the result of applying the triangle inequality. ■

Note that, compared to the result in Theorem 11.2, the lower bound is a monotonously decreasing function of iteration while the upper bound is not. For the special case when $\lambda_\omega \in \mathbb{R}$ the following result follows from Theorem 11.4.

Corollary 11.3
When $\lambda_\omega \in \mathbb{R}$ and $-1 < \lambda_\omega < 1$ then

$$\frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|} = |\lambda_\omega|^k(k(1-\lambda_\omega)+1)$$

i.e., the upper bound in Theorem 11.4.

Proof Follows from the fact that $1 - \lambda_\omega > 0$ for all $-1 < \lambda_\omega < 1$. ■

Now an example showing an application of Theorem 11.4.

Example 11.5 **A lower and an upper bound for $|\tilde{U}_{k,\omega}|$**

With the same choice of eigenvalues as in Example 11.2, i.e., $\lambda_\omega = 0.8$ and $\lambda_\omega = 0.7 + 0.6i$, the corresponding upper and lower bounds become as in Figure 11.5(a) and 11.5(b). ❑

According to the discussion in the previous section, it might be important that $|\tilde{U}_{k,\omega}|$ is less than $|\tilde{U}_{0,\omega}|$ for all $k \in \mathbb{Z}^+$. The following theorem gives a necessary and sufficient condition for this to be true.

Theorem 11.5 (Condition for $|\tilde{U}_{k,\omega}| < |\tilde{U}_{0,\omega}|$)
A necessary and sufficient condition for

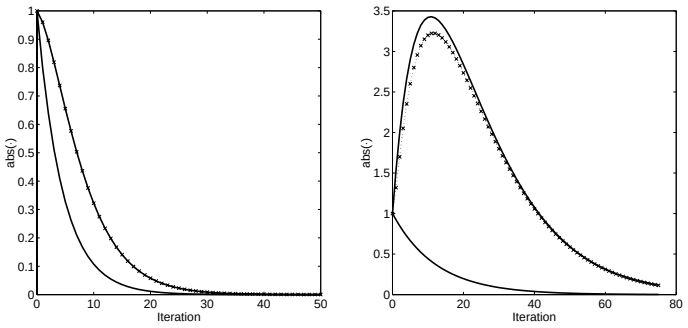
$$|\tilde{U}_{k,\omega}| < |\tilde{U}_{0,\omega}|, \ \forall k \in \mathbb{Z}^+$$

when $|\lambda_\omega| < 1$ is that

$$|2 - \lambda_\omega| < \frac{1}{|\lambda_\omega|}$$

Proof For the proof see Appendix 11.B. ■

The region for λ_ω given by Theorem 11.5 is shown in the complex plane in Figure 11.6. Note that for a real eigenvalue the theorem is fulfilled for $1-\sqrt{2} < \lambda_\omega < 1$ which is exactly the line dividing the two regions in Figure 11.4.



(a) $\lambda_\omega = 0.8$ *(b)* $\lambda_\omega = 0.7 + 0.6i$

Figure 11.5 *Example of the upper and lower bounds for* $|\tilde{U}_{k,\omega}|$ *for two different choices of eigenvalue,* λ_ω.

11.3.4 Relations between the different cases

From the results in Theorem 11.3 and Theorem 11.5 it is possible to formulate the following corollary.

Corollary 11.4 (Condition for $|\tilde{U}_{k,\omega}| < |\tilde{U}_{0,\omega}| \ \forall k \in \mathbb{Z}^+$***)***
Suppose $\lambda_{1,\omega}, \lambda_{2,\omega} \in \mathbb{R}$ *in the case of two distinct eigenvalues, or* $\lambda_\omega \in \mathbb{R}$ *for the case of one distinct eigenvalue. A necessary and sufficient condition for*

$$|\tilde{U}_{k,\omega}| < |\tilde{U}_{0,\omega}|, \ \forall k \in \mathbb{Z}^+$$

is that

$$-1 < \lambda_{1,\omega} < 1 \quad \textit{and} \quad \max\Big(\frac{|\lambda_{1,\omega}| - 1}{|\lambda_{1,\omega}| + 1}, -1\Big) < \lambda_{2,\omega} < 1$$

for two distinct eigenvalues and

$$1 - \sqrt{2} < \lambda_\omega < 1$$

for only one distinct eigenvalue.

Proof Follows from Theorem 11.3 and Theorem 11.5. ■

The region can be depicted in the λ-space according to Figure 11.7.

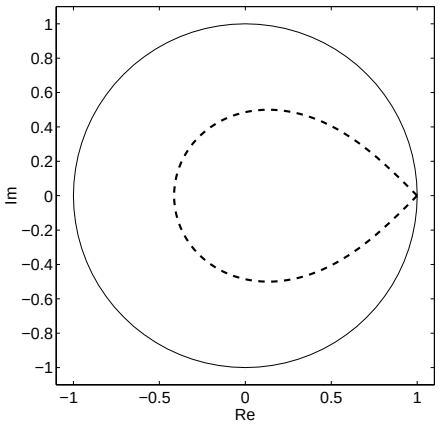


Figure 11.6 *The region in which to choose the eigenvalue, when F_ω has only one distinct eigenvalue, to ensure that $|\tilde{U}_{k,\omega}| < |\tilde{U}_{0,\omega}|$ for all $k \in \mathbb{Z}^+$.*

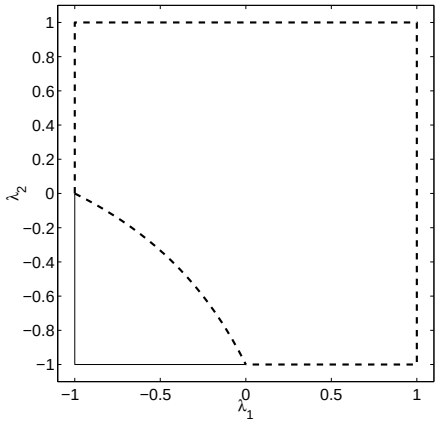


Figure 11.7 *The region in which to choose the real eigenvalues to ensure that $|\tilde{U}_{k,\omega}| < |\tilde{U}_{0,\omega}|$ for all $k \in \mathbb{Z}^+$.*

In the proof of Corollary 11.1 it becomes clear that the case with one distinct eigenvalue follows from the general case where F_ω has two distinct eigenvalues. Using this result it is possible to formulate the following robustness condition.

Theorem 11.6 (Robustness, nominal design has one distinct eigenvalue) *When the eigenvalues of the matrix F_ω can be described by*

$$\lambda_{1,\omega} = \lambda_\omega(1+\delta_{1,\omega}),\ \lambda_{2,\omega} = \lambda_\omega(1+\delta_{2,\omega})$$

where it is assumed that the relative uncertainty $\delta_{i,\omega} \in \mathbb{C}$ and $|\delta_{i,\omega}| < \gamma_\omega$ for $i = 1, 2$ it is possible to find an upper bound for $\frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|}$ according to

$$\frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|} \leq |\lambda_\omega|^k(1+\gamma_\omega)^k(k(|1-\lambda_\omega| + \gamma_\omega|\lambda_\omega|)+1)$$

Proof For the general case with two distinct eigenvalues it is true that

$$\begin{aligned}\frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|} &= \Big|\sum_{j=0}^{k}\lambda_{1,\omega}^{k-j}\lambda_{2,\omega}^{j} - \lambda_{1,\omega}\lambda_{2,\omega}\sum_{j=0}^{k-1}\lambda_{1,\omega}^{k-1-j}\lambda_{2,\omega}^{j}\Big| \\ &= |\lambda_\omega|^k\Big|\sum_{j=0}^{k}\Delta_{1,\omega}^{k-j}\Delta_{2,\omega}^{j} - \lambda_\omega\Delta_{1,\omega}\Delta_{2,\omega}\sum_{j=0}^{k-1}\Delta_{1,\omega}^{k-1-j}\Delta_{2,\omega}^{j}\Big|\end{aligned} \tag{11.37}$$

where $\Delta_{i,\omega}$ is defined as $\Delta_{i,\omega} = 1 + \delta_{i,\omega}$. Let

$$\Sigma_\omega = \sum_{j=0}^{k-1}\Delta_{1,\omega}^{k-1-j}\Delta_{2,\omega}^{j}$$

Using Σ_ω it is possible to rewrite (11.37) according to

$$\begin{aligned}|\lambda_\omega|^k\big|\Delta_{1,\omega}\Sigma_\omega + \Delta_{2,\omega}^k - \lambda_\omega\Delta_{1,\omega}\Delta_{2,\omega}\Sigma_\omega\big| &= |\lambda_\omega|^k\big|\Delta_{1,\omega}\Sigma_\omega(1-\lambda_\omega\Delta_{2,\omega}) + \Delta_{2,\omega}^k\big| \\ &\leq |\lambda_\omega|^k\big(|\Delta_{1,\omega}\Sigma_\omega||1-\lambda_\omega\Delta_{2,\omega}| + |\Delta_{2,\omega}|^k\big) \\ &\leq |\lambda_\omega|^k|1+\gamma_\omega|^k\big(k(|1-\lambda_\omega| + \gamma_\omega|\lambda_\omega|)+1\big)\end{aligned}$$

where it is used that

$$|\Delta_{i,\omega}| = |1+\delta_{i,\omega}| \leq 1+\gamma_\omega, \quad |1-\lambda_\omega\Delta_{2,\omega}| = |1-\lambda_\omega-\delta_{2,\omega}\lambda_\omega| \leq |1-\lambda_\omega| + \gamma_\omega|\lambda_\omega|$$

and

$$|\Sigma_\omega| \leq \sum_{j=0}^{k-1}|\Delta_{1,\omega}|^{k-1-j}|\Delta_{2,\omega}|^j \leq \sum_{j=0}^{k-1}(1+\gamma_\omega)^{k-1} = k(1+\gamma_\omega)^{k-1}$$

which concludes the proof. ■

From a design perspective the result in Theorem 11.6 is very important since it says something about the behavior of the second order ILC when the eigenvalues are uncertain. Note that the stability region for the result in Theorem 11.6 is given by

$$|\lambda_\omega|(1+\gamma_\omega) < 1$$

i.e., all eigenvalues in the uncertainty set have to be stable.

The following example shows how the result can be used.

Example 11.6

Assume that for a certain frequency it is known that there is a nominal eigenvalue λ_ω. The design has an uncertainty which has resulted in that the actual eigenvalues can be in a region around λ_ω given by

$$S(\lambda_\omega, \gamma_\omega) = \{\lambda_{1,\omega}, \lambda_{2,\omega} \in \mathbb{C} \mid \lambda_{i,\omega} = \lambda_\omega(1+\delta_{i,\omega}),\ |\delta_{i,\omega}| < \gamma_\omega,\ i = 1,2\}$$

Now assume $\lambda_\omega = 0.1$ and that the relative error is 100%, i.e., $\gamma_\omega = 1$. The resulting uncertainty region is shown in Figure 11.8(a) and the nominal convergence of $\frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|}$ together with the worst case convergence speed is depicted in Figure 11.8(b). ❑

The result in Theorem 11.6 gives an upper bound for the actual convergence speed when the nominal design gives only one distinct eigenvalue. Important is, of course, to find a similar result when there are two distinct eigenvalues in the nominal case and the two eigenvalues have individual uncertainties.

Theorem 11.7 (Robustness, nominal design has two distinct eigenvalues) *Suppose the eigenvalues of the matrix F_ω can be described by*

$$\lambda_{1,\omega} = \bar{\lambda}_{1,\omega} + \delta_{1,\omega}, \quad \lambda_{2,\omega} = \bar{\lambda}_{2,\omega} + \delta_{2,\omega}$$

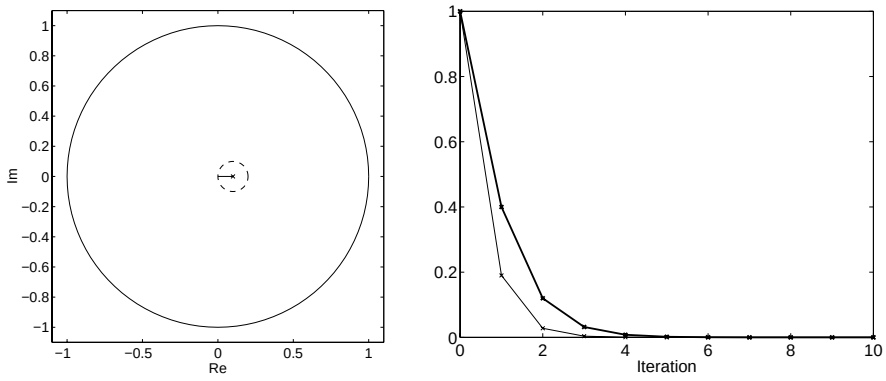
where $\bar{\lambda}_{i,\omega}$ is the nominal value and $\delta_{i,\omega}$ is a bounded uncertainty, $\delta_{i,\omega} \in \mathbb{C}$ and $|\delta_{i,\omega}| < \gamma_{i,\omega}$ for $i = 1, 2$. An upper bound for $\frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|}$ is given by

$$\frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|} \leq \frac{1}{|\bar{\lambda}_{1,\omega} - \bar{\lambda}_{2,\omega}|}\Big((|1-\bar{\lambda}_{2,\omega}| + \gamma_{2,\omega})(|\bar{\lambda}_{1,\omega}| + \gamma_{1,\omega})^{k+1} + (|\bar{\lambda}_{1,\omega} - 1| + \gamma_{1,\omega})(|\bar{\lambda}_{2,\omega}| + \gamma_{2,\omega})^{k+1}\Big)$$

were it is also assumed that the uncertainty regions for $\lambda_{i,\omega}$ do not overlap.

Proof Using the upper bound in Theorem 11.2 it follows that

$$\frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|} \leq |\kappa_{1,\omega}||\lambda_{1,\omega}|^{k+1} + |\kappa_{2,\omega}||\lambda_{2,\omega}|^{k+1}$$



(a) The nominal value of λ_ω together with the uncertainty region when the relative uncertainty is 100%.

(b) Nominal convergence of $\frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|}$ (thin) together with worst case (thick) convergence speed, given the chosen uncertainty.

Figure 11.8 *Illustration of the result in Theorem 11.6.*

where

$$|\kappa_{i,\omega}| = \frac{|1-\lambda_{i,\omega}|}{|\lambda_{1,\omega}-\lambda_{2,\omega}|} \leq \frac{|1-\bar{\lambda}_{i,\omega}|+\gamma_{i,\omega}}{|\bar{\lambda}_{1,\omega}-\bar{\lambda}_{2,\omega}|-(\gamma_{1,\omega}+\gamma_{2,\omega})}$$

since

$$\begin{aligned} |1-\lambda_{i,\omega}| &\leq |1-\hat{\lambda}_{i,\omega}|+\gamma_{i,\omega} \\ |\lambda_{1,\omega}-\lambda_{2,\omega}| &\geq |\bar{\lambda}_{1,\omega}-\bar{\lambda}_{2,\omega}|-(\gamma_{1,\omega}+\gamma_{2,\omega}) \\ |\lambda_{i,\omega}| &\leq |\hat{\lambda}_{i,\omega}|+\gamma_{i,\omega} \end{aligned}$$

and the result of the the theorem follows readily. Note that it is assumed that the regions are not overlapping since $|\bar{\lambda}_{1,\omega}-\bar{\lambda}_{2,\omega}|-(\gamma_{1,\omega}+\gamma_{2,\omega})$ is assumed to be positive. ■

One case that can be interesting to consider is a special case of the result in Theorem 11.6. This is when the eigenvalues are different but have the same argument, i.e., $\arg\lambda_{1,\omega} = \arg\lambda_{2,\omega}$. The result is given as a corollary.

Corollary 11.5
When the nominal eigenvalues fulfill the condition

$$\bar{\lambda}_{1,\omega} = \bar{\lambda}_\omega, \quad \bar{\lambda}_{2,\omega} = \alpha\bar{\lambda}_\omega$$

for $0 < \alpha < 1$ *and the true eigenvalues are in a uncertainty set described by*

$$\lambda_{1,\omega} = \bar{\lambda}_\omega(1+\delta_{1,\omega}), \quad \lambda_{2,\omega} = \alpha\bar{\lambda}_\omega(1+\delta_{2,\omega})$$

where $\delta_{i,\omega} \in \mathbb{C}$ *and* $|\delta_{i\omega}| < \gamma_\omega$ *it is possible to find an upper bound for* $\frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|}$ *according to*

$$\frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|} \leq |\bar{\lambda}_\omega|^k (1+\gamma_\omega)^k \left(\frac{\alpha^k - 1}{\alpha - 1}(|1 - \alpha\bar{\lambda}_\omega| + \alpha|\bar{\lambda}_\omega|\gamma_\omega) + \alpha^k\right)$$

Proof Following the same path as in the proof of Theorem 11.6 it is possible to arrive at a similar equation as in (11.37)

$$\begin{aligned} \frac{|\tilde{U}_{k,\omega}|}{|\tilde{U}_{0,\omega}|} &= \Big| \sum_{j=0}^{k} \lambda_{1,\omega}^{k-j} \lambda_{2,\omega}^{j} - \lambda_{1,\omega}\lambda_{2,\omega} \sum_{j=0}^{k-1} \lambda_{1,\omega}^{k-1-j} \lambda_{2,\omega}^{j} \Big| \\ &= |\lambda_\omega|^k \Big| \sum_{j=0}^{k} \Delta_{1,\omega}^{k-j} \alpha^j \Delta_{2,\omega}^{j} - \alpha\bar{\lambda}_\omega \Delta_{1,\omega}\Delta_{2,\omega} \sum_{j=0}^{k-1} \Delta_{1,\omega}^{k-1-j} \alpha^j \Delta_{2,\omega}^{j} \Big| \end{aligned} \tag{11.38}$$

where $\Delta_{i,\omega}$ is defined as $\Delta_{i,\omega} = 1 + \delta_{i,\omega}$. Now define

$$\Sigma_\omega = \sum_{j=0}^{k-1} \Delta_{1,\omega}^{k-1-j} \alpha^j \Delta_{2,\omega}^{j}$$

and rewrite (11.38) according to

$$\begin{aligned} &|\lambda_\omega|^k \Big| \Delta_{1,\omega}\Sigma_\omega + (\alpha\Delta_{2,\omega})^k - \alpha\bar{\lambda}_\omega \Delta_{1,\omega}\Delta_{2,\omega}\Sigma_\omega \Big| \\ &\quad \leq |\lambda_\omega|^k \Big(|\Delta_{1,\omega}\Sigma_\omega| \, |1 - \alpha\bar{\lambda}_\omega\Delta_{2,\omega}| + |\alpha\Delta_{2,\omega}|^k \Big) \\ &\quad \leq |\lambda_\omega|^k (1+\gamma_\omega)^k \Big(\frac{\alpha^k - 1}{\alpha - 1}(|1 - \alpha\bar{\lambda}_\omega| + \alpha|\bar{\lambda}_\omega|\gamma_\omega) + \alpha^k \Big) \end{aligned}$$

where the last inequality follows from the fact that

$$|\Sigma_\omega| \leq \sum_{j=0}^{k-1} |\Delta_{1,\omega}|^{k-1-j} \alpha^j |\Delta_{2,\omega}|^j \leq (1+\gamma_\omega)^{k-1} \sum_{j=0}^{k-1} \alpha^j = (1+\gamma_\omega)^{k-1} \frac{\alpha^k - 1}{\alpha - 1}$$

and

$$|1 - \alpha\bar{\lambda}_\omega\Delta_{2,\omega}| \leq |1 - \alpha\bar{\lambda}_\omega| + \alpha|\bar{\lambda}_\omega|\gamma_\omega$$

This concludes the proof. ■

After considering the robustness of the second order system the actual behavior of $U_{k,\omega}$ will be considered.

11.4 Actual Behavior of $U_{k,\omega}$

When applying a second order ILC algorithm it is not only important that the value of $|\tilde{U}_{k,\omega}|$ converges rapidly to zero. It is equally important that the limit value, $U_{\infty,\omega}$, actually produces an acceptable error level. From the first order ILC theory it is well known that after introducing a Q-filter the error is not sure to converge to zero, also in the nominal case. For the second order ILC algorithms it is of course important to find a similar condition. One such condition is given by the following theorem.

Theorem 11.8 (Sufficient condition for $|E_{\infty,\omega}| = 0$***)***
A sufficient condition for $E_{\infty,\omega}$ *to be equal to zero for a convergent second order ILC algorithm is that*

$$Q_{1,\omega} + Q_{2,\omega} = 1$$

and

$$|Q_{1,\omega}L_{1,\omega} + Q_{2,\omega}L_{2,\omega}| \neq 0.$$

Proof Follows from the fact that

$$U_{k+1,\omega} = Q_{1,\omega}(U_{k,\omega} + L_{1,\omega}E_{k,\omega}) + Q_{2,\omega}(U_{k-1,\omega} + L_{2,\omega}E_{k-1,\omega})$$

and when $k \to \infty$, $U_{k,\omega}$ tends to $U_{\infty,\omega}$. Use $Q_{1,\omega} + Q_{2,\omega} = 1$ and it follows that

$$|Q_{1,\omega}L_{1,\omega} + Q_{2,\omega}L_{2,\omega}||E_{\infty,\omega}| = 0$$

This is only true if $|E_{\infty,\omega}| = 0$. ■

The result in Theorem 11.8 is of great practical importance and will have an impact on all the design methods for second order ILC algorithms presented in the next chapter. A similar condition formulated in the time domain is found in Bien and Huh (1989).

Appendix

11.A Proof of Lemma 11.3

First it is shown that if $\lambda_{i,\omega}$ is chosen in the set described by (11.34) the lower bound is less than 1.

When $\operatorname{sign}\lambda_{1,\omega} \neq \operatorname{sign}\lambda_{2,\omega}$ the result follows trivially from Lemma 11.1 since the lower bound is always less than or equal to the upper bound.

Now consider the case when $\lambda_{i,\omega} > 0$. Expanding the definition of the lower bound using $\kappa_{i,\omega}$ from (11.12) the result becomes

$$|(1-\lambda_{2,\omega})\lambda_{1,\omega}^{k+1} - (1-\lambda_{1,\omega})\lambda_{2,\omega}^{k+1}| < |\lambda_{1,\omega} - \lambda_{2,\omega}| \tag{11.A.39}$$

where the absolute value of the two terms on the lefthand side of the inequality can be removed since the two terms are positive. Note that when $k = 0$ there will obviously be equality in (11.A.39). Now let

$$\begin{aligned} f(\lambda_{1,\omega},\lambda_{2,\omega},k) &= |(1-\lambda_{2,\omega})\lambda_{1,\omega}^{k+1} - (1-\lambda_{1,\omega})\lambda_{2,\omega}^{k+1}| \\ &= |\lambda_{1,\omega}^{k+1} - \lambda_{2,\omega}^{k+1} - \lambda_{1,\omega}\lambda_{2,\omega}(\lambda_{1,\omega}^{k} - \lambda_{2,\omega}^{k})| \end{aligned} \tag{11.A.40}$$

This can be simplified using that $\lambda_{1,\omega}^{k+1} - \lambda_{2,\omega}^{k+1}$ can be rewritten according to

$$\lambda_{1,\omega}^{k+1} - \lambda_{2,\omega}^{k+1} = (\lambda_{1,\omega} - \lambda_{2,\omega})(\lambda_{1,\omega}^{k} + \lambda_{1,\omega}^{k-1}\lambda_{2,\omega} + \ldots + \lambda_{1,\omega}\lambda_{2,\omega}^{k-1} + \lambda_{2,\omega}^{k}) \tag{11.A.41}$$

Obviously it is possible to introduce the following definition of a new function $g(\cdot)$,

$$|\lambda_{1,\omega} - \lambda_{2,\omega}|g(\lambda_{1,\omega},\lambda_{2,\omega},k) = f(\lambda_{1,\omega},\lambda_{2,\omega},k) \tag{11.A.42}$$

which means that

$$\begin{aligned} g(\lambda_{1,\omega},\lambda_{2,\omega},k) &= \lambda_{1,\omega}^{k} + \lambda_{1,\omega}^{k-1}\lambda_{2,\omega} + \ldots + \lambda_{1,\omega}\lambda_{2,\omega}^{k-1} + \lambda_{2,\omega}^{k} \\ &\quad - \lambda_{1,\omega}\lambda_{2,\omega}(\lambda_{1,\omega}^{k-1} + \lambda_{1,\omega}^{k-2}\lambda_{2,\omega} + \ldots + \lambda_{1,\omega}\lambda_{2,\omega}^{k-2} + \lambda_{2,\omega}^{k-1}) \end{aligned} \tag{11.A.43}$$

The inequality that has to be fulfilled can now be expressed in terms of the new function $g(\cdot)$,

$$g(\lambda_{1,\omega}, \lambda_{2,\omega}, k) < 1, \quad k \in \mathbb{Z}^+ \tag{11.A.44}$$

Using induction it is possible to prove (11.A.44) when $\lambda_{i,\omega} \in]0,1[$.

1. Check that (11.A.44) is fulfilled for $k = 1$.

   $$g(\lambda_{1,\omega}, \lambda_{2,\omega}, 1) = \lambda_{1,\omega} + \lambda_{2,\omega} - \lambda_{1,\omega}\lambda_{2,\omega}$$

   The gradient for $g(\cdot)$ becomes

   $$\frac{\partial}{\partial \lambda}\, g(\lambda_{1,\omega}, \lambda_{2,\omega}, 1) = \begin{bmatrix} 1 - \lambda_{2,\omega} \\ 1 - \lambda_{1,\omega} \end{bmatrix}$$

   and clearly it has a stationary point in $\lambda_{1,\omega} = \lambda_{2,\omega} = 1$. For $\lambda_{i,\omega} \in]0,1[$ the function $g(\lambda_{1,\omega}, \lambda_{2,\omega}, 1)$ is increasing and when $\lambda_{1,\omega}$ or $\lambda_{2,\omega}$ equal 1 $g(\lambda_{1,\omega}, \lambda_{2,\omega}, 1) = 1$. For $\lambda_{1,\omega} = \lambda_{2,\omega} = 0$ the function has the value 0. Since the function is strictly growing and equals 1 when $\lambda_{1,\omega} = \lambda_{2,\omega} = 1$ it will be strictly less than 1 when $0 < \lambda_{i,\omega} < 1$.

2. Assume that for k

   $$g(\lambda_{1,\omega}, \lambda_{2,\omega}, k) < 1$$

   In the induction step it is necessary to show that this is also valid for $k + 1$. Consider

   $$\begin{aligned} g(\lambda_{1,\omega}, \lambda_{2,\omega}, k+1) - g(\lambda_{1,\omega}, \lambda_{2,\omega}, k) &= (\lambda_{1,\omega} - 1)(\lambda_{1,\omega}^k + \lambda_{1,\omega}^{k-1}\lambda_{2,\omega}(1 - \lambda_{1,\omega}) \\ &+ \lambda_{1,\omega}^{k-2}\lambda_{2,\omega}^2(1 - \lambda_{1,\omega}) + \ldots + \lambda_{1,\omega}\lambda_{2,\omega}^{k-1}(1 - \lambda_{1,\omega}) + \lambda_{2,\omega}^k(1 - \lambda_{1,\omega}) - \lambda_{2,\omega}^{k+1}) \end{aligned}$$

   where (11.A.43) has been used twice and it is assumed that $\lambda_{1,\omega} > \lambda_{2,\omega}$. Now since $\lambda_{1,\omega} - 1 < 0$ it is sufficient to show that the second factor is greater than 0. Obviously $\lambda_{2,\omega}^{k+1} < \lambda_{1,\omega}^k$ and all the rest of the terms are positive so the sum must be positive. If instead $\lambda_{2,\omega} > \lambda_{1,\omega}$ just swap $\lambda_{1,\omega}$ and $\lambda_{2,\omega}$ above to get the same result.

Now assume that $-1 < \lambda_{i,\omega} < 0$ and consider again the inequality in (11.A.39). For simplicity let $\tilde{\lambda}_{i,\omega} = -\lambda_{i,\omega}$ and assume $\tilde{\lambda}_{1,\omega} > \tilde{\lambda}_{2,\omega}$. Obviously (11.A.39) becomes

$$|(1 + \tilde{\lambda}_{2,\omega})\tilde{\lambda}_{1,\omega}^{k+1} - (1 + \tilde{\lambda}_{1,\omega})\tilde{\lambda}_{2,\omega}^{k+1}| < |\tilde{\lambda}_{1,\omega} - \tilde{\lambda}_{2,\omega}| \tag{11.A.45}$$

Using a similar transformation as in (11.A.40) and the result in (11.A.41) it is possible to find the same kind of relation as (11.A.42) but with $g(\cdot)$ defined according to

$$\begin{aligned} g(\tilde{\lambda}_{1,\omega},\tilde{\lambda}_{2,\omega}, k) &= \tilde{\lambda}_{1,\omega}^k + \tilde{\lambda}_{1,\omega}^{k-1}\tilde{\lambda}_{2,\omega} + \ldots + \tilde{\lambda}_{1,\omega}\tilde{\lambda}_{2,\omega}^{k-1} + \tilde{\lambda}_{2,\omega}^k \\ &+ \tilde{\lambda}_{1,\omega}\tilde{\lambda}_{2,\omega}(\tilde{\lambda}_{1,\omega}^{k-1} + \tilde{\lambda}_{1,\omega}^{k-2}\tilde{\lambda}_{2,\omega} + \ldots + \tilde{\lambda}_{1,\omega}\tilde{\lambda}_{2,\omega}^{k-2} + \tilde{\lambda}_{2,\omega}^{k-1}) \end{aligned} \tag{11.A.46}$$

To show that

$$g(\tilde{\lambda}_{1,\omega}, \tilde{\lambda}_{2,\omega}, k) < 1, \quad k \in \mathbb{Z}^+ \tag{11.A.47}$$

is fulfilled the same kind of induction approach is used as in the first part of the proof.

1. Check that (11.A.44) is fulfilled for $k = 1$.

   $$g(\tilde{\lambda}_{1,\omega}, \tilde{\lambda}_{2,\omega}, 1) = \tilde{\lambda}_{1,\omega} + \tilde{\lambda}_{2,\omega} + \tilde{\lambda}_{1,\omega}\tilde{\lambda}_{2,\omega}$$

   The gradient for $g(\cdot)$ becomes

   $$\frac{\partial}{\partial \lambda}\, g(\tilde{\lambda}_{1,\omega}, \tilde{\lambda}_{2,\omega}, 1) = \begin{bmatrix} 1 + \tilde{\lambda}_{2,\omega} \\ 1 + \tilde{\lambda}_{1,\omega} \end{bmatrix}$$

   with the extreme point, $\tilde{\lambda}_{1,\omega} = \tilde{\lambda}_{2,\omega} = -1$, outside the set under consideration. Inside the set $0 < \tilde{\lambda}_{i,\omega} < 1$ the function $g(\cdot)$ is obviously an increasing function. Obviously $g(0,0,1) = 0$, $g(1,0,1) = g(0,1,1) = 1$, and $g(1,1,1) = 2$ which immediately shows that the inequality in (11.A.47) cannot be fulfilled for all possible choices of $\tilde{\lambda}_{i,\omega} \in]0,1[$. To find the limit for the set in which to choose $\tilde{\lambda}_{i,\omega}$ the following equation is solved,

   $$\tilde{\lambda}_{1,\omega} + \tilde{\lambda}_{2,\omega} + \tilde{\lambda}_{1,\omega}\tilde{\lambda}_{2,\omega} = 1 \Leftrightarrow \tilde{\lambda}_{2,\omega} = \frac{1 - \tilde{\lambda}_{1,\omega}}{\tilde{\lambda}_{1,\omega} + 1} \tag{11.A.48}$$

   which means that (11.A.47) is fulfilled for

   $$\tilde{\lambda}_{1,\omega} \in]0,1[, \quad \tilde{\lambda}_{2,\omega} < \frac{1 - \tilde{\lambda}_{1,\omega}}{\tilde{\lambda}_{1,\omega} + 1} \tag{11.A.49}$$

2. The second step is very similar to the second step in the previous case. First assume that for k it is true that

   $$g(\tilde{\lambda}_{1,\omega}, \lambda_{2,\omega}, k) < 1$$

   In the induction step it is necessary to show that this is true also for $k + 1$. Consider

   $$\begin{aligned} g(\tilde{\lambda}_{1,\omega}, \tilde{\lambda}_{2,\omega}, k+1) - g(\tilde{\lambda}_{1,\omega}, \tilde{\lambda}_{2,\omega}, k) &= (\tilde{\lambda}_{1,\omega} - 1)(\tilde{\lambda}_{1,\omega}^{k} + \tilde{\lambda}_{1,\omega}^{k-1}\tilde{\lambda}_{2,\omega}(1 + \tilde{\lambda}_{1,\omega}) \\ &+ \tilde{\lambda}_{1,\omega}^{k-2}\tilde{\lambda}_{2,\omega}^{2}(1 + \tilde{\lambda}_{1,\omega}) + \ldots + \tilde{\lambda}_{1,\omega}\tilde{\lambda}_{2,\omega}^{k-1}(1 + \tilde{\lambda}_{1,\omega}) \\ &+ \tilde{\lambda}_{2,\omega}^{k}(1 + \tilde{\lambda}_{1,\omega}) - \frac{\tilde{\lambda}_{2,\omega}^{k+1}}{\frac{1-\tilde{\lambda}_{1,\omega}}{\tilde{\lambda}_{1,\omega}+1}}) \end{aligned}$$

where (11.A.43) has been used twice. Now since $\tilde{\lambda}_{2,\omega}$ is chosen according to (11.A.49) it is true that $\frac{\tilde{\lambda}_{2,\omega}^{k+1}}{\frac{1-\tilde{\lambda}_{1,\omega}}{\tilde{\lambda}_{1,\omega}+1}} < \tilde{\lambda}_{2,\omega}^{k}$ and since $\tilde{\lambda}_{1,\omega}^{k} > \tilde{\lambda}_{2,\omega}^{k}$ it is clear that the result above becomes negative which concludes the "if" part of the proof. The "only if" part follows from the fact that the inequality is invalidated for

$$\tilde{\lambda}_{2,\omega} \geq \frac{1-\tilde{\lambda}_{1,\omega}}{\tilde{\lambda}_{1,\omega}+1}$$

since

$$|(1-\lambda_{2,\omega})\lambda_{1,\omega}^{k+1} - (1-\lambda_{1,\omega})\lambda_{2,\omega}^{k+1}| \geq |\lambda_{1,\omega} - \lambda_{2,\omega}|$$

when $k = 1$.

11.B Proof of Theorem 11.5

The proof can be established in a similar way as was done in the proof of Lemma 11.3. The idea is to use induction.

1. Show that $|\tilde{U}_{k,\omega}| < |\tilde{U}_{0,\omega}|$ when $k = 1$.

$$\frac{|\tilde{U}_{1,\omega}|}{|\tilde{U}_{0,\omega}|} = |\lambda_\omega|\,|(1-\lambda_\omega)+1| < 1 \Rightarrow |2-\lambda_\omega| < \frac{1}{|\lambda_\omega|} \tag{11.B.50}$$

   which is the set given in the theorem.

2. Assume that for k it is true that

$$|\lambda_\omega|^k |k(1-\lambda_\omega)+1| < 1$$

   It is now necessary to show that this is also true for $k+1$. Let $f(\cdot)$ be defined as

$$f(\lambda_\omega, k) = |\lambda_\omega|^k |k(1-\lambda_\omega)+1|$$

   and let

$$g(\lambda_\omega, k) = \ln f(\lambda_\omega, k) = k \ln|\lambda_\omega| + \ln|k(1-\lambda_\omega)+1|$$

   To show that $f(\lambda_\omega, k+1) < 1$ it is enough to show that

$$g(\lambda_\omega, k+1) - g(\lambda_\omega, k) < 0$$

and the lefthand side is equal to

$$\begin{aligned}\ln|\lambda_\omega| + \ln\left|\frac{(k+1)(1-\lambda_\omega)+1}{k(1-\lambda_\omega)+1}\right| &= \ln|\lambda_\omega| + \ln\left|1+\frac{1-\lambda_\omega}{k(1-\lambda_\omega)+1}\right| \\ &< \ln|\lambda_\omega| + \ln|2-\lambda_\omega| \\ &< \ln|\lambda_\omega| - \ln|\lambda_\omega| = 0\end{aligned}$$

where in the first inequality $k = 0$ gives the maximum and in the second inequality the result from Step 1 is applied.

This concludes the "if" part and the "only if" follows from the fact that all other choices of λ_ω will make $|\tilde{U}_{1,\omega}| \geq |\tilde{U}_{0,\omega}|$.

12

Design Methods with an Example

12.1 Design Issues

The discussion in the previous chapter has been kept very abstract without any assumptions on the matrix F_ω other than it is of the structure discussed in Section 10.2. Before going into the actual design examples and the experiments it is worth doing some general comments on the choices of eigenvalues that are actually reasonable.

First recall the actual definition of the matrix F from (10.6), i.e.,

$$F_\omega = \begin{bmatrix} Q_{1,\omega}(1 - L_{1,\omega}T_{u,\omega}) & Q_{2,\omega}(1 - L_{2,\omega}T_{u,\omega}) \\ 1 & 0 \end{bmatrix}$$

The eigenvalues can be calculated using (10.8) resulting in

$$\lambda_{(1,2),\omega} = \frac{f_{1,\omega}}{2} \pm \sqrt{\frac{f_{1,\omega}^2}{4} + f_{2,\omega}} \tag{12.1}$$

where

$$f_{1,\omega} = Q_{1,\omega}(1 - L_{1,\omega}T_{u,\omega}),\ f_{2,\omega} = Q_{2,\omega}(1 - L_{2,\omega}T_{u,\omega})$$

From now on it is assumed that the system where ILC is applied can be described as in (10.1), i.e.,

$$y_k(t) = T_r(q)r(t) + T_u(q)u_k(t)$$

The proposed second order ILC updating formula is given by (10.3), i.e.,

$$u_{k+1}(t) = Q_1(q)\big(u_k(t) + L_1(q)e_k(t)\big) + Q_2(q)\big(u_{k-1}(t) + L_2(q)e_{k-1}(t)\big)$$

A design methodology for this ILC algorithm must be able to, more or less systematically, find the filters Q_1, Q_2, L_1, and L_2. How to choose the Q and the L filters in a first order ILC algorithm is quite well known and there exist some algorithms that from a model can calculate these filters, see for example Bien and Xu (1998), Moore (1993), and Chapter 8.

In fact, the choice of f_1 is similar to the first order ILC design. Moreover, from (12.1) it follows that $\lambda_{(1,2),\omega}$ is symmetric with respect to $\frac{f_1}{2}$. The minimum absolute value of the two eigenvalues is achieved when F_ω has one distinct eigenvalue, and the value becomes $\lambda_\omega = \frac{f_1}{2}$. This might not be desirable from an application point of view as will be seen in the next sections.

12.1.1 Design algorithm proposals

A natural choice from which to start a design effort for a second order ILC algorithm is (12.1). From the discussion in Section 10.2.2 we know that it is possible to find a description of a set in which to choose $f_{2,\omega}$ in order to be sure that the resulting iterative system is stable. This is not a very constructive approach since stability (although the most important property) is not enough for a design algorithm.

In the next section the proposed design algorithms will be compared with a first order ILC algorithm. The idea is to understand the advantages and disadvantages of having a second order compared to a first order ILC algorithm.

A heuristic design algorithm

The first design algorithm proposal takes advantage of the already existing algorithms for finding good choices of the filters in a first order ILC algorithm. From the description of the second order ILC updating formula given in (10.3) it is obvious that the updating can be divided into two parts. The first one is the normal first order ILC updating formula, updating the next control input using the current control input and the current error. The second part does the same thing but is delayed one iteration, i.e., it updates the next control input, not from the current, but from the previous control input and the previous error.

The idea of the first design algorithm proposal is to simply take a first order ILC design and then divide it into a part that works on the data from the current

iteration and one part that works on the data from the previous iteration. This is formalized as follows.

Algorithm 12.1 (Heuristic second order ILC design)

1. First do a design of a first order ILC according to some design methodology for first order ILC algorithms. This results in the filters Q and L.
2. Let the filters in the second order ILC be chosen according to

$$\begin{aligned} Q_{1,\omega} &= \alpha Q_\omega,\ L_{1,\omega} = L_\omega \\ Q_{2,\omega} &= (1-\alpha) Q_\omega,\ L_{2,\omega} = L_\omega \end{aligned} \tag{12.2}$$

   where $\alpha \in \mathbb{R}$ is a design variable.
3. Try, on the true system or in a simulation, different values of the design parameter $\alpha \in [0, 1]$. Choose the value of α that gives a satisfactory result.

If α is chosen between 0 and 1 it is obvious that it decides how much of the control input should be updated from the current iteration and how much should be updated from the previous one. If $\alpha = 1$ the algorithm will give a pure first order ILC algorithm and with $\alpha = 0$ it will give raise to a kind of delayed (in iteration) first order ILC method. Note that the property of Theorem 11.8 will hold as long as the original design has $Q_\omega = 1$.

An eigenvalue based algorithm

By considering (12.1) it is clear that given a value of $f_{1,\omega}$ the best eigenvalue, with respect to the amplitude, is $\frac{f_{1,\omega}}{2}$. This is the case when $f_{2,\omega}$ is chosen such that

$$\frac{f_{1,\omega}^2}{4} + f_{2,\omega} = 0$$

which is equivalent to

$$Q_{2,\omega} = -\frac{Q_{1,\omega}^2}{4}, \quad L_{2,\omega} = L_{1,\omega}(2 - L_{1,\omega} T_{u,\omega})$$

This choice will, however, only fulfill the condition in Theorem 11.8 if $Q_{1,\omega} = 2$. The approach here will instead be to choose an approximate solution based on a first order ILC design.

Algorithm 12.2 (Eigenvalue based second order ILC design)

1. Design a first order ILC, i.e., choose the filters Q and L according to a design methodology for first order ILC algorithms.
2. Choose Q_1 and L_1 according to,

$$Q_{1,\omega} = \frac{5}{4}Q_\omega, \quad L_{1,\omega} = L_\omega$$

3. Choose Q_2 and L_2 such that

$$Q_{2,\omega} = -\frac{Q_\omega^2}{4}, \quad L_{2,\omega} = L_{1,\omega}(2 - L_{1,\omega}T_{u,\omega})$$

Note that the approach suggested in Algorithm 12.2 is in fact model based since T_u is used in the construction of the L_2 filter. One important difference compared to many suggested first order ILC updating schemes is however that it uses is the model and not its inverse. If the first order ILC is chosen as the optimal solution $L_1 = T_u^{-1}$, without considering robustness, then from the choice of L_2 in the algorithm it is clear that also $L_2 = T_u^{-1}$. The choice of Q_1 and Q_2 stems from the condition in Theorem 11.8. In the frequency band where the first order ILC has zero error convergence, i.e., $Q_\omega = 1$, the second order design using Algorithm 12.2 will also converge to zero since

$$Q_{1,\omega} + Q_{2,\omega} = \frac{5}{4}Q_\omega + \frac{Q_\omega^2}{4} \approx 1$$

Now an analysis of the proposed design algorithms are analyzed.

12.1.2 Analysis of the proposed design algorithms

A natural way of evaluating the second order ILC algorithm is to compare it with the corresponding first order design. Actually both design algorithms presented here are based upon a first order ILC design. Obviously it is natural to compare the result from this initial design with the result of the second order ILC design, both from a performance as well as a robustness point of view. First consider the first order ILC design.

Performance and robustness for a first order ILC design

Consider the first order ILC with the filters Q and L. With the updating equation according to (3.13) and $\tilde{U}_{k,\omega}^1$ defined as in (11.7) it follows that

$$\tilde{U}_{k+1,\omega}^1 = Q_\omega(1 - L_\omega T_{u,\omega})\tilde{U}_{k,\omega}^1$$

where also the frequency domain version of (4.23) is used. The superscript 1 in $\tilde{U}^1_{k,\omega}$ indicates that it is $\tilde{U}_{k,\omega}$ for the first order ILC algorithm. The control signal for the second order algorithm is referred to as $\tilde{U}^2_{k,\omega}$.

Now let

$$f_\omega = Q_\omega(1 - L_\omega T_{u,\omega})$$

It is clearly true that

$$\frac{|\tilde{U}^1_{k+1,\omega}|}{|\tilde{U}^1_{0,\omega}|} = |f_\omega|^{k+1} \tag{12.3}$$

which will be used to compare the performance for the first and the second order ILC designs.

It is necessary to find a bound such that the robustness for the first order ILC design can be compared with what is achieved with the second order ILC design. Assume that

$$f_\omega = \bar{f}_\omega(1 + \delta^r_\omega), \quad |\delta^r_\omega| < \gamma_\omega$$

where $\bar{f}_\omega$ is the nominal value. A sufficient condition for stability is that

$$\gamma_\omega < \frac{1}{|\bar{f}_\omega|} - 1$$

If it is an absolute uncertainty,

$$f_\omega = \bar{f}_\omega + \delta^a_\omega, \quad |\delta^a_\omega| < \gamma_\omega$$

the corresponding sufficient criterion for robust stability becomes

$$\gamma_\omega < 1 - |f_\omega|$$

Now the result of the proposed design algorithms can be compared with the corresponding first order ILC scheme.

Heuristic based design

The eigenvalues for the second order ILC scheme designed using Algorithm 12.1 are given by

$$\lambda_{(1,2),\omega} = \alpha\frac{f_\omega}{2} \pm \sqrt{\frac{\alpha^2 f^2_\omega}{4} + (1-\alpha)f_\omega} = \frac{f_\omega}{2}\Big(\alpha \pm \sqrt{\alpha^2 + 4\frac{1-\alpha}{f_\omega}}\Big)$$

First note that if α is chosen as $\alpha = 1$ the eigenvalues will be

$$\lambda_{1,\omega} = f_\omega, \quad \lambda_{2,\omega} = 0$$

which is equivalent to a normal first order ILC algorithm. If, instead, $\alpha = 0$ the eigenvalues become

$$\lambda_{(1,2),\omega} = \pm\sqrt{f_\omega}$$

Clearly, $|\lambda_{(1,2),\omega}| = \sqrt{|f_\omega|} > |f_\omega|$ since $|f_\omega| < 1$.

To compare the robustness of the heuristic based design with the corresponding first order ILC design, one possible approach is to compare the stability margin. The stability margin is here defined as

$$1 - |f_\omega|$$

for the first order ILC design and

$$1 - \max(|\lambda_{1,\omega}, \lambda_{2,\omega}|)$$

for the second order ILC design. In both cases this measures the distance to the border of the stability region.

In Figure 12.1 the difference between the stability margin for the first and the second order ILC, i.e.,

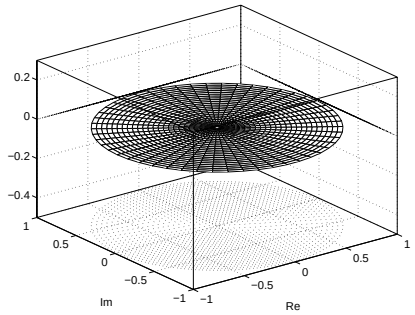
$$1 - |f_\omega| - (1 - \max(|\lambda_{1,\omega}|, |\lambda_{2,\omega}|))$$

is shown as a function of f_ω for some values of α between 0 and 1. This means that a positive value implies that the stability margin for the first order ILC algorithm is higher than the second order and vice versa. The difference is obviously zero for $\alpha = 1$ since the second order ILC in that case coincides with the first order design. Note that when the second order part is introduced the robustness for small values of $|f_\omega|$ is always worse compared to the first order ILC. Especially this is true for $\alpha = 0$ when the stability margin is worse for all f_ω. For intermediate values of α there are some f_ω that give a better robustness using the second order ILC compared to the first order. One conclusion that could be drawn is that it might be an idea to introduce an α that is not constant but instead f_ω dependent. In this way it might be possible to use more of the advantages of the second order method, without the drawback of having worse robustness properties for some f_ω.

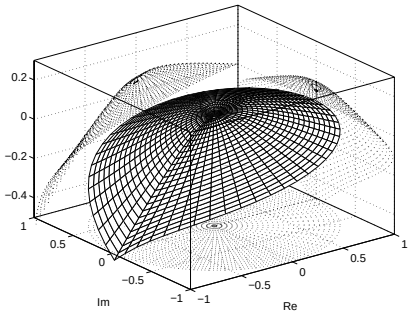
The performance of the resulting second order ILC algorithm is of great importance and the actual convergence speed will depend on the absolute value of the eigenvalues. Actually it is possible to see this in Figure 12.1 as well. Consider

$$1 - |f_\omega| - (1 - \max(|\lambda_{1,\omega}|, |\lambda_{2,\omega}|)) = \max(|\lambda_{1,\omega}, \lambda_{2,\omega}|) - |f_\omega|$$

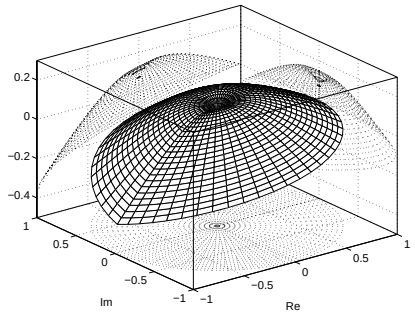
which means that also the difference between the maximum eigenvalue for the second order ILC and the first order ILC can be seen in the diagrams. Also with this interpretation positive values are in favor of the first order ILC and negative values are in favor of the second order algorithm.



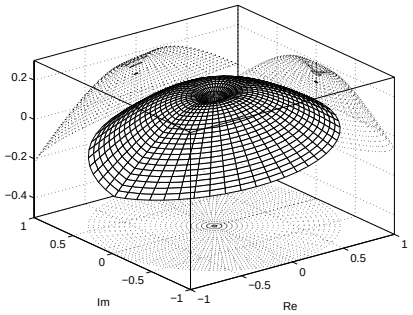
(a) $\alpha = 1$



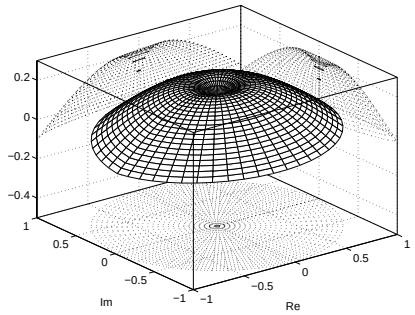
(b) $\alpha = 0.8$



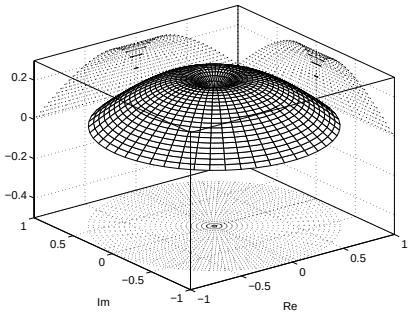
(c) $\alpha = 0.6$



(d) $\alpha = 0.4$



(e) $\alpha = 0.2$



(f) $\alpha = 0$

Figure 12.1 *Robustness bounds for the first order ILC compared to the heuristically designed second order ILC.*

Eigenvalue based design

When using Algorithm 12.2 the eigenvalues are given by

$$\lambda_{(1,2),\omega} = \frac{Q_\omega(1 - L_\omega T_{u,\omega})}{8}\big(5 \pm 3\big)$$

Obviously this means that if $f_\omega = Q_\omega(1 - L_\omega T_{u,\omega})$ is the result of the original design, the nominal eigenvalues are $\bar{\lambda}_{1,\omega} = f_\omega$ and $\bar{\lambda}_{2,\omega} = \frac{1}{4} f_\omega$. From a robustness perspective this gives exactly the same result as the first order design since the absolute amplitude margin for the second order ILC is exactly the same as the one of the first order ILC.

Important is also to consider the performance that can be achieved compared to the first order ILC design. For the second order design it follows from Theorem 11.1 that

$$\frac{|\tilde{U}^2_{k,\omega}|}{|\tilde{U}^2_{0,\omega}|} = |f_\omega|^k \frac{1}{3 \cdot 4^k} |4^{k+1} - 1 - f_\omega(4^k - 1)|$$

To compare the first and the second order ILC it is possible to study, for example,

$$\frac{|\tilde{U}^2_{k,\omega}|}{|\tilde{U}^2_{0,\omega}|} - \frac{|\tilde{U}^1_{k,\omega}|}{|\tilde{U}^1_{0,\omega}|} = |f_\omega|^k \Big(\frac{1}{3 \cdot 4^k} |4^{k+1} - 1 - f_\omega(4^k - 1)| - 1 \Big)$$

Using the following bound

$$1 < \frac{1}{3 \cdot 4^k} |4^{k+1} - 1 - f_\omega(4^k - 1)| < \frac{5}{3}$$

which is found using the triangular inequality and the fact that $|f_\omega| < 1$. It is clear that the second order ILC algorithm designed using the eigenvalue based design will never work better compared to the first order ILC on which it is based.

12.2 A Design Example with Experiments on an Industrial Robot

In the previous sections a theory has been developed for analyzing and understanding the behavior of second order ILC systems. Now this theory will be applied on a design example for a real industrial system. The system is the ABB industrial robot, IRB1400, described in Part II.

In this example ILC is applied to three joints. The robot has a total of 6-DOF but for the three wrist joints ILC is not applied. Each joint is modeled as a transfer operator description from the ILC control input to the measured motor position on the robot, i.e., what is called T_u in (10.1). It should be stressed that this T_u is in

fact a model of a closed loop system. The conventional feedback controller in the S4C control system is working in parallel with the ILC scheme. Since the controller is doing a very good job the closed loop from reference angular position to measured angular position can be described using a low order linear discrete time model. The models are calculated using *System Identification Toolbox* (Ljung, 1995) and are given by,

$$\begin{aligned} \hat{T}_{u_1}(q) = \hat{T}_{u_2}(q) &= \frac{0.1q^{-1}}{1 - 0.9q^{-1}} \\ \hat{T}_{u_3}(q) &= \frac{0.13q^{-1}}{1 - 0.87q^{-1}} \end{aligned} \tag{12.4}$$

and the sampling time is $t_s = 0.004$. The modeling process follows the description in Section 7.3.

12.2.1 Description of the experiment

The experiment is performed on the ABB IRB1400 robot described in Chapter 7. In Figure 12.2 the program used in the experiment is shown together with the resulting path on the arm-side. The instruction `movec p2,p3,v60` refers to an instruction that produces an arc on the arm-side of the robot. The arc starts from the current position, not explicitly stated, and goes through the points `p2` and `p3`. The speed along the path is in this case programmed to be 60 mm/s. The `movec` instruction is a simplified version of the corresponding RAPID™ instruction `MoveC` (ABB, 1997). The actual position of `p1` in the base coordinate system is x = 1300 mm, y = 100 mm, and z = 707 mm. The configuration of the robot is also shown in Figure 8.10.

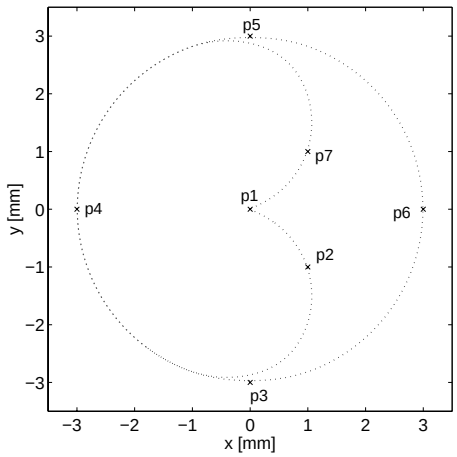
12.2.2 The first order ILC algorithm

The design of the first order ILC method is based on Algorithm 8.1. The procedure is briefly described here and the results from the experiment with the first order ILC are presented in Section 12.2.4.

Following Algorithm 8.1 the filter L is chosen as

$$L(q) = 0.9q^2 \tag{12.5}$$

and the corresponding Q filter is chosen as $Q(q) = Q_{1/2}(q)Q_{1/2}(\frac{1}{q})$, i.e., a zero-phase filter. The filter $Q_{1/2}(q)$ is chosen as a second order Butterworth filter with cut-off frequency 0.2 of the Nyquist frequency. The resulting nominal $Q_\omega(1 - L_\omega T_{u_i,\omega})$ are depicted in Figure 12.3 and it is obvious that the stability criterion formulated in the frequency domain, cf. (10.4), is fulfilled for all the three joints.




```
%% starting at p1
%% starting at p1
movec p2,p3,v60;
movec p4,p5,v60;
movec p6,p3,v60;
movec p4,p5,v60;
movec p7,p1,v60;
```

Figure 12.2 *The resulting trajectory from the program used in the experiment. The resulting path is translated such that the origin coincide with p1.*

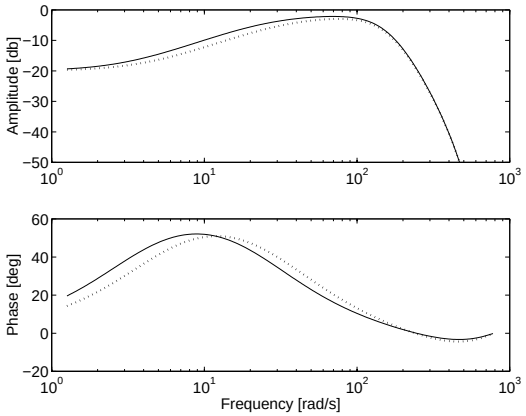


Figure 12.3 *The result from the design of the first order ILC. $Q_\omega(1-L_\omega T_{u_i,\omega})$, for $i = 1, 2$ (solid line) and $i = 3$ (dotted).*

12.2.3 The second order ILC algorithm

The design of the second order ILC algorithm is made using the design approaches suggested in Algorithm 12.1 and Algorithm 12.2. First the design using the heuristic algorithm is discussed.

Heuristic design

With the first order ILC design from Section 12.2.2 it is straightforward to apply Algorithm 12.1. The result is:

1. From the first order ILC design,
$$Q(q) = Q_{1/2}(q)Q_{1/2}(\frac{1}{q})$$
where $Q_{1/2}$ is a second order Butterworth filter with cut-off frequency 0.2 of the Nyquist frequency. The L-filter is given by
$$L(q) = q^2$$
2. The filters Q_1, L_1, Q_2, and L_2 are chosen according to (12.2).
3. The value of α is picked as 0.8 from the simulations shown in Figure 12.4.

From Figure 12.4 it is clear that the best performance is achieved having $\alpha = 1$ (which corresponds to the first order ILC design) but with a slightly smaller value of α the performance is nearly the same.

Eigenvalue placement design

Using, again, the first order ILC design from Section 12.2.2 it is easy to apply Algorithm 12.2 ending up with the following design.

1. Step one of the algorithm is exactly the same as in the previous section and uses the Q and L filters from the first order ILC design.
2. Do the choices of Q_1 and L_1 suggested by step 2 in Algorithm 12.2, i.e.,
$$Q_1(q) = \frac{5}{4}Q^2(q)$$
$$L_{1,i}(q) = L(q)$$
3. Choose Q_2 and L_2 according to step 3 in Algorithm 12.2, i.e.,
$$Q_2(q) = -\frac{Q^2(q)}{4}$$
$$L_{2,i}(q) = L_1(q)(2 - L_1(q)T_{u,i}(q))$$

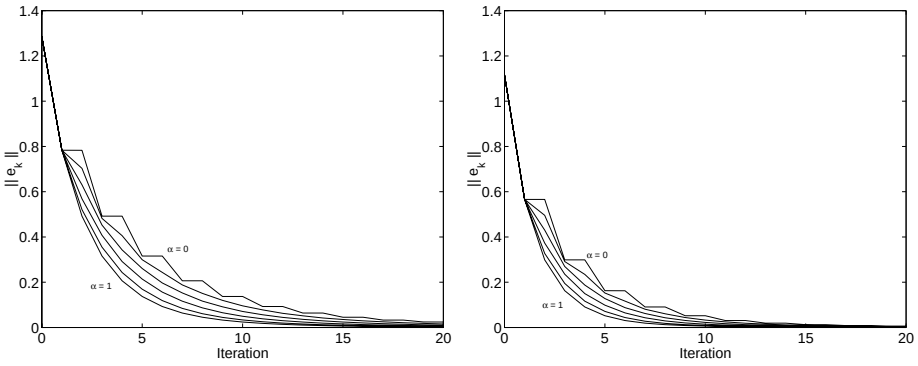


Figure 12.4 *The results from simulations with* $\alpha = 0, 0.2, \ldots, 0.8, 1$ *in the heuristic design approach. The left diagram shows* $\|e_k\|_2$ *for* $T_{u,1}$ *and* $T_{u,2}$ *while the right diagram shows the corresponding value of* $\|e_k\|_2$ *for* $T_{u,3}$.

where Q_2 is the same for all the joints while L_2 is chosen individually for each of the joints. The last fact follows from that L_2 is model dependent.

In Figure 12.5 the nominal value of the eigenvalues are depicted for the two different designs. According to the analysis of the design algorithm in the previous section, this is simply f_ω and $\frac{f_\omega}{4}$.

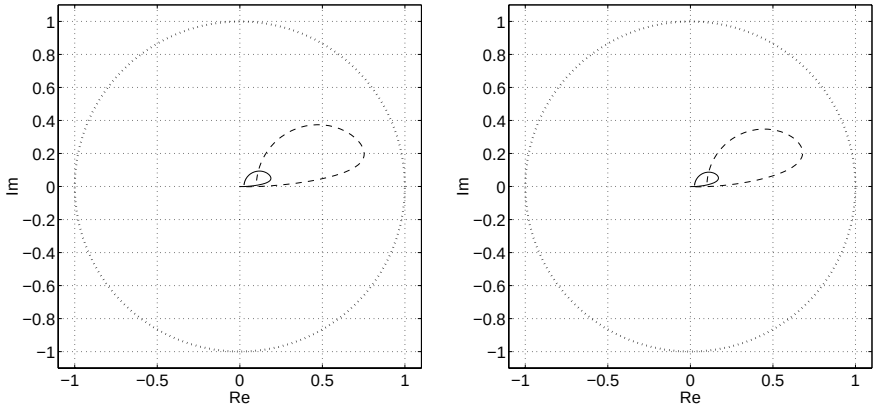
12.2.4 Results from the experiments

In Figure 8.10 the experimental setup is shown and it is possible to see how the robot is positioned while doing the programmed motion.

The result from the experiments can be evaluated from two different points of view. First the result achieved on the motor side is studied. This is the measure used by the ILC algorithms and it is the error in this measure that is being minimized. The other point of view is to consider what is the actual result on the arm-side. This measure is obviously more interesting from an application point of view since it is the path of the tool that should be controlled. Both point of views will be studied here but it is important to stress that the ILC explicitly minimizes only the error on the motor-side of the robot. Compare the discussion in Chapter 9.

Motor-side

The experiments with the three different ILC algorithms have all been running for 10 iterations. Since in the first iteration ILC is not applied, $u_0 \equiv 0$, the



(a) For T_{u_1} and T_{u_2}.

(b) For T_{u_3}.

Figure 12.5 *The result from the design of the second order ILC. Nominal eigenvalues for the second order ILC system, $\lambda_{1,\omega}$ (dashed), $\lambda_{2,\omega}$ (solid) are shown together with the stability region (dotted).*

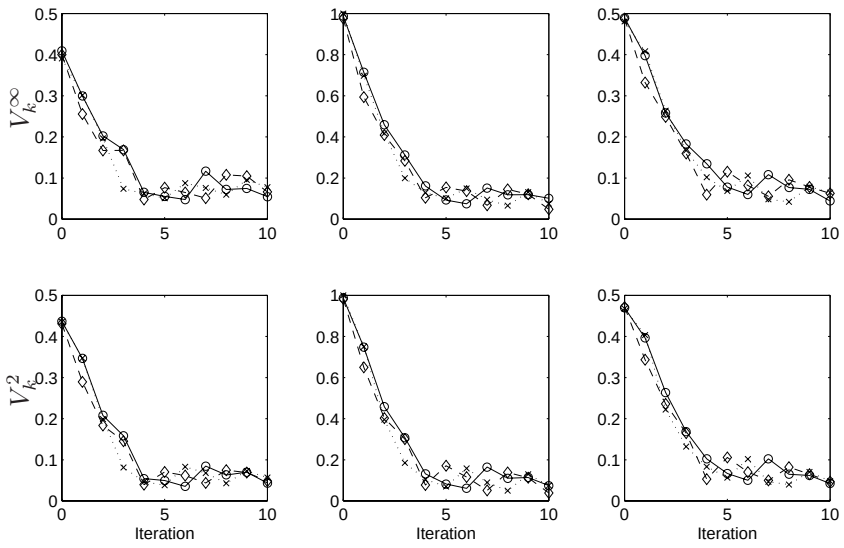
same circle has been done 11 times in each of the three cases. In Figure 12.6 the resulting normalized ∞-norm and 2-norm of the error on the motor-side are shown. The measures in Figure 12.6 are calculated as

$$V_k^p = \frac{\|e_k\|_p}{\max\limits_{\forall \text{ joints,designs}} \|e_k\|_p} \tag{12.6}$$

for $p = 2, \infty$. Using this measure the relative size of the norms for the different joints is kept constant. It is, for example, possible to see that maximum error of motor 3 is 50 % of the maximum error of motor 2.

It is clearly not so easy to decide which design method gives the best result. In the first iteration all the algorithms use the same updating equations and should therefore also reach the same level of error, in theory. As can be seen in Figure 12.6 this is not the case. The ILC algorithm designed according to Algorithm 12.2 gives a lower value of the error in the first iteration compared to the others. In the second iteration the size of the error is again about the same and after the fifth iteration the error stabilizes on a level where V_k^∞ is about 15 % of the initial level.

In Figure 12.7 the ILC control signal, $|U_k(e^{i\omega})|$, is shown for all the different design algorithms. These are also very similar in all the cases. Note that the energy of U_k is almost zero for all frequencies above a certain threshold. This depends on the choice of filter Q_i. This filter serves as a band limiter of the updating of the



(a) Motor 1. *(b)* Motor 2. *(c)* Motor 3.

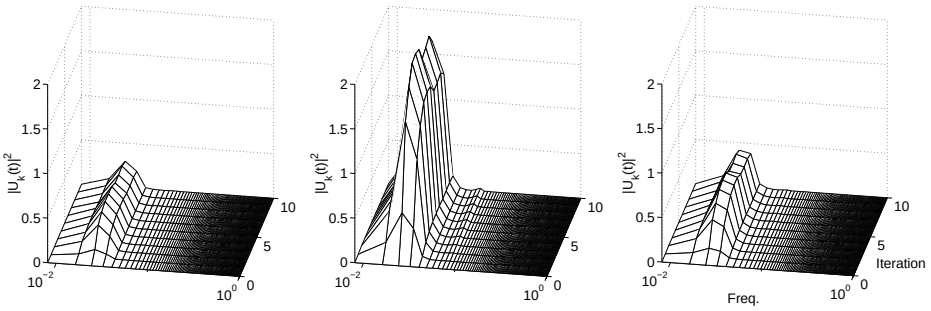
Figure 12.6 *The normalized ∞-norm and 2-norm of the error for the different designs. First order ILC ($\times$), Algorithm 12.1 ($\circ$), and Algorithm 12.2 ($\diamond$).*

control signal in the ILC iterations, cf. (3.13) and (10.3). This also explains why there is a remaining error at some frequencies.

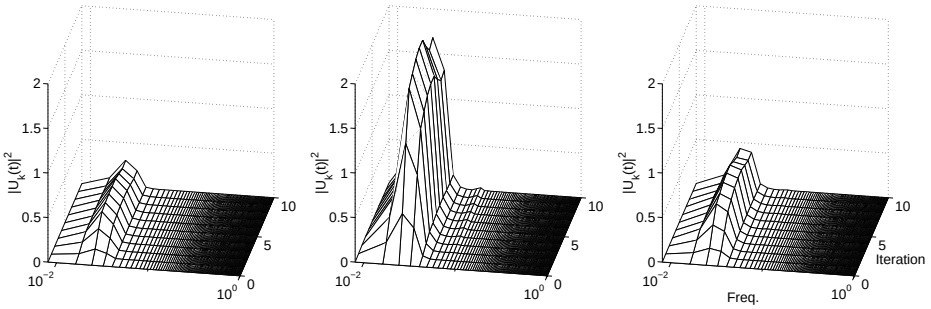
Arm-side

When evaluating the result of the ILC iterations on the arm-side of the robot a pen is used, as shown in Figure 8.10. Obviously the resulting drawing is not so easy to evaluate since the pen in itself produces a quite thick line but in Figure 12.8 the resulting circles are shown for the three different ILC algorithms in the first 5 iterations (iteration 0 to 5, from left to right). A conclusion that can be drawn from the result in Figure 12.8 is that the three ILC algorithms give a similar result also on the arm-side.

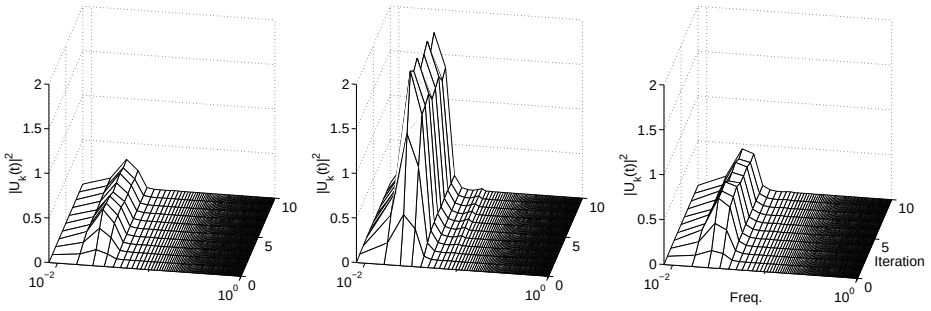
It is also possible to evaluate the result from the ILC experiments on the arm-side by a transformation of the motor signals to the arm side using the forward kinematic model of the IRB1400, see e.g., Norrlöf (1999). The result from this transformation is shown in Figure 12.9, Figure 12.10, and Figure 12.11. In the



(a) First order ILC.



(b) Second order ILC, designed according to Algorithm 12.1.



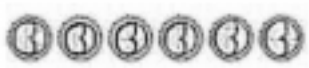
(c) Second order ILC, designed according to Algorithm 12.2.

Figure 12.7 *The ILC signal, $|U_k(e^{i\omega})|$ from the three experiments. The diagrams show from left to right, the resulting $|U_k(e^{i\omega})|$ for axis 1, 2 and 3.*

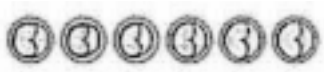
transformation process the outer circle segment is extracted from the path that the robot is actually doing, cf. Figure 8.11. In Figure 12.9 to Figure 12.11 the *root mean square* (RMS) and the *maximum deviation* is shown.

The RMS is here calculated by taking the sum of the squared distance between the actual path and the programmed circle path, divide by the number of samples and finally take the square root of the result. This result is then normalized such that the RMS at iteration 0 of the first order ILC is considered to be one. The maximum deviation is simply the largest distance from the actual path to the programmed path, measured along a normal to the programmed circle path. This is also normalized such that the distance at iteration zero of the first order ILC is considered to be one. The best result, when evaluated using the transformation to the arm-side, is obtained with the first order ILC algorithm. When considering the best fit on all the 10 iterations, the result becomes as shown in Figure 12.12. It is still the first order ILC that is the best although the maximum deviation is a bit bigger compared to the second best, the eigenvalue placement design (Algorithm 12.2).

(a) First order ILC.



(b) Second order ILC, Algorithm 12.1.



(c) Second order ILC, Algorithm 12.2.

Figure 12.8 *Results on arm-side in iterations* 0 *to* 5 *(from left to right). The inner circle is the circle drawn by the robot and the outer circle has as inner diameter of* 6 *mm, i.e., the programmed path.*

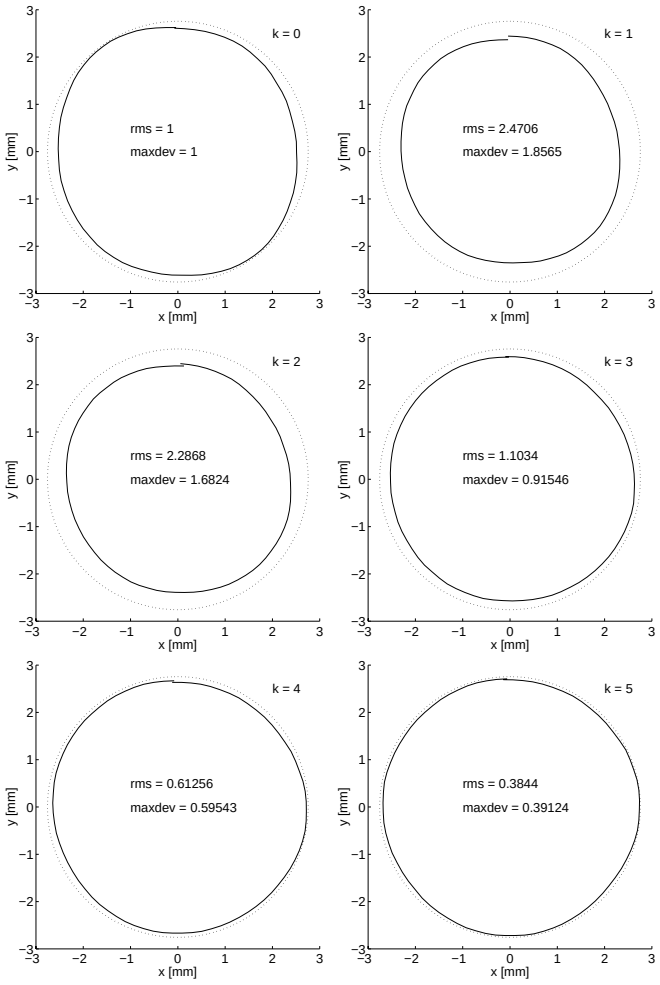


Figure 12.9 *Results of kinematic transformation from motor to arm, first order ILC.*

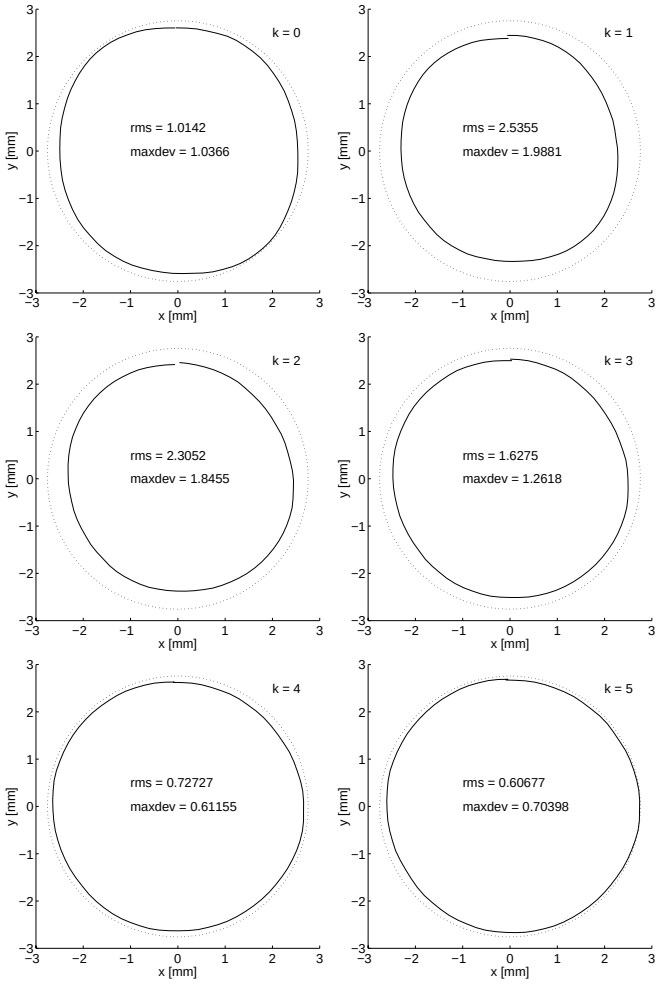


Figure 12.10 *Results of kinematic transformation from motor to arm, second order ILC Algorithm 12.1.*

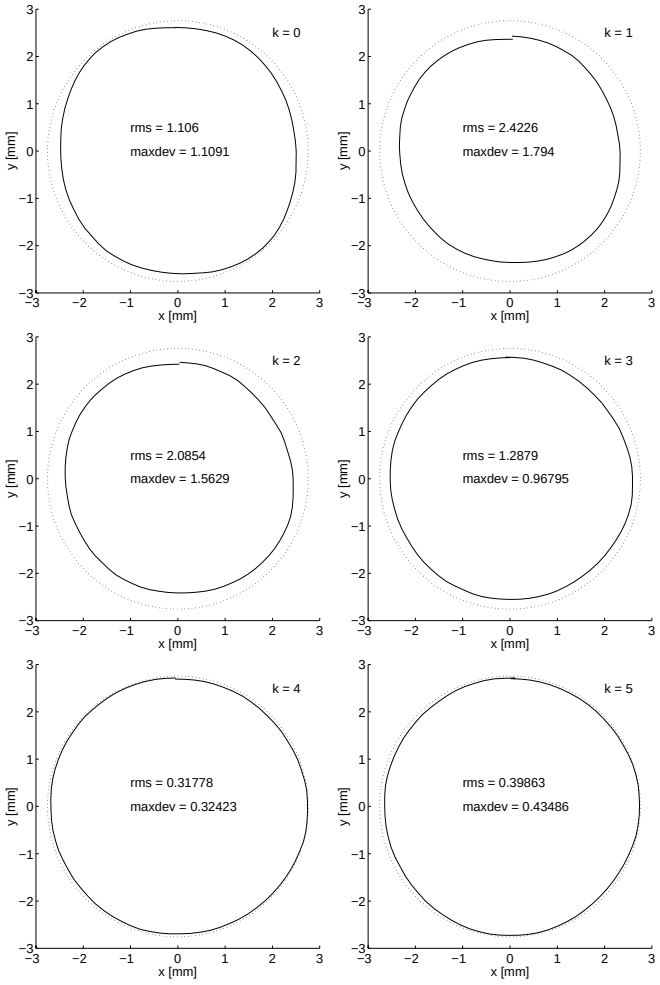
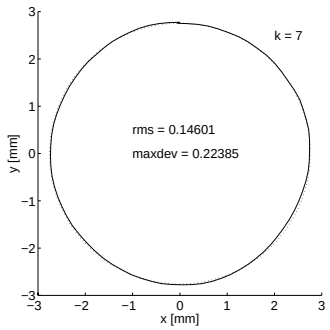
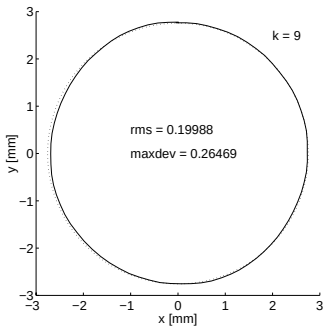


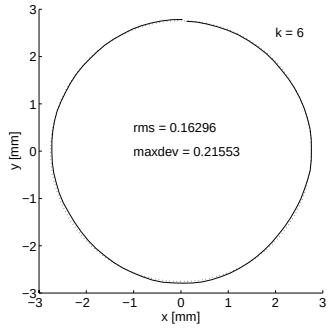
Figure 12.11 *Results of kinematic transformation from motor to arm, second order ILC Algorithm 12.2.*



(a) First order ILC.



(b) Second order ILC, Algorithm 12.1.



(c) Second order ILC, Algorithm 12.2.

Figure 12.12 *Best fit in the analysis of the motor-data transformed to the arm-side.*

13

Conclusions of Part IV

This part of the thesis gives an overall analysis of second order ILC. It also suggests some design algorithms for second order ILC schemes. It is not shown that a second order ILC algorithm does better than a first order ILC but from the analysis and, in fact, also the experiments it is evident that it works as well as the first order design. Some facts are however important to stress when considering going from a first order ILC design to a second order ILC design.

- The second order design should not use more information about the system than the first order design. If more information (models etc.) is available it should be used to produce a first order design that works as well as possible.

- The amount of memory required for the second order ILC scheme, as implemented here, is double the amount used by the corresponding first order ILC scheme. If a cost function measuring performance and memory allocation is used for the first and the second order ILC it is clear that the first order ILC scheme will win.

- One aspect that has not been considered here but that can make the second order ILC scheme very competitive is when an uncertainty is present in the plant and it makes the plant change between the iterations. The second order

algorithm can smooth also the behavior of the system by using the control and the error signal from more than one iteration.

Further work in the area could be to consider the effects of nonlinearities on the resulting control signals and the resulting error. For example, Coloumb friction is a nonlinear effect that is always present in real servo systems and a comparison between a first and second order ILC algorithms for dealing with this could be worth to consider.

Part V

An Adaptive Approach to ILC

14

An Adaptive ILC Algorithm

In this part of the thesis the disturbance rejection description of ILC, formulated in Section 3.2, will be used. An adaptive approach to the disturbance estimation and disturbance rejection problems is taken and the resulting ILC method is tested on an industrial process, the ABB IRB1400 robot described in Part II.

14.1 A State Space Based Approach to ILC

The structure of the system in the disturbance rejection formulation of ILC is shown in Figure 3.3 and mathematically it can be described according to (3.10). Next the mathematical system description will be discussed in more detail before the adaptive ILC method can be presented.

14.1.1 State space description of the system

An ILC system is characterized by the fact that it is only defined over a finite interval of time. If the sampling time is equal to one, this means that $0 \leq t \leq n-1$. This is also the reason why it is possible to write the system description in the matrix form as in (3.10). The system description from (3.10) is here repeated for

convenience,

$$\begin{aligned} \boldsymbol{z}_k &= \boldsymbol{G}^0 \boldsymbol{u}_k + \boldsymbol{d}_k \\ \boldsymbol{y}_k &= \boldsymbol{z}_k + \boldsymbol{n}_k \end{aligned} \tag{14.1}$$

with

$$\boldsymbol{d}_{k+1} = \boldsymbol{d}_k + \boldsymbol{\Delta}_{d_k} \tag{14.2}$$

From Section 3.2 it also follows that $\boldsymbol{z}_k, \boldsymbol{u}_k, \boldsymbol{d}_k, \boldsymbol{y}_k, \boldsymbol{n}_k, \boldsymbol{\Delta}_{d_k} \in \mathbb{R}^n$ and $\boldsymbol{G}^0 \in \mathbb{R}^{n \times n}$.

In this part of the thesis it will be assumed that $\boldsymbol{d}_k$ and $\boldsymbol{n}_k$ are random with covariance matrices for $\boldsymbol{\Delta}_{d_k}$ and $\boldsymbol{n}_k$ given by $\boldsymbol{R}_{\Delta_d,k}$ and $\boldsymbol{R}_{n,k}$ respectively. Often it will be assumed that $d_k(t) = \nu_d(\bar{t})$ and $\Delta_{d_k}(t) = \nu_\Delta(\bar{t})$ with $\bar{t} = k \cdot t$ and $\nu_{(.)}$ a white stationary stochastic process. This is similar to the assumptions in Section 4.3.1.

Another representation of the system in (14.1) that will be useful later is

$$\begin{aligned} z_k(t) &= G^0(q) u_k(t) + d_k(t) \\ y_k(t) &= z_k(t) + n_k(t) \end{aligned} \tag{14.3}$$

with

$$d_{k+1}(t) = d_k(t) + \Delta_{d_k}(t) \tag{14.4}$$

This description is also used in Section 3.5 although $d_k(t)$ is there assumed to be iteration independent, i.e., $d_k(t) = d(t)$. Note that the description in (14.1) is more general than (14.3) since it also covers linear time variant systems. When considering linear time invariant systems, however, the two representations are equivalent.

Using the updating formula for the disturbance, $\boldsymbol{d}_k$, it is possible to write (14.1) in the following form

$$\begin{aligned} \boldsymbol{z}_{k+1} &= \boldsymbol{z}_k - \boldsymbol{G}^0(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k) + \boldsymbol{\Delta}_{d_k} \\ \boldsymbol{y}_k &= \boldsymbol{z}_k + \boldsymbol{n}_k \end{aligned} \tag{14.5}$$

Of course, in a real case the true system description is never available. Instead a model $\boldsymbol{G}$ is used and the relation with the true system can, for example, be

$$\boldsymbol{G}^0 = \boldsymbol{G}(I + \boldsymbol{\Delta}_G) \tag{14.6}$$

where $\boldsymbol{\Delta}_G$ is a relative model error. This means that, from the user's point of view, the system in (14.5) is given by

$$\begin{aligned} \boldsymbol{z}_{k+1} &= \boldsymbol{z}_k - \boldsymbol{G}(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k) - \boldsymbol{G}\boldsymbol{\Delta}_G(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k) + \boldsymbol{\Delta}_{d_k} \\ \boldsymbol{y}_k &= \boldsymbol{z}_k + \boldsymbol{n}_k \end{aligned} \tag{14.7}$$

where the last two terms in the first equation can be considered as disturbances since they are both unknown. It is however known that the first one depends on the difference between two consecutive control signals. If the model uncertainty is small and/or the updating speed of the control signal is slow, this disturbance will have a small effect on the resulting system.

14.1.2 Estimation procedure

A linear estimator for the system described in (14.5) can be written as

$$\hat{\boldsymbol{z}}_{k+1} = \hat{\boldsymbol{z}}_k - \boldsymbol{G}(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k) + \boldsymbol{K}_k(\boldsymbol{y}_k - \hat{\boldsymbol{z}}_k) \tag{14.8}$$

where $\boldsymbol{K}_k$ is the gain of the estimator. This gain can be found by using, for example, a Kalman filter which will be discussed next. A general reference on estimation and Kalman filters is Anderson and Moore (1979) and the reader is referred there for a more thorough treatment of the theory.

For a general system given by,

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{N}\boldsymbol{v}_{1,k} \\ \boldsymbol{y}_k &= \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{D}\boldsymbol{u}_k + \boldsymbol{v}_{2,k} \end{aligned} \tag{14.9}$$

where the covariance matrices for $\boldsymbol{v}_{1,k}$ and $\boldsymbol{v}_{2,k}$ are given by $\boldsymbol{R}_{1,k}$ and $\boldsymbol{R}_{2,k}$, a linear estimator for the states is given by

$$\hat{\boldsymbol{x}}_{k+1} = \boldsymbol{A}\hat{\boldsymbol{x}}_k + \boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{C}\hat{\boldsymbol{x}}_k) \tag{14.10}$$

The gain $\boldsymbol{K}_k$ in the estimator is in the time varying Kalman filter calculated as

$$\boldsymbol{K}_k = \boldsymbol{A}\boldsymbol{P}_k\boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{P}_k\boldsymbol{C}^T + \widehat{\boldsymbol{R}}_{2,k})^{-1}$$

where it is assumed that there is no cross correlation between $\boldsymbol{v}_{1,k}$ and $\boldsymbol{v}_{2,k}$. $\boldsymbol{P}_{k+1}$ is calculated as

$$\boldsymbol{P}_{k+1} = \boldsymbol{A}\boldsymbol{P}_k\boldsymbol{A}^T + \boldsymbol{N}\widehat{\boldsymbol{R}}_{1,k}\boldsymbol{N}^T - \boldsymbol{A}\boldsymbol{P}_k\boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{P}_k\boldsymbol{C}^T + \widehat{\boldsymbol{R}}_{2,k})^{-1}(\boldsymbol{A}\boldsymbol{P}_k\boldsymbol{C}^T)^T$$

Note that the matrices, $\widehat{\boldsymbol{R}}_{1,k}$ and $\widehat{\boldsymbol{R}}_{2,k}$ can depend on k and that they can be used as design variables to get a desired behavior of the Kalman filter.

When the time varying Kalman filter described above is applied to the system in (14.7) the estimation procedure can be formulated according to,

$$\hat{\boldsymbol{z}}_{k+1} = \hat{\boldsymbol{z}}_k - \boldsymbol{G}(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k) + \boldsymbol{K}_k(\boldsymbol{y}_k - \hat{\boldsymbol{z}}_k) \tag{14.11a}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_k(\boldsymbol{P}_k + \widehat{\boldsymbol{R}}_{n,k})^{-1} \tag{14.11b}$$

$$\boldsymbol{P}_{k+1} = \boldsymbol{P}_k + \widehat{\boldsymbol{R}}_{\Delta_d,k} - \boldsymbol{P}_k(\boldsymbol{P}_k + \widehat{\boldsymbol{R}}_{n,k})^{-1}\boldsymbol{P}_k \tag{14.11c}$$

where it is assumed that $\boldsymbol{\Delta}_{d_k}$ and $\boldsymbol{n}_k$ are uncorrelated.

14.1.3 An optimization based approach to ILC

Consider the following criterion for (14.1),

$$J_k = \boldsymbol{z}_k^T\boldsymbol{W}_z\boldsymbol{z}_k + \boldsymbol{u}_k^T\boldsymbol{W}_u\boldsymbol{u}_k + (\boldsymbol{u}_k - \boldsymbol{u}_{k-1})^T\boldsymbol{W}_{\Delta u}(\boldsymbol{u}_k - \boldsymbol{u}_{k-1}) \tag{14.12}$$

By minimizing (14.12) it is possible to find an optimal input to the system, with respect to the criterion. This has been studied in for example Bien and Xu (1998), Amann et al. (1995a,b), Phan (1998), and Gunnarsson and Norrlöf (1999b). Compare also with the criterion chosen in Section 8.2.5. Note that the main difference is that here the last term is explicitly included in the criterion while in Section 8.2.5 it was added because of the constraint on the updating of the control signal. The choice in (14.12) gives additional freedom in the design since, instead of picking a scalar λ as in (8.4), a positive definite matrix $\boldsymbol{W}_{\Delta u}$ is chosen.

By using the definition of $\boldsymbol{z}_k$ from (14.3) in (14.12) and taking the derivative with respect to $\boldsymbol{u}_k$ it follows that

$$\frac{\partial J_k}{\partial \boldsymbol{u}_k} = ((\boldsymbol{G}^0)^T \boldsymbol{W}_z \boldsymbol{G}^0 + \boldsymbol{W}_u)\boldsymbol{u}_k + \boldsymbol{W}_{\Delta u}(\boldsymbol{u}_k - \boldsymbol{u}_{k-1}) + (\boldsymbol{G}^0)^T \boldsymbol{W}_z \boldsymbol{d}_k$$

Now solve for $\boldsymbol{u}_k$ when $\frac{\partial J_k}{\partial \boldsymbol{u}_k} = 0$. This leads to

$$\boldsymbol{u}^*_{k+1} = ((\boldsymbol{G}^0)^T \boldsymbol{W}_z \boldsymbol{G}^0 + \boldsymbol{W}_u + \boldsymbol{W}_{\Delta u})^{-1}(\boldsymbol{W}_{\Delta u}\boldsymbol{u}^*_k - (\boldsymbol{G}^0)^T \boldsymbol{W}_z \boldsymbol{d}_{k+1}) \tag{14.13}$$

where the * indicates that this is the optimal input.

If $\boldsymbol{W}_u = 0$ and $\boldsymbol{d}_{k+1}$ is known, then the updating scheme for the control $\boldsymbol{u}_k$ becomes

$$\boldsymbol{u}^*_{k+1} = ((\boldsymbol{G}^0)^T \boldsymbol{W}_z \boldsymbol{G}^0 + \boldsymbol{W}_{\Delta u})^{-1}(\boldsymbol{W}_{\Delta u}\boldsymbol{u}^*_k - (\boldsymbol{G}^0)^T \boldsymbol{W}_z \boldsymbol{d}_{k+1}) \tag{14.14}$$

Note that this expression actually contains a feedforward from the disturbance $\boldsymbol{d}_{k+1}$.

It is also clear why it is important for this algorithm to have the term $\|\boldsymbol{u}_k - \boldsymbol{u}_{k-1}\|_{W_{\Delta u}}$ in the criterion. Without this term the algorithm is not iterative. From a practical point of view (14.14) is not very useful since when $\boldsymbol{u}_{k+1}$ is calculated, $\boldsymbol{d}_{k+1}$ is in general not available. If $\boldsymbol{d}$ does not change as a function of iteration it will however work since old measurements of $\boldsymbol{d}$ can be used.

If the weights $\boldsymbol{W}_u$ and $\boldsymbol{W}_{\Delta u}$ are set to zero and equal weights are chosen for all $\boldsymbol{z}$, i.e., $\boldsymbol{W}_z = c \cdot I$, then the result of (14.13) becomes the well known inverse system solution,

$$\boldsymbol{u}^*_k = (\boldsymbol{G}^0)^{-1}\boldsymbol{d}_k$$

which was discussed in Section 3.2.

In practice the control solution can, of course, not use the true system description. If instead a model of the system is used the control signal u_k can be calculated as

$$\boldsymbol{u}_{k+1} = (\boldsymbol{G}^T \boldsymbol{W}_z \boldsymbol{G} + \boldsymbol{W}_u + \boldsymbol{W}_{\Delta u})^{-1}(\boldsymbol{W}_{\Delta u}\boldsymbol{u}_k - \boldsymbol{G}^T \boldsymbol{W}_z \hat{\boldsymbol{d}}_{k+1}) \tag{14.15}$$

In (14.15) it is also taken into account that the true $\boldsymbol{d}_k$ is not available directly as a measured signal. An estimate of $\boldsymbol{d}_k$ is found as

$$\hat{\boldsymbol{d}}_k = \hat{\boldsymbol{z}}_k - \boldsymbol{G}\boldsymbol{u}_k \tag{14.16}$$

which means that the expression for u_k can be simplified to

$$\boldsymbol{u}_{k+1} = (\boldsymbol{W}_u + \boldsymbol{W}_{\Delta u})^{-1}(\boldsymbol{W}_{\Delta u}\boldsymbol{u}_k - \boldsymbol{G}^T\boldsymbol{W}_z\hat{\boldsymbol{z}}_{k+1}) \tag{14.17}$$

Together with the observer in (14.11) this gives the ILC updating scheme

$$\begin{aligned}\boldsymbol{u}_{k+1} &= (\boldsymbol{W}_u + \boldsymbol{W}_{\Delta u} - \boldsymbol{G}^T\boldsymbol{W}_z\boldsymbol{G})^{-1}\big((\boldsymbol{W}_{\Delta u} - \boldsymbol{G}^T\boldsymbol{W}_z\boldsymbol{G})\boldsymbol{u}_k - \\ &\qquad \boldsymbol{G}^T\boldsymbol{W}_z((I - \boldsymbol{K}_k)\hat{\boldsymbol{z}}_k + \boldsymbol{K}_k\boldsymbol{y}_k)\big) \\ \hat{\boldsymbol{z}}_{k+1} &= \hat{\boldsymbol{z}}_k - \boldsymbol{G}(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k) + \boldsymbol{K}_k(\boldsymbol{y}_k - \hat{\boldsymbol{z}}_k)\end{aligned} \tag{14.18}$$

which involves two iterative updating formulas. One for the control input $\boldsymbol{u}_k$ and one for the estimated output $\hat{\boldsymbol{z}}_k$. Note that in the implementation of the ILC algorithm there will be one more iterative updating formula since also $\boldsymbol{P}_k$ has to be calculated.

If $\boldsymbol{W}_{\Delta u}$ is chosen as $\boldsymbol{W}_{\Delta u} = 0$, the updating formula for $\boldsymbol{u}_k$ in (14.17) becomes only a linear function of $\hat{\boldsymbol{z}}_k$. This can be plugged into the observer in (14.8) resulting in

$$\hat{\boldsymbol{z}}_{k+1} = \hat{\boldsymbol{z}}_k + (I + \boldsymbol{G}\boldsymbol{W}_u^{-1}\boldsymbol{G}^T\boldsymbol{W}_z)^{-1}\boldsymbol{K}_k(\boldsymbol{y}_k - \hat{\boldsymbol{z}}_k) \tag{14.19}$$

Together with (14.17) and the calculation of $\boldsymbol{K}_k$ from the previous section, this gives an ILC scheme with only two iterative updating formulas, including the one for $\boldsymbol{P}_k$. The case where $\boldsymbol{W}_{\Delta u} = 0$ in (14.12) is what will be considered in the rest of the discussion in this part of the thesis. Compared to Section 8.2.5 this means that the iterative behavior of the ILC algorithm has moved from the updating of the control signal to the estimator. As noted above in (14.17) and (14.18) it is also possible to let $\boldsymbol{W}_{\Delta u} \neq 0$ and use three iterative updating formulas for the ILC but this is not covered here.

14.1.4 Relations to other ILC updating schemes

Consider the case where $\widehat{\boldsymbol{R}}_{n,k} = \hat{r}_{n,k} \cdot I$ and $\widehat{\boldsymbol{R}}_{\Delta_d,k} = \hat{r}_{\Delta_d,k} \cdot I$. Assume also that the estimator is calculated according to a time varying Kalman filter as described in Section 14.1.2. Note that in the calculation of $\boldsymbol{P}_k$ the measured values of $\boldsymbol{y}_k$ are not utilized. Instead the value of $\boldsymbol{P}_k$ is completely dependent on the initial value, $\boldsymbol{P}_0$. This initial choice indicates how well the initial estimate of $\boldsymbol{z}$ describes the real value.

Assume that $\boldsymbol{P}_0 = p_0 \cdot I$, this means that $\boldsymbol{K}_k$ and $\boldsymbol{P}_k$ will be equal to $\kappa_k \cdot I$ and $p_k \cdot I$ respectively. Since $\boldsymbol{K}_k$ is a scalar times an identity matrix it is clear that the matrix $\boldsymbol{K}_k$ commutes with all other matrices. In particular, this means that we can rewrite (14.19) according to

$$\begin{aligned}\boldsymbol{u}_{k+1} = (I - (I + \boldsymbol{W}_u^{-1}\boldsymbol{G}^T\boldsymbol{W}_z\boldsymbol{G})^{-1}\boldsymbol{K}_k)\boldsymbol{u}_k \\ + \boldsymbol{W}_u^{-1}\boldsymbol{G}^T\boldsymbol{W}_z(I + \boldsymbol{G}\boldsymbol{W}_u^{-1}\boldsymbol{G}^T\boldsymbol{W}_z)^{-1}\boldsymbol{K}_k\boldsymbol{y}_k\end{aligned} \tag{14.20}$$

where (14.15) is used together with the fact that the following equality holds

$$\boldsymbol{W}_u^{-1}G^T\boldsymbol{W}_z(I+G\boldsymbol{W}_u^{-1}G^T\boldsymbol{W}_z)^{-1}\boldsymbol{K}_k = (I+\boldsymbol{W}_u^{-1}G^T\boldsymbol{W}_zG)^{-1}\boldsymbol{K}_k\boldsymbol{W}_u^{-1}G^T\boldsymbol{W}_z$$

when $\boldsymbol{K}_K$ is a scalar times an identity matrix. If the weights $\boldsymbol{W}_u$ and $\boldsymbol{W}_z$ are chosen such that $\boldsymbol{W}_u = I$ and $\boldsymbol{W}_z = \zeta I$, it means that the updating equation for the control signal in (14.20) becomes

$$\boldsymbol{u}_{k+1} = (I-(I+\zeta\boldsymbol{G}^T\boldsymbol{G})^{-1}\kappa_k)\boldsymbol{u}_k + \zeta\kappa_k\boldsymbol{G}^T(I+\zeta\boldsymbol{G}\boldsymbol{G}^T)^{-1}\boldsymbol{y}_k$$

Now, if ζ is chosen very large it follows that

$$\boldsymbol{u}_{k+1} \approx \boldsymbol{u}_k + \kappa_k\boldsymbol{G}^{-1}\boldsymbol{y}_k \tag{14.21}$$

which is recognized as a standard approach (although the gain κ_k is non standard), see e.g., Moore (1993) or Algorithm 8.2 in Chapter 8. Compare also the algorithm derived in Section 3.5.

As a result of the fact that $\widehat{\boldsymbol{R}}_{n,k}$, $\widehat{\boldsymbol{R}}_{\Delta_d,k}$, and $\boldsymbol{P}_0$ are all equal to a scalar times an identity matrix, (14.11b) and (14.11c) can be written as scalar equations according to,

$$\kappa_k = \frac{p_k}{p_k+\hat{r}_{n,k}}$$

and

$$p_{k+1} = \frac{p_k\hat{r}_{n,k}}{p_k+\hat{r}_{n,k}} + \hat{r}_{\Delta_d,k}$$

Assume that $\hat{r}_n$ and $\hat{r}_{\Delta_d}$ are k-independent, then it is possible to find the limit value, p_∞, by solving

$$p_\infty = \frac{p_\infty\hat{r}_n}{p_\infty+\hat{r}_n} + \hat{r}_{\Delta_d}$$

which gives

$$p_\infty = \frac{\hat{r}_{\Delta_d}}{2}\left(1+\sqrt{1+2\frac{\hat{r}_n}{\hat{r}_{\Delta_d}}}\right) \tag{14.22}$$

Note that the value of p_∞ depends on the actual value of $\hat{r}_{\Delta_d}$ while for κ_∞ it is only the value of $\frac{\hat{r}_{\Delta_d}}{\hat{r}_n}$ that matters. This means that multiplying both $\hat{r}_{\Delta_d}$ and $\hat{r}_n$ with the same factor, the value of κ_∞ will not change.

If it is assumed that $\boldsymbol{d}_k$ is k-independent, i.e., $\hat{r}_{\Delta_d} = 0$, it is obvious that $p_\infty = 0$ which also implies that $\kappa_\infty = 0$. It is perhaps more interesting to study the transient behavior of p_k and κ_k. If the initial guess of $\hat{z}_0$ is not so reliable, it is

reasonable to assume that p_0 is chosen as a large number. If $p_0 \gg \hat{r}_n$ this means that $\kappa_0 \approx 1$ and since

$$p_{k+1} = \frac{p_k \hat{r}_n}{p_k + \hat{r}_n} \tag{14.23}$$

it follows that $p_1 \approx \hat{r}_n$ which in turn implies that $k_1 \approx \frac{1}{2}$. By considering (14.23) for general k it becomes obvious that, in fact, $p_k \approx \frac{\hat{r}_n}{k}$ for all $k > 0$ and hence $\kappa_k \approx \frac{1}{k+1}$.

Note that combining the result in (14.21) with the one obtained when $\hat{r}_{\Delta_d}$ is assumed to be zero and $p_0 \gg \hat{r}_n$ the resulting ILC becomes

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \frac{\boldsymbol{G}^{-1}}{k+1}\boldsymbol{y}_k$$

which is similar to the result in (3.30). The difference is that in (3.30) the system is described using a transfer operator model while here the system mapping is described using a matrix $\boldsymbol{G}$.

14.1.5 An adaptive algorithm for ILC

The calculations of $\boldsymbol{P}_k$ and $\boldsymbol{K}_k$ in the time varying Kalman filter in Section 14.1.2 do not include the measurements from the system. In this section a possible extension to the algorithm presented in the previous sections is given. The new algorithm takes advantage of the measurements from the system and use them to adapt a measure of the variability of the system disturbance, $\widehat{\boldsymbol{R}}_{\Delta,k}$. The algorithm is adaptive since the value of $\boldsymbol{K}_k$ will depend on the variability measure through $\boldsymbol{P}_k$.

To explain the idea behind the measure of variability used in the algorithm first note that the system model $\boldsymbol{G}$ does not capture the true system dynamics perfectly. Instead the relation given by (14.7) describes the true system in terms of the model and the uncertainty.

The idea is to use

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k - \boldsymbol{G}(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k) - \underbrace{\boldsymbol{G}\boldsymbol{\Delta}_G(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k) + \boldsymbol{\Delta}_{d_k}}_{\Delta} \tag{14.24}$$

and find a measure of the size of the variation of Δ. The following equation gives a measure of the variability of Δ in (14.24)

$$\hat{r}_{\Delta,k} = \frac{1}{n-1}(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k)^T \hat{\boldsymbol{\Delta}}_G^T \boldsymbol{G}^T \boldsymbol{G} \hat{\boldsymbol{\Delta}}_G (\boldsymbol{u}_{k+1} - \boldsymbol{u}_k) + \hat{r}_{\Delta_d} \tag{14.25}$$

where $\hat{\boldsymbol{\Delta}}_G$ is an estimate of the true model uncertainty and $\hat{r}_{\Delta_d}$ is an estimate of the variance of $\boldsymbol{\Delta}_{d_k}$. The algorithm can now be formulated.

Algorithm 14.1 (Adaptive optimization based ILC algorithm)

1. Design an ILC updating equation using the LQ design in Section 14.1.3.
2. Assume $\widehat{\boldsymbol{R}}_{\Delta_d}$ and $\widehat{\boldsymbol{R}}_n$ diagonal with the diagonal elements equal to $\hat{r}_{\Delta_d}$ and $\hat{r}_n$ respectively. Choose $\hat{r}_{\Delta_d}$ and $\hat{r}_n$ from physical insight or such that p_∞ in (14.22) and the corresponding κ_∞ get the desired values.
3. Let $\hat{z}_0 = 0$.
4. Choose an initial value for p_0. This can be a large number since it will converge to $p_1 \approx \hat{r}_n + \hat{r}_{\Delta_d}$ already after one iteration.
5. Implementation of the ILC algorithm:
   (a) Let $k = 0$, and $\boldsymbol{u}_0 = \boldsymbol{W}_u^{-1}\boldsymbol{G}^T\boldsymbol{W}_z\hat{\boldsymbol{z}}_0$.
   (b) Apply $\boldsymbol{u}_k$ and measure $\boldsymbol{y}_k$.
   (c) Calculate,

$$\begin{aligned}
\kappa_k &= \frac{p_k}{p_k + \hat{r}_n} \\
\hat{\boldsymbol{z}}_{k+1} &= \hat{\boldsymbol{z}}_k + (I + \boldsymbol{G}\boldsymbol{W}_u^{-1}\boldsymbol{G}^T\boldsymbol{W}_z)^{-1}\kappa_k(\boldsymbol{y}_k - \hat{\boldsymbol{z}}_k) \\
\boldsymbol{u}_{k+1} &= \boldsymbol{W}_u^{-1}\boldsymbol{G}^T\boldsymbol{W}_z\hat{\boldsymbol{z}}_{k+1} \\
\hat{r}_{\Delta,k} &= \frac{1}{n-1}(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k)^T\hat{\boldsymbol{\Delta}}_G^T\boldsymbol{G}^T\boldsymbol{G}\hat{\boldsymbol{\Delta}}_G(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k) + \hat{r}_{\Delta_d} \\
p_{k+1} &= \frac{p_k\hat{r}_n}{p_k + \hat{r}_n} + \hat{r}_{\Delta,k}
\end{aligned}$$

   (d) Let $k = k + 1$. Start again from (b).

It is important to understand the properties of the proposed algorithm and this, especially, includes stability and performance. It is well known that the analysis of adaptive algorithms is not easy nor straightforward but there exist some methods as pointed out by, e.g., Åström and Wittenmark (1995). In the next section the adaptive algorithm presented in this section is analyzed with respect to stability and performance but first some notes on the design and implementation of the algorithm will be given.

14.1.6 Design and implementation issues for the optimization based approach to ILC

When designing an ILC scheme using the linear quadratic optimal control approach discussed in the previous section it is of great importance to understand the possibilities that are involved and how these effect the result.

The design process involves a lot of steps and there are many degrees of freedom in the design process. The design parameters involved are:

- In the LQ design,
  - $\boldsymbol{G} \in \mathbb{R}^{n\times n}$
  - $\boldsymbol{W}_z \in \mathbb{R}^{n\times n}$
  - $\boldsymbol{W}_u \in \mathbb{R}^{n\times n}$
- In the Kalman filter,
  - $\boldsymbol{G} \in \mathbb{R}^{n\times n}$
  - $p_0 \in \mathbb{R}$
  - $\hat{r}_{\Delta_d} \in \mathbb{R}$
  - $\hat{r}_n \in \mathbb{R}$
  - $\hat{\boldsymbol{\Delta}}_G \in \mathbb{R}^{n\times n}$

The model $\boldsymbol{G}$ is used in both the LQ design and the Kalman filter. By just considering the number of possibilities that are offered by these parameters it might seem that the usefulness of the proposed scheme can be questioned. From a user's point of view it is important that the number of parameters is small and that the effect of the parameters are easy to understand. Note that the suggested parameters, given above, also imply a simplification compared to the originally proposed algorithm. Only scalar p_k and κ_k are considered here. The implication of the different design parameters is discussed next.

Design scheme

Assume that the model of the system, $\boldsymbol{G} \in \mathbb{R}^{n\times n}$, is available from an identification experiment. This experiment can also give an idea on which kind of uncertainties are present in the model, i.e., the size of $\hat{\boldsymbol{\Delta}}_G$. In many traditional design schemes for ILC the following updating equation is used,

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}(\boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{e}_k) \tag{14.26}$$

where $\boldsymbol{u}_k, \boldsymbol{e}_k \in \mathbb{R}^n$ and $\boldsymbol{Q}, \boldsymbol{L} \in \mathbb{R}^{n\times n}$. Often it is suggested that, for robustness of the ILC algorithm, $\boldsymbol{Q}$ should be chosen as a realization of a low pass filter. This makes the ILC method robust against model errors at high frequencies, where usually the model of the system does not capture the true dynamics very well. The LQ solution of the ILC problem can take this into consideration by introducing a kind of frequency domain weighting in the optimization criterion (14.12). This is done by using the fact that the matrices $\boldsymbol{W}_z$ and $\boldsymbol{W}_u$ do not have to be diagonal. With a frequency domain perspective to the optimization problem it is possible to say that in the criterion, high frequencies in the control signal $\boldsymbol{u}_k$ should have a higher weight than low frequencies. This can be done by choosing the matrix $\boldsymbol{W}_u^{-1}$ as a realization of a zero phase low pass filter with cut-off frequency at the desired bandwidth of the ILC algorithm. To create such a matrix let $\boldsymbol{H}$ be a

lower triangular Toeplitz matrix with the first column being the n first Markov parameters for a low pass filter, e.g., a Butterworth filter. Next define

$$\boldsymbol{W}_u = (\boldsymbol{H}\boldsymbol{H}^T)^{-1} \tag{14.27}$$

i.e., the inverse of the zero phase low pass filter $\boldsymbol{H}\boldsymbol{H}^T$. The matrix $\boldsymbol{W}_z$ is here simply chosen as a scalar times an identity matrix i.e.,

$$\boldsymbol{W}_z = \zeta \cdot I \tag{14.28}$$

and the value of ζ will decide how much the ILC scheme should try to resemble the inverse system approach, as was also discussed in Section 14.1.4.

For the Kalman filter, the system model $\boldsymbol{G}$ and an estimated model uncertainty $\hat{\boldsymbol{\Delta}}_G$ are supposed to be available from the identification experiments. The algorithm is not sensitive to the initial value of p_0 as was noted in Section 14.1.4. If the value is initially set to be a large number the value of κ_0 will be close to one and the next value of p_1 will be

$$p_1 \approx \hat{r}_n + \hat{r}_{\Delta,0}$$

This shows that the initial value is not so important for the behavior of the algorithm as long as it is large enough.

The values of $\hat{r}_{\Delta_d}$ and $\hat{r}_n$ are still to be chosen. As was shown in Section 14.1.4 it is true that asymptotically, if $\|\boldsymbol{u}_{k+1} - \boldsymbol{u}_k\|$ becomes small, it is only the value of $\frac{\hat{r}_{\Delta_d}}{\hat{r}_n}$ that has an impact on the value of κ_k. To decide the value of the two parameters the following strategy will be used here: Let the value of $\hat{r}_n$ be based on physical knowledge of the process and adjust $\hat{r}_{\Delta_d}$ such that the limit value of p_∞ in (14.22) and the corresponding κ_∞ have the right value. Note that it is important that the value of $\hat{r}_{\Delta_d}$ is chosen not too large. A too large value would imply that $\hat{r}_{\Delta,k}$ is only determined by the value of $\hat{r}_{\Delta_d}$. The algorithm would in this case loose the adaptivity and the gain would decrease as $\frac{1}{k+1}$.

Implementation

The matrix approach to ILC, described so far in this chapter, might be difficult to implement in practice and there are, at least, two reasons why. First, the matrices have to be stored in a memory and when the matrices become large this will be a critical problem since the space necessary will be significant. Second, the calculation of the matrices is not trivial and takes time. The calculation of the inverse of an $n \times n$ matrix is a computationally intense task when n is large.

One way of making the calculations much less demanding is to use a technique presented in Gunnarsson and Norrlöf (1999b) and also discussed in Section 8.2.5. The idea is to do a frequency domain interpretation of the matrix multiplications and try to come back to a transfer operator formulation instead of the matrix formulation. This means that the goal is to come back to the description in (14.3).

The discussion on how this should be done starts from the LQ design. The model $G(q)$ and the model uncertainty $\hat{\Delta}_G(q)$ are assumed to be known from the modeling experiments. The matrix H is easily chosen as $H(q)$ being a Butterworth filter. With this choice, $W_u^{-1}(q)$ becomes,

$$W_u^{-1}(q) = H(q)H(q^{-1})$$

where $H(q^{-1})$ correspond to the matrix $\boldsymbol{H}^T$. $W_z(q)$ becomes

$$W_z(q) = \zeta$$

i.e., just a scalar.

For the choice of the parameters in the Kalman filter, it is assumed that the system model and the system uncertainty are given. The other parameters are not different from the case where the system description is in matrix form and therefore the same arguments apply as in the previous section.

The implementation is now presented as an algorithm.

Algorithm 14.2 (Filter implementation of Algorithm 14.1, version 1)

1. Let $k = 0$, and $u_0(t) = \zeta W_u^{-1}(q)G(q^{-1})\hat{z}_0(t)$.
2. Apply $u_k(t)$ and measure $y_k(t)$, for $0 \leq t \leq n-1$.
3. Calculate,

$$\begin{aligned}
\kappa_k &= \frac{p_k}{p_k + \hat{r}_n} \\
\hat{z}_{k+1}(t) &= \hat{z}_k(t) + (1 + \zeta W_u^{-1}(q)G(q)G(q^{-1}))^{-1}\kappa_k(y_k(t) - \hat{z}_k(t)) \\
u_{k+1}(t) &= \zeta W_u^{-1}(q)G(q^{-1})\hat{z}_{k+1}(t) \\
\hat{r}_{\Delta,k} &= \frac{1}{n-1}\sum_{\tau=1}^{n}\Big(G(q)\Delta_G(q)(u_{k+1}(\tau) - u_k(\tau))\Big)^2 + \hat{r}_{\Delta_d} \\
p_{k+1} &= \frac{p_k\hat{r}_n}{p_k + \hat{r}_n} + \hat{r}_{\Delta,k}
\end{aligned}$$

4. Let $k = k + 1$. Start again from (2).

The sampling time is assumed to be equal to 1.

From an implementation point of view it might be convenient to rewrite the actual implementation, compared to Algorithm 14.2. First note that in the calculation of $\hat{\boldsymbol{z}}_{k+1}$ it is actually possible to formulate the inverse filter $(1+\zeta W_u^{-1}(q)G(q)G(q^{-1}))^{-1}$ according to

$$(1 + \zeta W_u^{-1}(q)G(q)G(q^{-1}))^{-1} = (1 + \zeta G(q)H(q)G(q^{-1})H(q^{-1}))^{-1}$$

which is clearly a zero phase filtering operation. It is actually possible to go one step further and find a filter $F(q)$ such that

$$(1+\zeta G(q)H(q)G(q^{-1})H(q^{-1}))^{-1} = F(q)F(q^{-1}) \tag{14.29}$$

and this filtering operation is easy to implement in MATLAB™.

To find the factorization in (14.29) use the fact that $G(q)$ and $H(q)$ can be described according to

$$G(q) = \frac{b_G(q)}{a_G(q)}, \quad H(q) = \frac{b_H(q)}{a_H(q)}$$

where $a_G(q), b_G(q), a_H(q)$ and $b_H(q)$ are all polynomials in q. This means that it is possible to write

$$\begin{aligned} F(q)F(q^{-1}) &= (1+\zeta G(q)H(q)G(q^{-1})H(q^{-1}))^{-1} \\ &= \frac{a_G(q)a_H(q)a_G(q^{-1})a_H(q^{-1})}{a_G(q)a_H(q)a_G(q^{-1})a_H(q^{-1}) + b_G(q)b_H(q)b_G(q^{-1})b_H(q^{-1})} \end{aligned}$$

and the numerator polynomial of $F(q) = \frac{b_F(q)}{a_F(q)}$ is easily found as

$$b_F(q) = a_G(q)a_H(q) \tag{14.30}$$

To find the denominator, $a_F(q)$, first find the roots of

$$a_G(q)a_H(q)a_G(q^{-1})a_H(q^{-1}) + b_G(q)b_H(q)b_G(q^{-1}) \tag{14.31}$$

Create a polynomial, $\tilde{b}_F(q)$, having the stable part of the roots of (14.31) as its roots. The polynomial $b_F(q)$ is found by calculating the constant γ such that

$$b_F(q) = \gamma\tilde{b}_F(q) \tag{14.32}$$

and

$$\gamma^2(\tilde{b}_F(1))^2 = (a_G(1)a_H(1))^2 + (b_G(1)b_H(1))^2 \tag{14.33}$$

The sign of the solution does not matter but for convenience it can be chosen as $\gamma > 0$.

By using the equation describing how to calculate $u_k(t)$ from $\hat{z}_k(t)$ it is also possible to formulate $\hat{r}_{\Delta,k}$ as

$$\begin{aligned} \hat{r}_{\Delta,k} &= r_{\Delta_d}\frac{\zeta}{n-1} \\ &\sum_{\tau=1}^{n}\Big(\Delta_G(q)W_u^{-1}(q)G(q)G(q^{-1})F(q)F(q^{-1})\kappa_k(y_k(\tau)-\hat{z}_k(\tau))\Big)^2 \end{aligned} \tag{14.34}$$

which obviously can be calculated together with $\hat{z}_{k+1}(t)$. A new algorithm taking these updates into account is now formulated.

Algorithm 14.3 (Filter implementation of Algorithm 14.1, version 2)

1. Find $F(q)$ such that
$$F(q)F(q^{-1}) = (1 + \zeta W_u^{-1}(q)G(q)G(q^{-1}))^{-1}$$
2. Let $k = 0$, and $u_0(t) = \zeta W_u^{-1}(q)G(q^{-1})\hat{z}_0(t)$.
3. Apply $u_k(t)$ and measure $y_k(t)$, for $1 \leq t \leq n$.
4. Calculate,
$$\begin{aligned}
\kappa_k &= \frac{p_k}{p_k + \hat{r}_n} \\
x_{temp}(t) &= F(q)F(q^{-1})\kappa_k(y_k(t) - \hat{z}_k(t)) \\
\hat{z}_{k+1}(t) &= \hat{z}_k(t) + x_{temp}(t) \\
u_{k+1}(t) &= \zeta W_u^{-1}(q)G(q^{-1})\hat{z}_{k+1}(t) \\
\hat{r}_{\Delta,k} &= \frac{\zeta^2}{n-1}\sum_{\tau=1}^{n}\left(\Delta_G(q)W_u^{-1}(q)G(q)G(q^{-1})x_{temp}(t)\right)^2 + \hat{r}_{\Delta_d} \\
p_{k+1} &= \frac{p_k\hat{r}_n}{p_k + \hat{r}_n} + \hat{r}_{\Delta,k}
\end{aligned}$$
5. Let $k = k + 1$. Start again from (3).

The adaptive ILC algorithm in Algorithm 14.3 can easily be implemented in for example MATLAB™.

14.2 Analysis of the Adaptive ILC Algorithm

The analysis of adaptive systems is, generally, very difficult. The discussion here will be restricted to the adaptive algorithm for ILC presented in the previous section under some particular assumptions on the model uncertainty.

One property that will be used in the analysis is that κ_k is bounded which can be seen easily from
$$\kappa_k = \frac{1}{1 + \frac{r_n}{p_k}}$$
Since $p_k > 0$ it is obvious that the value of κ_k is restricted to $0 < \kappa_k < 1$ where the lower bound is reached when $p_k \to 0$ and the upper bound is reached when $p_k \to \infty$. This leads to a sufficient condition for stability of the adaptive ILC algorithm. First, however, some assumptions are made on the disturbances,
$$\|n_k\|_\infty \leq \gamma_n, \quad \|d_k\|_\infty \leq \gamma_d \tag{14.35}$$

If it is assumed that n_k is uniformly distributed the first assumption is easily satisfied. The second is more technically difficult but very straightforward from an application point of view since $\|d_k\|_\infty$ will always be bounded in practice.

Theorem 14.1 (Sufficient condition for boundedness of $\hat{z}_k$)
A sufficient condition for the adaptive ILC algorithm (Algorithm 14.1) to give a bounded $\hat{\boldsymbol{z}}_k$ is that

$$\sup_{0<\kappa_k<1} \rho\big((1-\kappa_k)\cdot I - \kappa_k(I + \boldsymbol{G}\boldsymbol{W}_u^{-1}\boldsymbol{G}^T\boldsymbol{W}_z)^{-1}\boldsymbol{G}\boldsymbol{\Delta}_G\boldsymbol{W}_u^{-1}\boldsymbol{G}^T\boldsymbol{W}_z\big) < 1$$

Proof Use that

$$\hat{\boldsymbol{z}}_{k+1} = \hat{\boldsymbol{z}}_k + (I + \boldsymbol{G}\boldsymbol{W}_u^{-1}\boldsymbol{G}^T\boldsymbol{W}_z)^{-1}\kappa_k(\boldsymbol{y}_k - \hat{\boldsymbol{z}}_k)$$

and

$$\boldsymbol{y}_k = -\boldsymbol{G}(I + \boldsymbol{\Delta}_G)\boldsymbol{W}_u^{-1}\boldsymbol{G}^T\boldsymbol{W}_z\hat{\boldsymbol{z}}_k + \boldsymbol{d}_k + \boldsymbol{n}_k$$

This means that it is possible to write the estimate, $\hat{\boldsymbol{z}}_{k+1}$, as

$$\begin{aligned}\hat{\boldsymbol{z}}_{k+1} = &\big((1-\kappa_k)\cdot I - \kappa_k(I + \boldsymbol{G}\boldsymbol{W}_u^{-1}\boldsymbol{G}^T\boldsymbol{W}_z)^{-1}\boldsymbol{G}\boldsymbol{\Delta}_G\boldsymbol{W}_u^{-1}\boldsymbol{G}^T\boldsymbol{W}_z\big)\hat{\boldsymbol{z}}_k \\ &+ (I + \boldsymbol{G}\boldsymbol{W}_u^{-1}\boldsymbol{G}^T\boldsymbol{W}_z)^{-1}\kappa_k(\boldsymbol{d}_k + \boldsymbol{n}_k)\end{aligned}$$

This can be rewritten on the following form,

$$\hat{\boldsymbol{z}}_{k+1} = (I - \kappa_k(I + \tilde{F}))\hat{\boldsymbol{z}}_k + (I + \boldsymbol{G}\boldsymbol{W}_u^{-1}\boldsymbol{G}^T\boldsymbol{W}_z)^{-1}\kappa_k(\boldsymbol{d}_k + \boldsymbol{n}_k)$$

with

$$\tilde{F} = (I + \boldsymbol{G}\boldsymbol{W}_u^{-1}\boldsymbol{G}^T\boldsymbol{W}_z)^{-1}\boldsymbol{G}\boldsymbol{\Delta}_G\boldsymbol{W}_u^{-1}\boldsymbol{G}^T\boldsymbol{W}_z$$

To establish boundedness it is first necessary to show that the updating equation for $\hat{\boldsymbol{z}}_k$ is uniformly exponentially stable. Using Lemma 4.3 this condition is easily shown to be satisfied. Next uniform bounded-input bounded-state stability can be established. Using Lemma 4.2 and the fact that

$$\kappa_k\|(I + \boldsymbol{G}\boldsymbol{W}_u^{-1}\boldsymbol{G}^T\boldsymbol{W}_z)^{-1}\|$$

is always bounded since $0 < \kappa_k < 1$ and the above stated matrix is always chosen such that its norm is bounded. This concludes the proof of boundedness of $\hat{\boldsymbol{z}}_k$. ■

The condition in Theorem 14.1 is obviously not possible to test for in practice since the model uncertainty is not known. Normally some kind of knowledge is however present about the model uncertainty and using this knowledge it is possible to test for boundedness at least for the estimated model uncertainty.

Remark 14.1
Note that in the result given in Theorem 14.1 it is assumed that the model uncertainty is k-independent. If the true system is nonlinear or iteration/time variant and the model, G, is iteration/time invariant it is clear that the model error will change between iterations. In fact, if $\boldsymbol{\Delta}_G$ changes as a function of iteration the theorem will no longer guarantee the boundedness.

It is possible to move on and find conditions for boundedness of $\hat{z}_k$ also when the model uncertainty changes as a function of iteration.

Corollary 14.1 (Bounded $\hat{\boldsymbol{z}}_k$ with slowly varying system matrix)
If there exists a constant α such that, for all k

$$\sup_{0<\kappa_k<1} \|I - \kappa_k(I + F(\boldsymbol{G}_k, \boldsymbol{\Delta}_{G_k}))\| \leq \alpha$$

and the spectral radius

$$\sup_{0<\kappa_k<1} \rho\big(I - \kappa_k(I + F(\boldsymbol{G}_k, \boldsymbol{\Delta}_{G_k}))\big) \leq \mu$$

where $0 \leq \mu < 1$. The function $F(\cdot)$ is assumed to be defined according to

$$F(\boldsymbol{G}_k, \boldsymbol{\Delta}_{G_k}) = (I + \boldsymbol{G}_k \boldsymbol{W}_u^{-1} G_k^T \boldsymbol{W}_z)^{-1} \boldsymbol{G}_k \boldsymbol{\Delta}_{G_k} \boldsymbol{W}_u^{-1} \boldsymbol{G}_k^T \boldsymbol{W}_z$$

It exists a positive constant β such that $\hat{z}_k$ is bounded if

$$\|\kappa_k(I + F(\boldsymbol{G}_k, \boldsymbol{\Delta}_{G_k})) - \kappa_{k-1}(I + F(\boldsymbol{G}_k, \boldsymbol{\Delta}_{G_{k-1}}))\| \leq \beta$$

for all k.

Proof Follows readily from Theorem 4.3. ■

The result in Corollary 14.1 is not constructive since it only says that there exists a β. It says that if this is chosen sufficiently small the boundedness of $\hat{\boldsymbol{z}}_k$ can be guaranteed. This kind of results are typical for the analysis of adaptive algorithms and their robustness against model errors.

15

Example

In this chapter a design example using one of the algorithms from the previous chapter is presented. The resulting ILC algorithm is also implemented on the ABB IRB1400 industrial robot described in Part II of the thesis.

15.1 The Process

The experimental setup is similar to the one described in Section 12.2. Also here ILC is applied to 3 of the total 6 joints of the IRB1400. Each joint is modeled as a transfer operator description from the ILC control input to the measured motor position of the robot, i.e., $\boldsymbol{G}^0$ in (14.3). The procedure used to find these models is presented in Chapter 7. The models are calculated using *System Identification Toolbox* (Ljung, 1995) and are given by,

$$\begin{aligned} \hat{G}_1(q) = \hat{G}_2(q) &= \frac{0.1q^{-1}}{1 - 0.9q^{-1}} \\ \hat{G}_3(q) &= \frac{0.13q^{-1}}{1 - 0.87q^{-1}} \end{aligned} \tag{15.1}$$

The simplicity of the models comes from the fact that they all describe closed loop systems, as was also noted in Section 7.3.

15.2 Description of the Experiment

To evaluate the proposed adaptive ILC algorithm the experiment described in Section 8.3.2 is used. In Figure 8.11 the program is shown together with the resulting desired trajectory on the arm-side of the robot. The configuration of the robot is also shown in Figure 8.10. For a more thorough description of the experiment see Section 8.3.2.

To make it possible to rank the adaptive ILC algorithm, two different algorithms have been chosen for comparison. The first is a traditional ILC algorithm formulation with the updating scheme given by

$$u_{k+1}(t) = Q(q)(u_k(t) + L(q)y_k(t)) \tag{15.2}$$

The second algorithm is the same as the adaptive ILC algorithm, except that the Kalman gain κ_k is fixed to a value slightly less than one. The second case is made to show the advantage of having an adaptive gain in the updating formula.

15.3 Design

The design procedure presented in Section 14.1 shows that it is necessary to have a model of the system in order to find the ILC algorithm. In Section 15.1 the models for each of the three joints of the robot are presented and the models are represented by linear discrete time transfer functions. The design that will be used here is based on the ideas presented in Section 14.1.6 and we will use the filter based approach in the implementation.

The filter $H(q)$ is simply chosen as a second order Butterworth filter with cut-off frequency 0.2 of the Nyquist frequency. In MATLAB™ this means that

```
>> [bH,aH] = butter(2,0.2);
```

To achieve the filter $W_u^{-1}(q)$, the filter $H(q)$ is simply applied using the function `filtfilt` in MATLAB™. The filter $F(q)$ is found by using the method described in Section 14.1.6. In MATLAB™ it can be implemented as,

```
>> bF = conv(aG,aH);
>> denFF = conv(conv(aG,aH),conv(fliplr(aG),fliplr(aH))) ...
    + conv(conv(Gb,Hb),conv(fliplr(Gb),fliplr(Hb)));
>> rootsFF = roots(denFF);
>> tildeFb = poly(rootsFF(find(abs(rootsFF)<1)));
>> Fa = tildeFb/sqrt(sum(conv(tildeFb,fliplr(tildeFb)))) ...
      *sqrt(sum(denFF));
```

With the filter F given as above it is straightforward to implement the ILC algorithm according to Algorithm 14.3 in MATLAB™. It is of course necessary to

decide the values for the other design parameters. The following values are used in the experiment,

$$\begin{aligned} p_0 &= 10^4 \\ \zeta &= 10^3 \\ \hat{r}_{\Delta_d} &= 10^{-6} \\ \hat{r}_n &= 5 \cdot 10^{-5} \\ \Delta_G(q) &= 0.5 \end{aligned}$$

This means that p_∞ defined according to (14.22) becomes equal to $5.5 \cdot 10^{-6}$ and the corresponding κ_∞ becomes equal to 0.10 which is a reasonable lower limit for the gain κ_k.

The filters in the traditional ILC algorithm, given by (15.2), are chosen such that $Q(q) = \bar{Q}(q)\bar{Q}(\frac{1}{q})$, with $\bar{Q}(q)$ as a second order Butterworth filter with cut-off frequency 0.2 of the Nyquist frequency and $L(q) = 0.9q^4$, cf. Section 8.3.2.

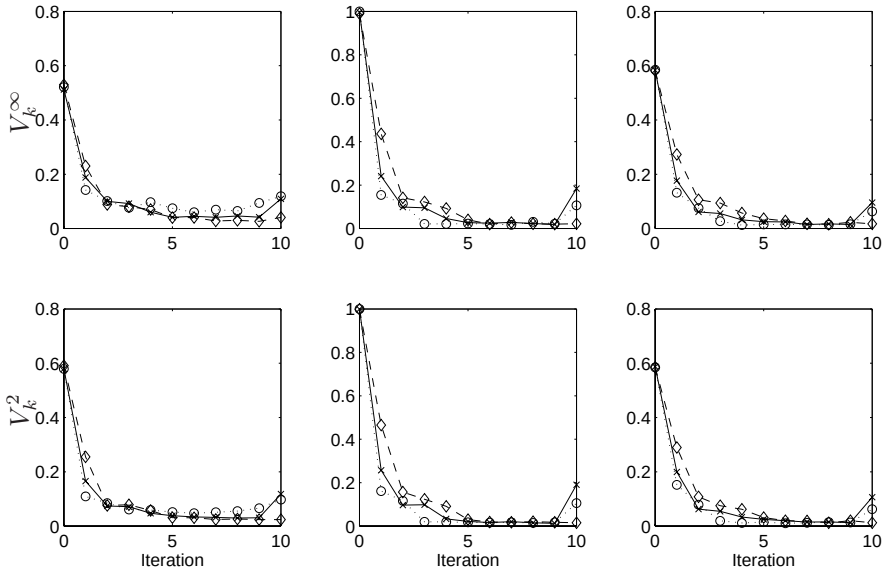
15.4 Results

Using the experiment described in Section 15.2, the three designs are tested on the robot. First using the proposed adaptive ILC scheme, then with the design according to the proposed adaptive ILC design but with fixed gain (κ_k), and finally using a "traditional" ILC updating scheme. The result fom the experiments are evaluated on the motor side of the robot. This is also where the measurements and the control are performed. The evaluation is also done on the arm side of the robot, considering the trajectory that the robot is actually following on the arm-side. This is done by using a pen mounted as a tool of the robot and making it draw the actual trajectory on a piece of paper.

The results on the motor-side from the experiments with the three ILC algorithms are shown in Figure 15.1. The measures in Figure 15.1 are calculated as in (12.6) but with the error e_k replaced with y_k.

Clearly, the transient response of the learning is better with the adaptive ILC scheme. For all the three controlled motors it is, in fact, true that the best performance is achieved with the ILC algorithm designed according to the adaptive ILC scheme but with the Kalman gain kept constant. This scheme is, however, not so robust which is shown by the fact that $\|\boldsymbol{y}_k\|_2$ for motor 1 actually starts growing after 6-7 iterations. In Figure 15.2 the value of the gain, κ_k, in the adaptive ILC algorithms are shown as a function of iteration. They are large in the first iterations where $d_k(t)$ has not been compensated completely but as $d_k(t)$ vanishes and the error decreases, the gains also decreases. Note that it is important to choose the correct size of $\hat{r}_{\Delta_d}$ in order to get this effect, cf. Algorithm 14.3. If $\hat{r}_{\Delta_d}$ is chosen too large this value will dominate $\hat{r}_{\Delta,k}$ and the κ_k will not decrease as shown in Figure 15.2, instead it will decrease like $\frac{1}{k+1}$.

It is also important to consider the actual result on the arm-side, i.e., the actual trajectory that the robot does with the tool. The desired trajectory is shown in Figure 8.11 and in Figure 15.3 the actual result is shown for the adaptive ILC scheme and the ILC scheme with constant Kalman gain. From the result in Figure 15.3 it is obvious that introducing ILC does not imply any improvement in the performance, evaluated on the arm-side. It is important to note that the ILC algorithm does not use the error on the arm-side when updating the control signal. The result in Figure 15.3 indicates that measurements from the arm-side have to be introduced in the learning algorithm in order to get a good behavior also on the motion of the tool. Note that the error in the path on the arm-side is also very different although the actual norm of y_k is similar. The introduction of additional sensors for improved ILC on the arm-side of the robot is discussed in Chapter 9.



(a) Motor 1. *(b)* Motor 2. *(c)* Motor 3.

Figure 15.1 *The normalized ∞-norm and 2-norm of the error for the different ILC algorithms. The adaptive ILC scheme ($\times$), the adaptive scheme with κ_k constant ($\circ$), and the traditional ILC scheme given by* (15.2) *($\diamond$).*

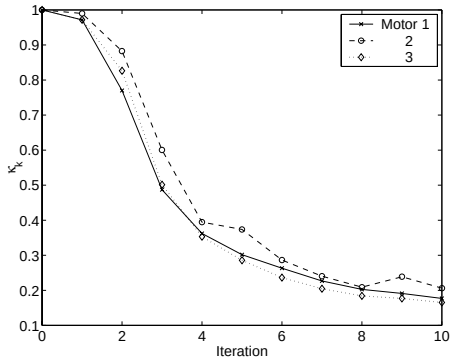
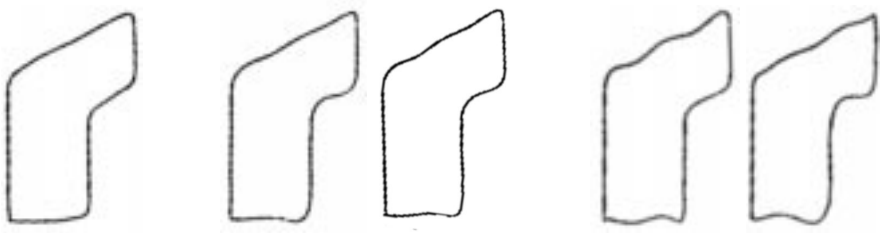


Figure 15.2 *The value of κ_k for the ILC associated with the three different motors.*



(a) Trajectory without ILC.

(b) Adaptive ILC scheme, iteration 5 (left) and iteration 10 (right).

(c) Design according to the adaptive ILC scheme but with constant Kalman gain, iteration 5 (left) and iteration 10 (right).

Figure 15.3 *Results on arm-side.*

16

Conclusions of Part V

Using the disturbance rejection formulation of ILC some new ILC algorithms have been developed. The basic idea in all the presented algorithms is to introduce an iteration varying gain in the ILC procedure.

Especially when taking the measurement disturbance into account it becomes obvious that it is possible to get a better result by introducing an iteration varying gain in the L-filter. This was also shown in Section 4.3.

In Chapter 14, results from state space modeling and design are used to create an ILC algorithm. This ILC algorithm has many properties in common with the estimation based approach from Section 3.5 and Section 4.3, but works also when the system is not perfectly known. The algorithm is based on an LQ-solution and a time variable Kalman filter where one of the design variables in the Kalman filter is calculated from data. This means that the algorithm is, in fact, adaptive. Under some assumptions on the model and the true system, conditions for stability for resulting ILC system are found. Also some results on robustness are presented.

Chapter 15 describes the experiments with the proposed adaptive algorithm applied to an industrial robot. The results show an improvement in the tracking on the motor-side of the robot and the proposed adaptive and model based ILC algorithm is shown to give better result than a traditional ILC algorithm with constant gain. When considering the result on the arm-side of the robot it is however shown that

the best result is achieved when no ILC algorithm is applied. This might seem to be a very negative result but, since measurements from the arm-side are not used in the algorithm it is not so surprising. Instead it shows the importance of introducing also measurements on the arm-side of the robot in order to get a good performance in the path following (considered on the arm-side). This has also been discussed previously in Chapter 9.

Part VI

Final Remarks

17

Conclusions and Future Work

This chapter gives a summary of the conclusions in the thesis and it also points out some open questions for future work.

17.1 Summary and Conclusions

The basis for the stability analysis, presented in the thesis, is the theory for *linear iterative systems*. Clearly, when a system using ILC is transformed into the linear iterative systems framework the analysis can be made by just applying standard tools from discrete time linear systems theory. This is an important observation since the stability analysis for linear systems is very well developed. In the design of well behaved ILC systems it is also clear from the results in the thesis that a frequency domain interpretation can be useful. The frequency domain interpretation is also applied to the analysis of systems using second order ILC algorithms. The conclusion of this work is that, in the case that we have considered, applying a second order ILC algorithm does not give a better result than a first order algorithm. As pointed out in Chapter 13 there might be cases where the second order algorithm is more compatible, e.g., when the system where it is applied is iteration variant.

In the introduction we stated that one of the aims of the thesis was to combine theory and practice. A result of this is that five different design schemes for ILC algorithms have been tested on a commercial industrial robot. The proposed ILC schemes stretches from a very simple structure, as in Algorithm 8.1, to the LQ and Kalman filter based adaptive approach in Part V.

A natural question is obviously: Which is the best ILC algorithm in the thesis? In the conclusions of Chapter 8 we have tried to make it clear that this depends on the demands that the user has on the resulting performance. If the performance that can be achieved using the heuristic design in Algorithm 8.1 is satisfactory, then this is clearly the approach that should be used. Second choice could be the optimization based approach in Algorithm 8.4 or Algorithm 8.5. If the designer is familiar with LQ design then this approach is straightforward to apply. If this is not the case it could be worth considering the model based approach in Algorithm 8.2 before trying the optimization based approach. If the measurements from the system are perturbated by a measurement disturbance, then it is possible to significantly improve the performance by introducing the adaptive ILC algorithms presented in Part V. The method presented in Section 3.5 and Section 4.3 can also be applied, although the performance can be very poor when the true system description is not known.

17.2 Future Work

One important question that has not been answered completely in the thesis is: What happens when ILC is applied to a non-minimum phase system? In Chapter 4 and in Chapter 8 some small notes have been made on the topic. The key point is that the gain of the inverse system becomes very large when the system is non-minimum phase. This is clear since the inverse is unstable. Although, in theory, an ILC algorithm that fulfills the stability conditions will find the optimal input that gives zero error it is not clear what happens if there are disturbances acting on the system. A large gain will also result in a large optimal control input, $\boldsymbol{u}_d$ in (4.21). This can lead to saturation of the control signal and, in turn, bad performance.

ILC applied to non-linear systems has not been discussed in the thesis and this is probably an area where many applications can be found. In the industrial robot case the system contains non-linearities in the form of, for example, friction, backlash, and saturations. The theory for ILC systems should be extended also to this class of systems.

In the industrial robot applications there are still many unsolved problems. One of the most important is clearly that the result on the arm-side of the robot might get worse by applying ILC compared to not applying the method at all, cf. Chapter 9. To be able to successfully apply ILC in a robotics application it will be necessary to use additional sensors, e.g., accelerometers, or build extremely accurate models of the mechanical structure of the robot. It is clearly possible to use other systems for

measuring the position of the tool, e.g., laser-based positioning systems. This has already been implemented in a commercial system by ABB Robotics and it is used with great success in laser-cutting applications. Future work could therefore focus on applying ILC using cheap sensors, such as accelerometers, and sensor-fusion. This means to implement on the robot the ideas presented in Chapter 9.

The testbed used in the experiments needs some updates in order to work completely satisfactory. The main improvements are pointed out in Chapter 7.

Bibliography

RAPID Reference Manual. ABB Flexible Automation, RobotWare 2.0 edition, 1997.

N. Amann and D. H. Owens. Non-minimum phase plants in norm-optimal iterative learning control. Technical Report 94/14, Centre for Systems and Control Engineering, University of Exeter, 1994.

N. Amann, D. H. Owens, and E. Rogers. 2d systems theory applied to learning control systems. In *Proc. of the 33rd IEEE Conf. on Decision and Control*, Lake Buena Vista, FL, USA, Dec 1994.

N. Amann, D. H. Owens, and E. Rogers. Iterative learning control for discrete time systems with exponential rate of convergence. Technical Report 95/14, Centre for Systems and Control Engineering, University of Exeter, 1995a.

N. Amann, D. H. Owens, and E. Rogers. Iterative learning control using optimal feedback and feedforward actions. Technical Report 95/13, Centre for Systems and Control Engineering, University of Exeter, 1995b.

N. Amann, D. H. Owens, and E. Rogers. Iterative learning control using prediction with arbitrary horizon. In *Proceedings of the European Control Conference 1997*, Brussels, Belgium, July 1997.

Brian D.O. Anderson and John B. Moore. *Optimal Filtering*. Prentice-Hall, 1979.

S. Arimoto. Mathematical theory of learning with applications to robot control. In K.S. Narendra, editor, *Adaptive and Learning Systems: Theory and Applications*, pages 379–388. Yale University, New Haven, Connecticut, USA, Nov 1985.

S. Arimoto. Passivity of robot dynamics implies capability of motor program learning. In *Proceedings of the International Workshop on Nonlinear and Adaptive Control : Issues in Robotics*, pages 49–68, Grenoble, France, 1991.

S. Arimoto, S. Kawamura, and F. Miyazaki. Bettering operation of robots by learning. *Journal of Robotic Systems*, 1(2):123–140, 1984a.

S. Arimoto, S. Kawamura, and F. Miyazaki. Iterative learning control for robot systems. In *Proceedings of IECON*, Tokyo, Japan, Oct 1984b.

S. Arimoto, S. Kawamura, F. Miyazaki, and S. Tamakie. Learning control theory for dynamic systems. In *Proc. of the 24th IEEE Conf. on Decision and Control*, pages 1375–1380, Ft. Lauderdale, Florida, Dec 1985.

G.J. Balas, J.C. Doyle, K. Glover, A. Packard, and R. Smith. *μ-Analysis and Syntheis Toolbox - For Use with Matlab*. MUSYN Inc. and The MathWorks Inc., 1994.

Z. Bien and K.M. Huh. Higher-order iterative learning control algorithm. In *IEE Proceedings*, volume 136, pages 105 – 112, May 1989.

Z. Bien and J.-X. Xu. *Iterative Learning Control: Analysis, Design, Integration and Application*. Kluwer Academic Publishers, 1998.

P. Bondi, G. Casalino, and L. Gambardella. On the iterative learning control theory for robotic manipulators. *IEEE Journal of Robotics and Automation*, 4: 14–22, Feb 1988.

E. Burdet, L. Rey, and A. Codourey. A trivial method of learning control. In *Preprints of the 5th IFAC symposium on robot control*, volume 2, Nantes, France, Sep 1997.

Ulf Carlsson. Model based high precision control of an industrial robot for laser-cuting. Master's thesis, Stockholm Institute of Technology, 2000. MMK 2000:37 MDA:140.

G. Casalino and G. Bartolini. A learning procedure for the control of movements of robotic manipulators. In *IASTED Symposium on Robotics and Automation*, pages 108–111, 1984.

Y. Chen, Z. Gong, and C. Wen. Analysis of a high order iterative learning control algorithm for uncertain nonlinear systems. *Automatica*, 34(3):345–353, March 1998.

Y. Chen and C. Wen. *Iterative Learning Control: Convergence, Robustness and Applications*, volume 248 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag, 1999.

Y. Chen, C. Wen, and H. Dou. High-order iterative learning control of functional neuromuscular stimulation systems. In *Proc. of the 36th IEEE Conf. on Decision and Control*, Hyatt Regency San Diego, California, Dec 1997a.

Y. Chen, J.-X. Xu, and T. H. Lee. Current iteration tracking error assisted iterative learning control of uncertain nonlinear discrete-time systems. In *Proc. of the 35th IEEE Conf. on Decision and Control*, pages 3040–5, Kobe, Japan, Dec 1996a.

Y. Chen, J.-X. Xu, and T. H. Lee. An iterative learning controller using current iteration tracking error information and initial state learning. In *Proc. of the 35th IEEE Conf. on Decision and Control*, pages 3064–9, Kobe, Japan, Dec 1996b.

Y. Chen, J.-X. Xu, and C. Wen. A high-order terminal iterative learning control scheme. In *Proc. of the 36th IEEE Conf. on Decision and Control*, Hyatt Regency San Diego, California, Dec 1997b.

J. J. Craig. *Adaptive Control of Mechanical Manipulators*. Addison-Wesley Publishing Company, 1988.

J.J. Craig. Adaptive control of manipulators through repeated trials. In *Proc. of ACC*, San Diego, CA, June 1984.

D. de Roover. Synthesis of a robust iterative learning controller using an H_∞ approach. *Selected Topics in Identification, Modelling and Control*, 9:89–96, Dec 1996a.

D. de Roover. Synthesis of a robust iterative learning controller using an H_∞ approach. In *Proceedings of 35th Conference on Decision and Control*, pages 3044–3049, Kobe, Japan, 1996b.

K. Dobrovodsky. Quaternion position representation in robot kinematic structures. In *Proceedings of the International Conference on Control 1994*, pages 561–564, 1994.

J. Funda and R.P. Paul. A comparison of transforms and quaternions in robotics. In *Proceedings of the 1988 IEEE Conference on Robotics and Automation*, volume 2, pages 886–891, 1988.

J. Funda, R.H. Taylor, and R.P. Paul. On homogeneous transforms, quaternions, and computational efficiency. *IEEE Transactions on Robotics adn Automation*, 6 (3):382–388, 1990.

K. Galkowski, E. Rogers, and D. H. Owens. 1d representation and systems theory for a class of 2d linear systems. In *Proceedings of the European Control Conference 1997*, Brussels, Belgium, July 1997.

Murray Garden. Learning control of actuators in control systems. US Patent, US03555252, Jan 1971. Leeds & Northrup Company, Philadelphia, USA.

D.M. Gorinevsky, D. Torfs, and A.A. Goldenberg. Learning approximation of feedforward dependence on the task parameters: Experiments in direct-drive manipulator tracking. In *Proc. ACC*, pages 883–887, Seattle, Washington, 1995.

U. Grenander and G. Szegö. *Toeplitz forms and their applications*. Chelsea publishing company, New York, second edition, 1984.

K. Guglielmo and N. Sadegh. Theory and implementation of a repetetive robot controller with cartesian trajectory description. *Journal of Dynamic Systems, Measurement, and Control*, 118:15–21, March 1996.

S. Gunnarsson and M. Norrlöf. On the use of learning control for improved performance in robot control systems. In *Proceedings of the European Control Conference 1997*, Brussels, Belgium, July 1997a.

S. Gunnarsson and M. Norrlöf. A short introduction to iterative learning control. Technical Report LiTH-ISY-R-1926, Department of Electrical Engineering, Linköping University, Feb 1997b.

S. Gunnarsson and M. Norrlöf. Some experiences of the use of iterative learning control for performance improvement in robot control systems. In *Preprints of the 5th IFAC symposium on robot control*, volume 2, Nantes, France, Sep 1997c.

S. Gunnarsson and M. Norrlöf. On the design of ILC algorithms using optimization. Technical Report LiTH-ISY-R-2209, Department of Electrical Engineering, Linköping University, Dec 1999a.

S. Gunnarsson and M. Norrlöf. Some aspects of an optimization approach to iterative learning control. In *Proc. of the 38th IEEE Conference on Decision and Control*, Pheonix, Arizona, USA, Dec 1999b.

S. Gunnarsson and M. Norrlöf. terative learning control of a flexible mechanical system using accelerometers. In *Preprints of the 6th IFAC symposium on robot control*, Vienna, Austria, Sep 2000.

S. Gunnarsson, M. Norrlöf, G. Hovland, U. Carlsson, T. Brogårdh, T. Svensson, and S. Moberg. Method for high accuracy performance of an industrial robot. Swedish Patent Application No. 0001312-8, April 2000.

S. Gunnarsson, O. Rousseaux, and V. Collignon. Iterative feedback tuning applied to robot joint controllers. In *Proceedings of the 14th IFAC Triennial World Congress*, volume I, Aug 1999.

S. Hara, Y. Yamamoto, T. Omata, and M. Nakano. Repetitive control system: A new type servo system for periodic exogenous signals. *IEEE Transactions on Automatic Control*, 33(7):659–668, 1988.

L.M. Hideg. *Stability of Learning Control Systems*. PhD thesis, Oakland University, 1992.

H. Hjalmarsson, S. Gunnarsson, and M. Gevers. A convergent iterative restricted complexity control design scheme. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pages 1735–1740, Orlando, Florida, USA, 1994.

R. Horowitz. Learning control of robot manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 115:402–411, June 1993.

R. Horowitz. Learning control applications to mechatronics. *JSME International Journal*, 37(3):421–430, 1994. Series C.

R. Horowitz, W. Messner, and J. B. Moore. Exponential convergence of a learning controller for robot manipulators. *IEEE Transactions on Automatic Control*, 36 (7):890–894, July 1991.

JR. J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1983.

Ping Jiang, Hui tang Chen, and Yue juan Wang. Iterative learning control for glass cutting robot. In *Proceedings of the 14th IFAC Triennial World Congress*, pages 263–268, Bejing, P. R. China, 1999.

T. Kailath. *Linear Systems*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1980.

F. Lange and G. Hirzinger. Learning accurate path control of industrial robots with joint elasticity. In *Proc. IEEE Conference on Robotics and Automation*, pages 2084–2089, Detriot, MI, USA, 1999.

Jay. H. Lee, Kwang S. Lee, and Won C. Kim. Model-based iterative learning control with a quadratic criterion for time-varying linear systems. *Automatica*, 36 (5):641–657, May 2000.

K.S. Lee and J.H. Lee. *Design of Quadratic Criterion-Based Iterative Learning Control. In Iterative Learning Control: Analysis, Design, Integration and Applications. Z. Bien and J.X. Xu., eds.* Kluwer Academic Publishers, 1998.

Y.-J. Liang and D. P. Looze. Performance and robustness issues in iterative learning control. In *Proc. of the 32nd Conf. on Decision and Control*, pages 1990–1995, San Antonio, Texas, USA, Dec 1993.

L. Ljung. *System Identification: Theory for the User*. Prentice-Hall, 1987.

L. Ljung. *System Identification Toolbox - For Use with Matlab*. The MathWorks Inc., 1995.

R.W. Longman. Iterative learning control and repetitive control for engineering practice. *International Journal of Control*, 73(10):930 – 954, July 2000.

T. Manabe and F. Miyazaki. Learning control based on local linearization by using dft. *Journal of Robotic Systems*, 11(2):129–141, 1994.

T. Mita and E. Kato. Iterative control and its application to motion control of robot arm - a direct approach to servo-problems -. In *Proc. of 24th Conf. on Decision and Control*, pages 1393–1398, Ft. Lauderdale, USA, Dec 1985.

F. Miyazaki, S. Kawamura, M. Matsumori, and S. Arimoto. Learning control scheme for a class of robots with elasticity. In *Proc. of the 25th IEEE Conference on Decision and Control*, pages 74–79, Athens, Greece, 1986.

K. L. Moore. *Iterative Learning Control for Deterministic Systems*. Advances in Industrial Control. Springer-Verlag, 1993.

K. L. Moore. Iterative learning control - an expository overview. *Applied and Computational Controls, Signal Processing and Circuits*, 1998a.

Kevin L. Moore. Multi-loop control approach to designing iterative learning controllers. In *Proc. of the 37th IEEE Conference on Decision and Control*, Tampa, Florida, USA, Dec 1998b.

K. Nilsson. *Industrial Robot Programming*. PhD thesis, Department of Automatic Control, Lund Institute of Technology, May 1996.

M. Norrlöf. On analysis and implementation of iterative learning contol. Licentiate thesis LIU-TEK-LIC-1998:62 Linköping Studies in Science and Technology. Licentiate Thesis No 727, Department of Electrical Engineering, Linköpings universitet, Oct 1998.

M. Norrlöf. Modeling of industrial robots. Technical Report LiTH-ISY-R-2208, Department of Electrical Engineering, Linköping University, Dec 1999.

M. Norrlöf. Adaptive iterative learning control algorithms for disturbance rejection. Technical Report LiTH-ISY-R-2244, Department of Electrical Engineering, Linköping University, May 2000a.

M. Norrlöf. Analysis of a second order iterative learning controller. Technical Report LiTH-ISY-R-2181, Department of Electrical Engineering, Linköping University, Feb 2000b.

M. Norrlöf. Comparative study on first and second order ILC – frequency domain analysis and experiments. In *Proc. of the 39th IEEE Conference on Decision and Control*, Sydney, Australia, Dec 2000c.

M. Norrlöf and S. Gunnarsson. Some results on iterative learning control with disturbances. In *Proceeding First Conference on Computer Science and Systems Engineering in Linköping*, pages 193–202. ECSEL, March 1998.

M. Norrlöf and S. Gunnarsson. A frequency domain analysis of a second order iterative learning control algorithm. In *Proc. of the 38th IEEE Conference on Decision and Control*, Pheonix, Arizona, USA, Dec 1999.

M. Norrlöf and S. Gunnarsson. Disturbance aspects of iterative learning control. Technical Report LiTH-ISY-R-2261, Department of Electrical Engineering, Linköping University, May 2000a.

M. Norrlöf and S. Gunnarsson. A model based iterative learning control method applied to 3 axes of a commercial industrial robot. In *Preprints of the 6th IFAC symposium on robot control*, Vienna, Austria, Sep 2000b.

D. H. Owens and G. S. Munde. Stability of a multi-input, multi-output adaptive iterative learning control system. In *Proceedings of the European Control Conference 1997*, Brussels, Belgium, July 1997.

S. Panzieri and G. Ulivi. Disturbance rejection of iterative learning control applied to trajectory tracking for a flexible manipulator. In *Proceedings of 3rd European Control Conference*, pages 2374–2379, Sep 1995.

J. S. Park and T. Hesketh. Model reference learning control for robot manipulators. In *Proceedings of the IFAC 12th Triennial World Congress*, pages 341–344, Sydney, Australia, 1993.

J.A Frueh M.Q. Phan. Linear quadratic optimal learning control (LQL). In *Proceedings of the 37th IEEE Conference on Decision and Control*, pages 678–683, Tampa, Florida, USA, 1998.

J. B. Plant. *Some Iterative Solutions in Optimal Control*. Number 44 in Research monograph. The M.I.T. Press, Cambridge, Massachusetts, 1968.

M. Poloni and G. Ulivi. Iterative learning control of a one-link flexible manipulator. In *IFAC Robot Control*, pages 393–398, Vienna, Austria, 1991.

E. Rogers and D. H Owens. *Stability Analysis for Linear Repetitive Processes*, volume 175 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag, 1992.

E. Rogers and D. H. Owens. 2d systems theory and applications - a maturing area. In *Proceedings of the UKACC International Conference on Control 1994*, March 1994.

W. J. Rugh. *Linear system theory*. Prentice Hall, second edition, 1996.

K. Shoemake. Animating rotation with quaternion curves. In *Proceedings of the ACM SIGGRAPH'85*, pages 245–254, San Francisco, USA, 1985.

J.-J. E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice Hall, Englewood Cliffs, New Jersey 07632, 1991.

M. W. Spong, F. L. Lewis, and C. T. Abdallah, editors. *Robot Control: Dynamics, Motion Planning, and Analysis*, 1992. IEEE Control Systems Society, IEEE Press.

M. W. Spong and M. Vidyasagar. *Robot Dynamics and Control*. Wiley, 1989.

W. Stadler. *Analytical robotics and mechatronics*. McGraw-Hill International editions, 1995.

W. Stallings. *Operating Systems*. Maxwell MacMillan International, 1992.

K. J. Åström and B. Wittenmark. *Computer Controlled Systems: Theory and Design*. Prentice-Hall, 1984.

K.J. Åström and B. Wittenmark. *Adaptive Control*. Addison-Wesley Publishing Company, Inc., second edition, 1995.

M. Togai and O. Yamano. Analysis and design of an optimal learning control scheme for industrial robots: a discrete system approach. In *Proc. of the 24th IEEE Conf. on Decision and Control*, pages 1399–1404, Ft. Lauderdale, FL., Dec 1985.

M. Tomizuka, T. C. Tsao, and K. K. Chew. Discrete-time domain analysis and sythesis of repetetive controllers. *Journal of Dynamic Systems, Measurement, and Control*, 111:353–358, 1989.

M. Uchiyama. Formulation of high-speed motion pattern of a mechanical arm by trial. *Trans. SICE (Soc. Instrum. Contr. Eng.)*, 14(6):706–712, 1978. Published in Japanese.

W.J.R. Velthuis, T.J.A. de Vries, and J. van Amerongen. Learning feed forward control of a flexible beam. In *Proceedings of the 1996 IEEE International Symposium on Intelligent Control*, pages 103–108, Dearborn, MI, Sep 1996.

K. Zhou, J.C. Doyle, and K. Glover. *Robust and Optimal Control*. Prentice Hall, Upper Saddle River, New Jersey 07458, 1996.

Index

PhD Dissertations, Division of Automatic Control, Linköping University

M. Millnert: Identification and control of systems subject to abrupt changes. Thesis no. 82, 1982. ISBN 91-7372-542-0.

A.J.M. van Overbeek: On-line structure selection for the identification of multivariable systems. Thesis no. 86, 1982. ISBN 91-7372-586-2.

B. Bengtsson: On some control problems for queues. Thesis no. 87, 1982. ISBN 91-7372-593-5.

S. Ljung: Fast algorithms for integral equations and least squares identification problems. Thesis no. 93, 1983. ISBN 91-7372-641-9.

H. Jonson: A Newton method for solving non-linear optimal control problems with general constraints. Thesis no. 104, 1983. ISBN 91-7372-718-0.

E. Trulsson: Adaptive control based on explicit criterion minimization. Thesis no. 106, 1983. ISBN 91-7372-728-8.

K. Nordström: Uncertainty, robustness and sensitivity reduction in the design of single input control systems. Thesis no. 162, 1987. ISBN 91-7870-170-8.

B. Wahlberg: On the identification and approximation of linear systems. Thesis no. 163, 1987. ISBN 91-7870-175-9.

S. Gunnarsson: Frequency domain aspects of modeling and control in adaptive systems. Thesis no. 194, 1988. ISBN 91-7870-380-8.

A. Isaksson: On system identification in one and two dimensions with signal processing applications. Thesis no. 196, 1988. ISBN 91-7870-383-2.

M. Viberg: Subspace fitting concepts in sensor array processing. Thesis no. 217, 1989. ISBN 91-7870-529-0.

K. Forsman: Constructive commutative algebra in nonlinear control theory. Thesis no. 261, 1991. ISBN 91-7870-827-3.

F. Gustafsson: Estimation of discrete parameters in linear systems. Thesis no. 271, 1992. ISBN 91-7870-876-1.

P. Nagy: Tools for knowledge-based signal processing with applications to system identification. Thesis no. 280, 1992. ISBN 91-7870-962-8.

T. Svensson: Mathematical tools and software for analysis and design of nonlinear control systems. Thesis no. 285, 1992. ISBN 91-7870-989-X.

S. Andersson: On dimension reduction in sensor array signal processing. Thesis no. 290, 1992. ISBN 91-7871-015-4.

H. Hjalmarsson: Aspects on incomplete modeling in system identification. Thesis no. 298, 1993. ISBN 91-7871-070-7.

I. Klein: Automatic synthesis of sequential control schemes. Thesis no. 305, 1993. ISBN 91-7871-090-1.

J.-E. Strömberg: A mode switching modelling philosophy. Thesis no. 353, 1994. ISBN 91-7871-430-3.

K. Wang Chen: Transformation and symbolic calculations in filtering and control. Thesis no. 361, 1994. ISBN 91-7871-467-2.

T. McKelvey: Identification of state-space models from time and frequency data. Thesis no. 380, 1995. ISBN 91-7871-531-8.

J. Sjöberg: Non-linear system identification with neural networks. Thesis no. 381, 1995. ISBN 91-7871-534-2.

R. Germundsson: Symbolic systems – theory, computation and applications. Thesis no. 389, 1995. ISBN 91-7871-578-4.

P. Pucar: Modeling and segmentation using multiple models. Thesis no. 405, 1995. ISBN 91-7871-627-6.

H. Fortell: Algebraic approaches to normal forms and zero dynamics. Thesis no. 407, 1995. ISBN 91-7871-629-2.

A. Helmersson: Methods for robust gain scheduling. Thesis no. 406, 1995. ISBN 91-7871-628-4.

P. Lindskog: Methods, algorithms and tools for system identification based on prior knowledge. Thesis no. 436, 1996. ISBN 91-7871-424-8.

J. Gunnarsson: Symbolic methods and tools for discrete event dynamic systems. Thesis no. 477, 1997. ISBN 91-7871-917-8.

M. Jirstrand: Constructive methods for inequality constraints in control. Thesis no. 527, 1998. ISBN 91-7219-187-2.

U. Forssell: Closed-loop identification: methods, theory, and applications. Thesis no. 566, 1999. ISBN 91-7219-432-4.

A. Stenman: Models on demand: algorithms, analysis and applications. Thesis no. 571, 1999. ISBN 91-7219-450-2.

N. Bergman: Recursive Bayesian estimation: navigation and tracking applications. Thesis no. 579, 1999. ISBN 91-7219-473-1.

K. Edström: Switched bond graphs: simulation and analysis. Thesis no. 586, 1999. ISBN 91-7219-493-6.

M. Larsson: Behavioral and structural model based approaches to discrete diagnosis. Thesis no. 608, 1999. ISBN 91-7219-615-5.

F. Gunnarsson: Power control in cellular radio systems: analysis, design and analysis. Thesis no. 623, 2000. ISBN 91-7219-689-0.

V. Einarsson: Model checking methods for mode switching systems. Thesis no. 652, 2000. ISBN 91-7219-836-2.