Linköping Studies in Science and Technology
Thesis No. 727

On Analysis and Implementation of Iterative Learning Control

Mikael Norrlöf

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

Linköping 1998

On Analysis and Implementation of Iterative Learning Control

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

ISBN 91-7219-309-3
ISSN 0280-7971
LiU-TEK-LIC-1998:62

Printed by UniTryck, Linköping, Sweden 1998

To my Mother and my Father

Abstract

Many of the control systems used in factory production today are programmed to perform the same task repeatedly. In particular this is the case for industrial robots where the same motion is performed every time the same program is executed. An interesting observation, for the industrial robot, is that the error in the different iterations of the same exercise is highly repetitive.

In the thesis *Iterative Learning Control* is applied to an industrial robot control system from ABB. Using Iterative Learning Control the tracking error on the motor side has been reduced without changing the internal structure or any parameters in the robot controller. The results from the experiments show that Iterative Learning Control can be used to successfully reduce the tracking error in an industrial robot control system.

The implementation of the functions needed in the robot controller is described. By using a combination of already present functions in the system and software development the Iterative Learning Control method has been successfully applied to the commercial robot controller, S4C.

The Iterative Learning Control method uses knowledge from previous exercises to improve the control in future executions of the same exercise. For the robot control case, this means remembering the error that was achieved in the previous iteration of the exercise and to change the input signal to the system based on this knowledge.

A theory including analysis and synthesis for Iterative Learning Control is provided, and a starting point for a more general theory on iterative systems is given. A general discussion on how the Iterative Learning Control method can deal with repetitive and random disturbances is also included.

Two of the given design algorithms are evaluated by experiments. The experiments show that the methods apply very well to the control of the ABB robot. The error reaches steady state levels, in fact the quantization level, already after 3 iterations with a model based synthesis approach.

Acknowledgments

Climbing the stairs of life can sometimes be difficult and, for sure, some steps are bigger than others. To reach this step, now when the thesis is written, one of the most difficult steps in my life had to be conquered. This doesn't mean, of course, that is wasn't fun. In fact it was GREAT fun! Life has also taught me a lesson to remember: "Don't relax. It's not over until the fat lady sings..."

First of all I would like to thank my supervisor, Dr. Svante Gunnarsson, and my industrial supervisor Dr. Torgny Brogårdh. This thesis had never been possible without all our fruitful discussions and all the valuable comments that I have received from you. Torgny read and gave me feedback on an early and a late version of the manuscript, these comments have improved the final version considerably.

I am also grateful to Prof. Lennart Ljung and Prof. Torkel Glad. The research environment that you have created in the group is unique. I am happy and proud to be a member of your staff.

All the guys in the Automatic Control group are gratefully acknowledged. I especially thank Ulla, our secretary, for the administrative help and for always thinking beyond everyone else in the group[1]. Ulla, you are invaluable.

A number of people have contributed to the work in different ways. Valur Einarsson read the manuscript and gave me many valuable comments and suggestions. Valur and Fredrik Tjärnström have also answered a lot of my questions and taken their time to listen to, and discuss, topics of my research. Thank you! Måns Östring has provided me with the controller in Chapter 7 and, also, with a lot of knowledge on the control of systems having flexibilities (among other things). Thank you Måns (I owe you one). Sören Hansson is gratefully acknowledged for helping me with all the practical things in the lab. I also would like to thank Prof. Kevin Moore for sending me a preview of his survey article [Moo98].

The people at ABB Robotics have provided me with a new "family" in Västerås. I am grateful to all of you. I send special thanks to Laila and Jappe for your great hospitality. I would also like to thank, Staffan, Håkan, Fredrik, Bosse, Peter E, Steve, Geir, Daniel, Anders, Antonio, Stig, and all the rest of the people at ABB that have helped me during the work on this thesis. It would never have been possible without your help.

[1] At least when it comes to practical things.

This work was supported by ABB Robotics Products AB within NUTEK's Center of Excellence ISIS[2], which is gratefully acknowledged.

I also would like to thank my family, my Mother and my Father, my sister Lisbeth and her boyfriend Janne, and big-brother Mats. You have always taken cared of me and helped me to think of other things than work.

Finally, I would like to thank Anna for all the love, support, and encouragement she has given me during the writing of this thesis.

Ti amo! Il mio ❤ appartiene a te.

Linköping, October 1998

Mikael Norrlöf

[2]Information Systems for Industrial Control and Supervision

Contents

Notation

Symbols

$\mathbb{R}$, $\mathbb{C}$	The sets of real and complex numbers.
$r(t)$	Reference signal.
$u_k(t)$, $u_\infty(t)$	Control signal in iteration k, asymptotic value.
$y_k(t)$, $y_\infty(t)$	Output signal in iteration k, asymptotic value.
$e_k(t)$, $e_\infty(t)$	Error signal in iteration k, asymptotic value.
$e^*(t)$, $u^*(t)$	The equilibrium trajectories.
$\nu(t)$	General disturbance, includes $d(t)$ and $n(t)$.
$d(t)$, $n(t)$	Repetitive disturbance, random disturbance.
G_C	Closed loop system.
G_D	The system used in the disturbance rejection approach, see Section 2.3.3.
$Q(q)$, $L(q)$	Filters in the ILC updating formula, cf. (2.39).
T	General system, used in the tracking approach, see Section 2.3.2.
t_f	Final time, used to define the length of an exercise, see Definition 2.5.
t_s	Sample time.
$\Phi_u(\omega)$	Spectrum for the signal $u(t)$.

Φ, Φ_i	Joint coordinates ($\Phi \in \mathbb{R}^n$), joint coordinate for joint i.

Operators and Functions

∂, p	Derivation operator.
q, q_t	Delay operator in time direction.
q_k	Delay operator in iteration direction.
$\dot{x}, \ddot{x}, x^{(i)}$	First, second and i^{th} derivative of x w.r.t. t.
$\equiv$	Equivalent to.
$\Longrightarrow$	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.
$\|F(e^{i\omega t_s})\|_\infty$	Matrix H_∞-norm. Definition see (2.12).
$\|e_k(t)\|_\infty$	Vector H_∞-norm. Defined as $\sup_{t\in[0,t_f]} \|e_k(t)\|$.
$\|e_k(t)\|_2$	2-norm. Defined as $\|e_k(t)\|_2^2 = \int \|e_k(t)\|^2 dt$. In the discrete time case $\int$ is replaced by $\sum$.

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.
ILC	Iterative Learning Control.
ILCF	Iterative Learning Control Feedback.
I/O	Input/Output.
IRB	Industrial Robot.
ISIS	Information Systems for Industrial Control and Supervision.
LTI	Linear Time Invariant.
LTV	Linear Time Varying.

MIMO	Multiple Input Multiple Output.
MINO	Multiple Input No Output.
MOC	Motion Control.
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

The aim of this thesis is to investigate how the properties of an industrial control system can be improved by applying Iterative Learning Control (ILC). The ILC-method will be presented and the application of the method, industrial robot control, will be described. Experiments on an industrial robot from ABB will show some important aspects of the method. The actual implementation of the functions needed in the industrial robot controller is also described in the thesis.

1.1 Background

Control of systems can basically be divided into two different problems,

- *stabilization* and
- *performance.*

In this thesis the performance issue will be in focus, but of course the stability demands must always be satisfied. We will consider the problem of improving the tracking capability of an industrial system, where our knowledge of the implementation of the controller is limited. Of course, changing the structure of the controller would be a very interesting exercise but since the control systems in modern industrial systems are very complicated, this is not a realistic task.

The method that we have considered, called Iterative Learning Control, is an off-line method in the sense that the improvement that we do to the system can be calculated off-line. We will now give an example of a control problem to emphasize the ideas behind ILC and also to relate the approach to other control paradigms.

Consider the system depicted in Figure 1.1. We assume that the structure of the system inside the frame can not be changed. We can only observe some of the signals, depicted in the figure with dashed arrows, crossing the border of the frame. The signals are only available as complete sequences, $\{r_{G_C}(t)\}_0^{t_f}$ and $\{y(t)\}_0^{t_f}$. In the same way it is only possible to interact with the system using sequences of data, $\{u(t)\}_0^{t_f}$, where the sequence must be completely defined at time 0. Because of the fact that the sequences have to be defined in advance, it is not possible to use conventional feedback.

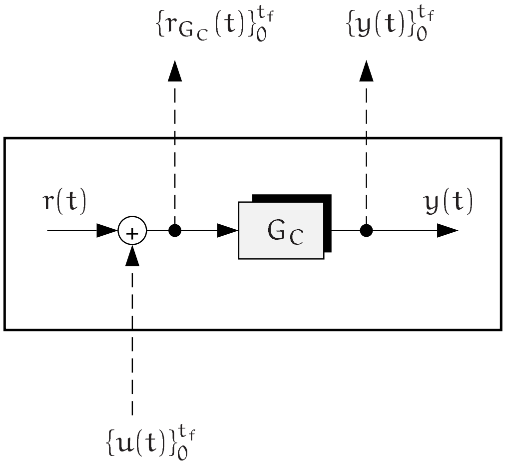
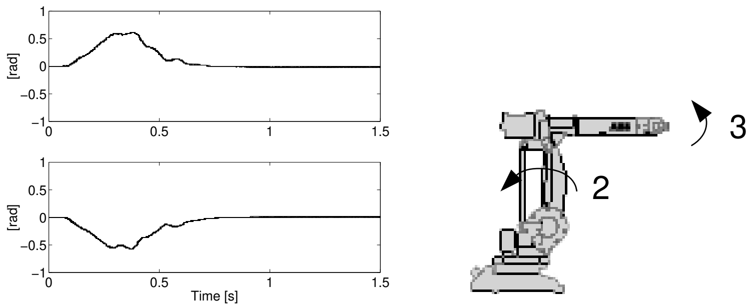


Figure 1.1 *The structure of the system that we consider in the thesis.*

If we now know that the system G_C performs the same exercise repeatedly, we can use this fact to improve the control of the system. For example, assume that the system G_C is a robot arm that we have programmed to iteratively perform the same exercise. Assume that the length of the exercise is t_f and that we start the exercise at time 0. We can now observe the behavior of the system during one of the exercises by using the two sets of output data that we get from the system. Twenty such observations from an exercise performed 20 times by an industrial robot[1] are depicted in Figure 1.2. We can see that the error is highly repetitive. Using the knowledge of the error from the previous exercises, the input, $\{u(t)\}_0^{t_f}$, can be updated to reduce the error next time the exercise is performed.



a. Position error $(r_{G_C}(t) - y(t))$, axis 2 (above), axis 3 (below).

b. Combined motion, axis 2 and 3.

Figure 1.2 *Twenty iterations on one exercise using a standard ABB IRB 1400 (without ILC). The size of the motion is 1 rad on the arm side for both axes.*

An alternative way of solving the problem would be to build a model of the system, G_C, and use this model for the control. This approach would perhaps not have the disadvantage of being local to the particular exercise, which is the case for ILC. If the model is valid for other reference signals than a particular realization of $r(t)$ the identification approach would clearly be to prefer. If, on the other hand, we have the assumption that the system inside the frame in Figure 1.1 can not be altered, the ILC approach becomes preferable. Using a combination of the two methods would, of course, also be possible.

1.2 Overview of the ILC Research Area

The first contribution to ILC was a paper by Uchiyama [Uch78] published in 1978. Since it was published in Japanese only, the ideas did not be-

[1]The robot is described in Chapter 5.

come widely spread until Arimoto et al. published a paper in the Journal of Robotic Systems 1984 [AKM84a]. This paper is usually referred to as the original paper of ILC, although the method was referred to as "a betterment process" in the paper. The name Iterative Learning Control was first introduced, also by Arimoto et al., in [AKM84b]. During the 1980's and the beginning of the 90's Arimoto's group have published a number of conference and journal papers, among these are, e.g., [AKMT85, KMA88, Ari91, AN92]. Arimoto has also contributed with chapters on ILC in some books, e.g., a book edited by Narendra [Ari85] and a book edited by Bien and Xu [Be98]. Independently of Arimoto's group the ideas of ILC were presented, also in 1984, by Casalino and Bartolini [CB84] and by Craig [Cra84].

The development of ILC stem from the robotics area where the repetitive motions are common in applications. Even though most of the results in ILC are for linear systems, many of the contributions have been for non-linear systems, e.g., [KMA88, Cra88, Hor93]. Many of the results on non-linear systems are, not surprisingly, on the special structure of non-linear systems that is found in robotics.

During the 90's a lot of research on ILC has been made by research groups in the south east Asia. For example, at the 2nd Asian Control Conference, held in Seoul 1997, more than thirty papers were on ILC. In the book edited by Bien and Xu [Be98] some of the contributions from the conference are presented. A good survey of ILC and its applications is found in the book by Moore [Moo93] and in a survey article by the same author [Moo98]. We refer to [Moo93, Moo98] and [Be98] for a thorough discussion of the contributions and applications of ILC.

1.3 Outline of the Thesis

The thesis can be divided into four main parts, not considering the conclusions.

In this introductory chapter the ILC method has been explained very briefly, we have also presented a brief overview of the research field of ILC. The motivation for this is to give some perspective to the work presented in the thesis and also to give the reader an overview of the evolution of ILC as a research field.

Chapters 2, 3, and 4, are devoted to definitions to build the framework that we use in the thesis, to present an analysis for the stability of systems using ILC feedback and, finally, to give some design algorithms for the ILC. We will in the synthesis only consider linear time invariant systems and the emphasis in the analysis will also be on linear systems, even though some results will be presented for non-linear systems.

In the third part of the thesis we will discuss the application that we have used in the evaluation of the method. Chapter 5 is an overview of robot modeling and control. The chapter also includes a general discussion on the ABB industrial robot family and the IRB1400, in particular. The IRB1400 is the robot that we have used in the experiments. In Chapter 6 the implementation and the changes that we have made to the ABB robot control system is described. A general overview of the MOC test environment is also given. MOC is the platform that we have used for the implementation. The actual implementation of the ILC updating algorithm for the control signal, cf. Figure 1.1, is presented as well. The ILC is implemented in MATLAB™ on a PC and the interface between the robot controller and the PC is also discussed in Chapter 6.

The last part of the thesis covers the experiments that we have made on the industrial robot control system. Chapter 7 contains the results from the experiments and also some connections to the second part of the thesis. A modeling of the closed loop system, G_C in Figure 1.1, is performed to make it possible to use one of the design algorithms from Chapter 4. The results from the experiments are very promising and we have reached the limit for what is possible with the current implementation of the robot controller. The limit in the control is actually the quantization level in the DSP on which the internal feedback loop of the robot control system is executed.

1.4 Contributions

The main contributions of this thesis are the following:

- The ILC stability definitions in Section 2.4. The idea is to generalize the stability theory for nonlinear systems to the ILC case.
- The *disturbance rejection approach* to the ILC problem, described in Sections 2.3.3 and 3.3.
- A general approach to the theory of linear iterative systems presented in Section 3.1.1.
- Analysis of stability and performance for ILC applied to systems with repetitive disturbances and random disturbances in Section 3.2.1.
- A heuristic approach to the design of the ILC, presented in Section 4.2.1.

- The implementation of the functions needed to incorporate the ILC method in the industrial control system, presented in Chapter 6.
- The experiments on an industrial robot showing important properties of the ILC method, presented in Chapter 7. The heuristic design algorithm for the ILC is also evaluated.

2

Problem Definition

This chapter has two purposes. First, to describe the ideas behind *Iterative Learning Control (ILC)*. Second, to introduce and explain the notation and the terminology used in the thesis. ILC will be introduced by an example, showing how it can be applied to a linear system.

2.1 Introduction

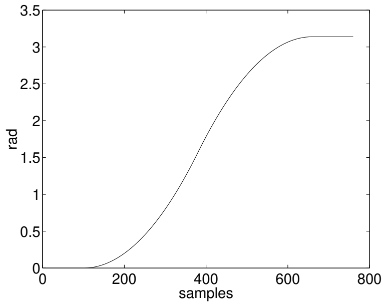
The abbreviation ILC stands for Iterative Learning Control, three words that give a good idea of what the method is all about. *Iterative*, because we repeat something over and over again. For example, the repeated execution of the same trajectory using an industrial robot. An iterative process will be formally defined later in this chapter. The name of the method also includes *learning* which refers to the idea that by repeating the same thing we should be able to perform better. This does not mean, of course, that it is learning in the human sense. Learning is here used only in a mathematical sense implying that the system performs better after some iterations. *Control* is included in the name to emphasize that the result of the learning is used to control the system. Usually the system already has a traditional controller and, hence, the ILC will add a new control loop to the system. The ILC will introduce new control signals based on learning from previous iterations and result in a reduced overall control error.

We will define the terminology of ILC later in this chapter but to show some of the basic properties of ILC, an example is used. The example shows how ILC can be applied to improve the tracking performance of a control system.

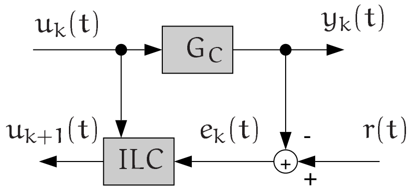
2.2 An Introductory Example of ILC

Assume that we have a certain reference signal $r(t)$ over a finite time interval $[0, t_f]$. We have a system that should track this reference trajectory repeatedly with a very high accuracy. A typical application where this problem arises is in the control of robot arms [AKM84a, Ari91, HMM91, Hor93, Cra88, BRC97]. Later in the thesis an example of ILC applied to the control of an industrial robot will be presented.

Consider the system and the reference signal in Figure 2.1. Assume that 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 we assume that every time the same motion is repeated the system starts from the original initial condition.



a. The reference signal



b. An ILC feedback system

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

Assume that we have an initial guess on the input to the system, denoted by $u_0(t)$. Feeding the system with the initial guess of the input gives,

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

where q represents the delay operator[1]. Let us define the tracking error,

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

as the difference between the desired output and the system output.

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

The fundamental idea behind ILC can now be introduced. Assume that

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

These assumptions lead to the conclusion that the tracking error will also be the same between the iterations. For simplicity we assume that there are no disturbances. The idea of ILC is now to utilize the assumptions and, by using a search method, find the optimal input to the system. Optimal in this particular case means to find the signal $u(t) = G_C^{-1}(q)r(t)$. If, of course, G_C is invertible. Let us use the following ILC update formula for the control input of the system,

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

where $L(q)$ is a linear discrete time filter. This updating formula is the most common in literature, used in e.g., [AKM84a, TY85, PH93]. Compare also the structure of the updating formula with the one used in numerical analysis methods [DB74]. We can now generalize (2.2), (2.1), and (2.3), according to

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

where k is the iteration index, mentioned before, and q is the delay operator.

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

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

[1] See also Definition 2.3.

We see that in order for the error not to grow, the expression

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

must be less than or equal to 1 for all $\omega \in [-\pi, \pi]$. The sampling time is assumed to be 1. This is the convergence, or stability criterion for this ILC updating formula[2].

Let us look at a numerical example showing the important convergence and stability properties of the method.

Example 2.1 (A Numerical Example) *Consider the following system*

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

Assume that the system starts from zero initial condition and that we want to track the reference signal given by Figure 2.2b. Let us do two simulations using the updating formula for the control signal in (2.4c) with two different $L(q)$,

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

and

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

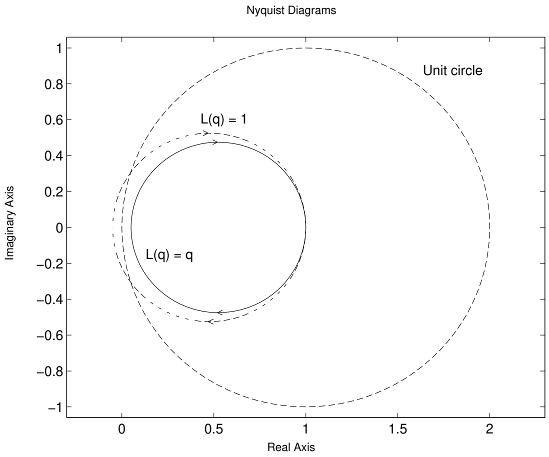
The convergence criterion is given by

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

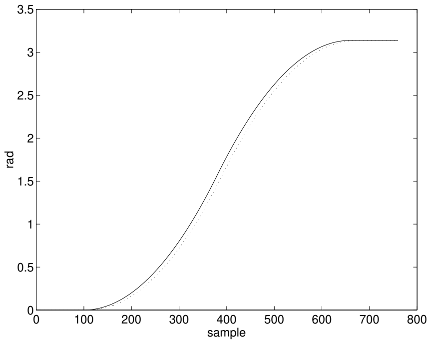
cf. (2.6). The expression can be interpreted in the following way; the Nyquist curve for $G_C(q)L(q)$ *must be contained in a region in the complex plane given by a unit circle centered at 1. In Figure 2.2a the Nyquist curves are shown for the two choices of* $L(q)$ *given by (2.8) and (2.9). We can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

$$\|1 - G_C(e^{i\omega})L(e^{i\omega})\|_\infty \tag{2.11}$$

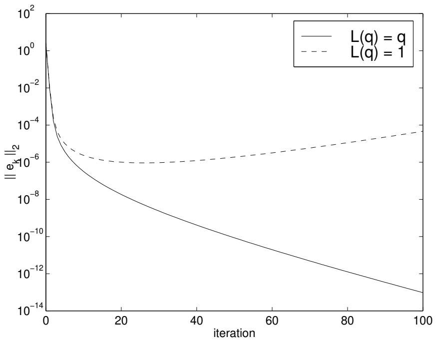
[2]The criterion will be extended to general linear updating formulas in Chapter 3.



a. A graphical interpretation of the convergence criteria.



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 (2.10) 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 *Result of ∞-norm calculation.*

The function $\|\cdot\|_\infty$ is defined as

$$\|F\|_\infty := \sup_{\omega \in [-\pi,\pi]} \bar{\sigma}[F(e^{i\omega}] \tag{2.12}$$

where $\bar{\sigma}$ symbolizes the maximum singular value of $F(e^{i\omega})$. An approximate measure of this norm can, e.g., be found in MATLAB™ using the command `dhfnorm`*. The result from this calculation is shown in Table 2.1. We see that using $L(q) = q$ will give a norm less than 1. Using $L(q) = 1$, however, will give a norm greater than 1. This is the same result that we found using the Nyquist diagram. Yet another way of examining the stability is of course simulations. In Figure 2.2b the result of feeding the system with the initial guess of the input 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 2.2c 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 $L = q$ gives a stable behavior but $L = 1$ results in a growing error after a first phase with rapidly decreasing error.*

□

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

Remark 2.1
In the unstable case the size of the error is first reduced before it starts to grow, according to Figure 2.2c. The reason for this behavior can be understood by considering (2.5) and the result of the evaluation of the criteria using the Nyquist diagram in Figure 2.2a. For low frequencies the error is reduced even in the unstable case but for high frequencies the error will be attenuated in the unstable case. This can be seen in the Nyquist diagram by the fact that the Nyquist curve is outside the unit circle for high frequencies.

Since, in the first iteration, the error is dominated by low frequency components, the energy of the error will be reduced. After some time, however, the high frequency part of the error will have grown so that the energy in this part will be greater than in the low frequency part.

□

In this introductory example of ILC we have used a frequency domain approach for the convergence analysis of the ILC feedback scheme. This approach has been applied in more general cases in, e.g. [MK85, Hor94, GN97b, NG98]. A time domain approach to solve the same problem is made by Arimoto et al. in, e.g., [AKM84a, Ari85, AKMT85]. 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 three different items will be more thoroughly discussed.

2.3 General Formulation

We are now going to describe the ILC concept in a more general form. The following two approaches to ILC will be discussed:

- The tracking approach.
- The disturbance rejection approach.

The example in the previous section represents the first category. This approach is also the one covered in , e.g., [Moo93]. The *disturbance rejection approach* is a new and alternative way of viewing the problem.

2.3.1 Preliminaries

Before formulating the abstract ILC approaches we are going to give some definitions. First the term iterative process will be defined and then some mathematical notations for iterative processes will be defined.

Definition 2.1 (Iterative Process)
An iterative process is a process that iteratively performs the same exercise over and over again.

□

Figure 2.3 shows an example of an iterative process: Water jet cutting of dashboards for cars and trucks. The dashboards are made of a plastic material.

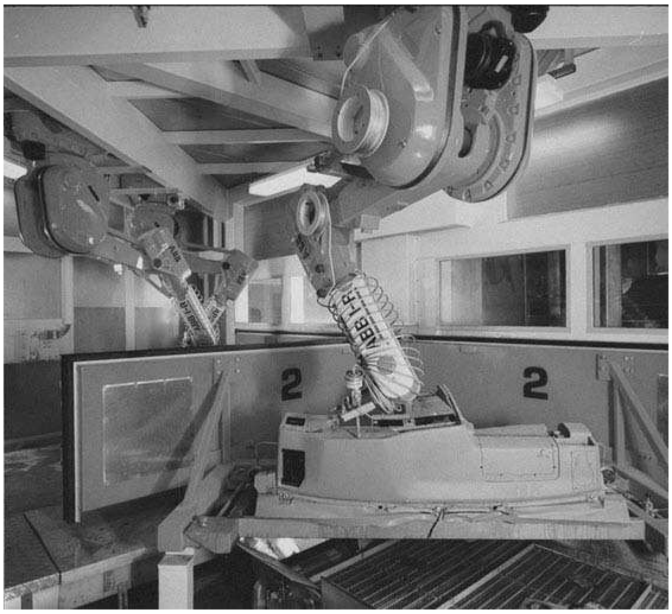


Figure 2.3 *An example of an iterative process, a water cut system from ABB. Robots are used to cut holes in dashboards for cars and trucks.*

To describe the controlled system we need a model, or in other words, a mathematical description. We are now going to define the different types of models that will be used in the thesis. First a general nonlinear model is defined and then a linear model is considered.

Definition 2.2 (A State Space Model)
A continuous time nonlinear state space model consists of two sets of functions, a set of first order differential equations called state equations*, and a set of* output equations*. This can be expressed as,*

$$\dot{x}(t) = f_c(x(t), u(t)) \tag{2.13a}$$
$$y(t) = g_c(x(t), u(t)) \tag{2.13b}$$

where $x(t) \in \mathbb{R}^n$ *is the* state*,* $u(t) \in \mathbb{R}^m$ *is the* control input*, and* $y(t) \in \mathbb{R}^p$ *is the* output*. In discrete time a nonlinear system is described by,*

$$x(t+1) = f_d(x(t), u(t)) \tag{2.14a}$$
$$y(t) = g_d(x(t), u(t)) \tag{2.14b}$$

with the sampling time, t_s*, assumed to be 1. A solution to the differential (difference) equations on an interval* $[0, t_f]$ *is called a* state trajectory*. The corresponding output,* $y(t) \in [0, t_f]$*, is called the* output signal*.*

□

The nonlinear state space model is often a result of physical modeling, but can also be the result of an identification process using a nonlinear black-box model [Lju87]. A useful subclass of the general system in Definition 2.2 is the linear time invariant (LTI) system. In this class of systems the state space description is,

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.15a}$$
$$y(t) = Cx(t) + Du(t) \tag{2.15b}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, and $D \in \mathbb{R}^{p\times m}$. A discrete time LTI state space description is obtained by changing $\dot{x}(t)$ to $x(t+1)$ in (2.15a).

Definition 2.3 (A Transfer Operator Model)
In continuous time a SISO transfer operator model *is given by,*

$$G(p) = \frac{B(p)}{A(p)} \tag{2.16}$$

where $y(t) = G(p)u(t)$, $A(p)$ *and* $B(p)$ *are polynomials in* p. *The symbol* p *represents the derivative operator, that is*

$$py(t) = \frac{d}{dt}y(t) \tag{2.17}$$

In discrete time we use the q *operator. This operator represents a time shift function,*

$$qy(t) = y(t+1) \tag{2.18}$$

Often the inverse operator, q^{-1}, *is used representing an inverse time shift operator.*

□

There is a correspondence between the linear state space representation and the transfer operator model. For the continuous time state space representation it is possible to express the transfer operator model by,

$$G(p) = C(pI - A)^{-1}B + D \tag{2.19}$$

We can arrive at the discrete time transfer operator from the discrete time state space representation in the same way, just replace p with q in (2.19). We will use the transfer operator model also in the MIMO case. The polynomials in p and q are then changed to matrices with polynomial fractions as elements. An example of a MIMO system $G(p)$ described on this form is,

$$G(p) = \begin{bmatrix} \frac{1}{p(p+1)} & \frac{0.1}{p+1} \\ \frac{10}{p(p+0.1)} & \frac{0.01}{p} \end{bmatrix} \tag{2.20}$$

which can be written using a matrix fraction description, $G(p) = B_R(p)A_R^{-1}(p)$. More on this representation and the properties of it can be found in [Kai80], here we will only use the form shown in (2.20).

In Example 2.1 we also used the frequency function.

Definition 2.4 (Frequency Function)
The frequency response function, or simply the frequency function can be calculated from the transfer operator by substitution. In the continuous time case substitute p *with* $i\omega$, $\omega \in \mathbb{R}$, *and in the discrete time case substitute* q *with* $e^{i\omega t_s}$, $\omega \in [-\pi/t_s, \pi/t_s]$, *where* t_s *is the sampling time.*

□

In Example 2.1 the transfer operator

$$G_C(q) = \frac{0.09516q^{-1}}{1 - 0.9048q^{-1}} \tag{2.21}$$

was used as a representation of the system. The corresponding frequency function is

$$G_C(e^{i\omega}) = \frac{0.09516e^{-i\omega}}{1 - 0.9048e^{-i\omega}} \tag{2.22}$$

A Bode diagram of this function is shown in the plot of Figure 2.4. The frequency function can also be generalized to the MIMO case in the same way as for the transfer operator.

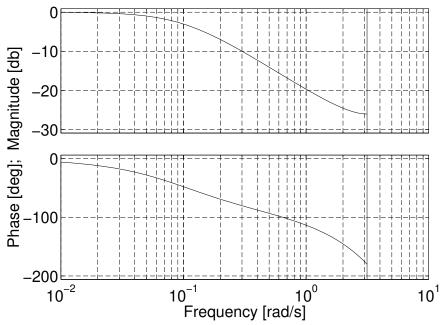


Figure 2.4 *A Bode diagram of the system in Example 2.1.*

Before considering the ILC approaches we will define the mathematical notation that we will use for ILC systems, i.e., the processes found in Definition 2.1.

Definition 2.5 (Mathematical Notation for Iterative Systems)
We will use the term iterative system for a system that, over a time interval $[0, t_f]$, repeats the same exercise. An index, k, called the iteration index, is used to separate the signals corresponding to different iterations.

□

With an *exercise* we will mean that the system starts from the same initial condition, possibly with a disturbance on the state, and that the system state trajectory, $x_k(t)$, and the output signal $y_k(t)$ are defined on the interval

$[0, t_f]$. It is not necessary that the state trajectory and the output signal is the same in the different iterations. We illustrate with an example.

Example 2.2 (An Iterative System) *Assume that we have a linear state space description of a system,*

$$\dot{x}_k(t) = Ax_k(t) + Br(t) + w_k(t) \tag{2.23a}$$

$$y_k(t) = Cx_k(t) + n_k(t) \tag{2.23b}$$

and that we repeatedly feed the system with the same reference signal, $r(t)$*, defined over the interval* $[0, t_f]$*. By introducing the iteration index* k *we can reset* t *every time the reference signal is followed and instead increase the iteration index* k *by one. In (2.23) we see that it is only the reference signal that is the same every iteration, all the other signals are separated among the iterations by the iteration index. The disturbances in* (2.23) *are the state disturbance,* $w_k(t)$*, also called the* *system disturbance*, *and the* *measurement disturbance*, $n_k(t)$.

□

In Figure 2.5 the error signal from the simulation in Example 2.1 is depicted. The error signal is defined as $e_k(t) = r(t) - y_k(t)$, The two-dimensionality of the problem is obvious when displaying the error as both a function of time and iteration. The stability of such two-dimensional systems have been investigated and some results are presented in [RO92].

Consider the system and the ILC feedback loop in Figure 2.6, cf. Figure 2.1. We will now define the terminology that will be used for the different components in the loop.

Definition 2.6 (The ILC updating formula)
The mechanism that updates the control signal in the ILC method will be referred to as the *ILC updating formula* *or simply the* *ILC* .

□

In Figure 2.6 the block that takes care of the updating of the control signal is labeled ILC. Note that the ILC can use the reference signal and all the previous control signals and output signals in the calculation of the new control signal. It is also, of course, possible to include the traditional feedback controller in the ILC. We will return to these topics in Section 2.5. An example of an ILC is the one used in Example 2.1,

$$u_{k+1}(t) = u_k(t) + e_k(t+1) \tag{2.24}$$

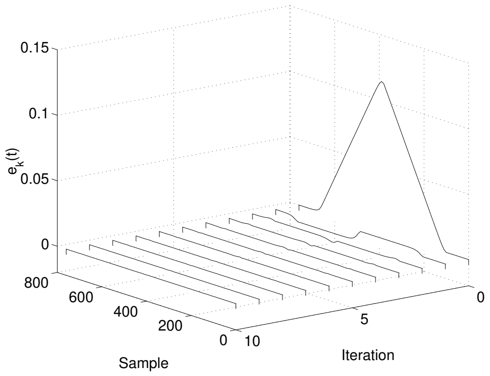


Figure 2.5 *A plot of the error signal $e_k(t)$ in Example 2.1 as a function of both k and t.*

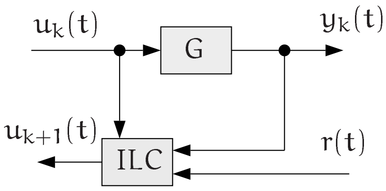


Figure 2.6 *A system G with an ILC feedback.*

where $L(q) = q$ and $e_k(t)$ is defined as the difference between the reference signal and the output signal. This ILC is a special case of a more general linear updating formula that will be discussed in Section 2.5. We can now define the name for the closed loop system.

Definition 2.7 (Iterative Learning Control Feedback)
The system under ILC feedback is called the ILCF system, (Iterative Learning Control Feedback system).

□

In Figure 2.6, the system G and the ILC forms an ILCF system. Causality for the ILC is something that has to be defined since it looks a bit different from what we are used to from the control literature. The ILC in (2.24) might not look like a causal updating formula, but since the error used in the updating of the control signal comes from the previous iteration k, it is actually possible to feed back the error from time instance $t+1$.

Definition 2.8 (Causality in Iterative Learning Control)
The ILC is causal when the control signal $u_{k+1}(t)$ is a function of

a) $r(t_1)$, $t_1 \in [0, t_f]$,

b) $y_{k+1}(t_2)$, $t_2 < t$, *and*

c) $u_j(t_3)$, $y_j(t_3)$, *with* $j \leq k$ *and* $t_3 \in [0, t_f]$.

□

Note that, if we use an ILC updating formula, e.g. (2.24), having the causality property of ILC we will have a boundary problem since the control signal, $u_{k+1}(t)$, at time $t = t_f$ will depend of the reference signal and the output signal at time instance $t = t_f + t_s$. The same type of boundary problem will of course also be present at time 0 if we, for example, use a zero phase Q filter. We are not going to cover these problems here but an idea to a solution is to assume that outside the time interval $[0, t_f]$, the signals will take the value that they have on the boundary. For example $y_k(t) \equiv y_k(t_f)$ when $t > t_f$. This problem is discussed in the computer vision literature where signals normally are limited in space, see e.g. [GK95].

We will now continue by defining the two ILC approaches that we cover in the thesis.

2.3.2 The Tracking Approach

Consider the system depicted in Figure 2.7b. It can be considered a general system with three inputs, the reference signal $r(t) \in \mathbb{R}^p$, an externally generated control signal $u(t) \in \mathbb{R}^m$, and a disturbance $\nu(t) \in \mathbb{R}^s$ that we cannot control. The output is the signal $y(t) \in \mathbb{R}^p$, representing the measurements on the system.

Remark 2.2
The reason why the system has also the reference signal as input is that the ILC method should be seen as a complement to the ordinary control scheme. Hence, the system T is a complete control system including both the controlled system, e.g. a robot arm, and the traditional controller. The traditional controller can in itself be a complicated system, perhaps nonlinear, with both a feedback and a feedforward term. Note that T does not

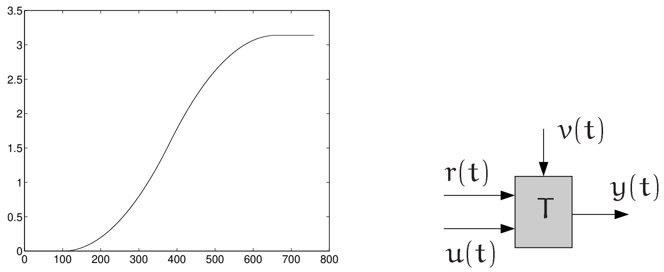
include the ILC. In the ILC sense the system T *is in open loop even if it involves feedback in the traditional control sense.*

□

The goal is to find an input signal, $u(t)$, to the system such that the tracking error is as small as possible. This means solving the following optimization problem,

$$\min_{u(t),\ t\in[0,t_f]} \|r(t) - y(t)\| \tag{2.25}$$

possibly with constraints on the control signal, $u(t)$. We are not going to solve this problem explicitly but, instead, try to find an iterative solution. The problem of iteratively finding the optimal control input has been discussed in, e.g., [Pla68]. The tracking problem, however, is not covered there. We can now formulate the tracking ILC problem.



a. A reference signal, $r(t)$.

b. A general system.

Figure 2.7 *A reference signal,* $r(t)$*, and a general system with inputs* $r(t), u(t)$ *and output* $y(t)$*. The input* $v(t)$ *is a disturbance acting on the system.*

Definition 2.9 (The Tracking ILC Problem)
Assume that the we have a system T*, as in Figure 2.7b. On the time interval* $[0, t_f]$ *the system* T *can be described by a nonlinear state space model, given by Definition 2.2, but with three inputs. The three inputs should be as in Figure 2.7b, i.e.,* $r(t) \in \mathbb{R}^p$*,* $u_k(t) \in \mathbb{R}^m$*, and* $v_k(t) \in \mathbb{R}^s$*, the reference*

input, control input and the disturbance input respectively. The output is $y_k(t) \in \mathbb{R}^p$. The reference, $r(t)$, is defined on the interval $[0, t_f]$.

The tracking ILC problem is to find a causal ILC updating formula,

$$u_{k+1} = f_{ILC}(r, y_{k+1}, y_k, \dots, y_{k-j}, u_k, u_{k-1}, \dots, u_{k-l}) \text{ and } j, l \geq 0 \tag{2.26}$$

such that when $k \to \infty$, the output, y_k, converges to a signal that minimizes the criterion

$$V = \|r(t) - y_k(t)\| \tag{2.27}$$

The norm in (2.27) is arbitrary, truncated to the interval $[0, t_f]$. Additional constraints can be applied to the control signal, u_k.

□

Note that the ILC updating formula (2.26) described here is very general. In the continuous time representation the mapping is from a set of functions of time, to another function of time, u_{k+1}. However, in reality this controller will never be implemented in continuous time and when nothing else is explicitly noted the mapping is assumed to be in discrete time. The system T is formulated in continuous time because it is usually a physical system, e.g. a robot, and this kind of system is more naturally modeled by differential equations.

We will now give an example of a system where the tracking ILC approach apply.

Example 2.3 *A realization of the system T can be a control system as depicted in Figure 2.8. In this case we have a closed loop system containing a feedback controller F and a feedforward controller F_f, controlling a system G. Given a reference signal, like in Figure 2.7a, and an ILC updating formula, like (2.4c), we have a tracking ILC problem.*

□

The formulation of the ILC problem in Definition 2.9 is so general that it is difficult to produce anything but very abstract results. However, in a more specific formulation, like for linear systems, it is quite easy to find criteria for convergence as seen in Example 2.1. It is also possible to formulate algorithms for the design of ILC updating formulas, as will be discussed in Chapter 4.

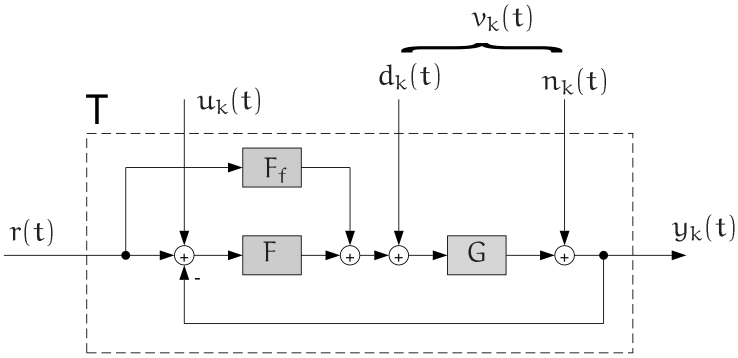


Figure 2.8 *An example of a realization of the system* T.

2.3.3 The Disturbance Rejection Approach

In the previous section we saw how ILC can be used to solve a tracking problem. Now we are going to approach the same problem from a slightly different angle and show that the problem of tracking can also be seen as a problem of disturbance rejection.

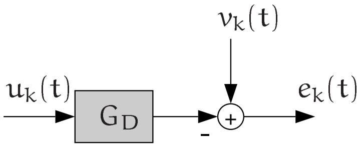


Figure 2.9 *A system with input* $u_k(t)$ *and an unknown disturbance,* $\nu_k(t)$*, acting on the output of the system* G_D.

Consider the system in Figure 2.9. Assume that the input, $u_k(t)$, should be altered in such a way that the output, $e_k(t)$, is minimized. The immediate solution to this problem would be to identify the system G_D and try to predict the disturbance $\nu_k(t)$. We will now use the ILC approach to minimize the signal $e_k(t) = \nu_k(t) - G_D(q)u_k(t)$ as we did in the previous section, cf. (2.27).

Definition 2.10 (The Disturbance Rejection ILC Problem)
Suppose that we have a system G_D*, as in Figure 2.9. On the time interval* $[0, t_f]$ *the system* G_D *is described by the following linear discrete time*

operator model,

$$e_k(t) = -G_D(q)u_k(t) + \nu_k(t) \tag{2.28}$$

where the disturbance $\nu_k(t)$ *is given by,*

$$\nu_k(t) = d(t) + n_k(t) \tag{2.29}$$

the term $d(t)$*, for fixed* t*, is constant* $\forall k$*, and* $n_k(t)$ *is a random variable.*

The disturbance rejection ILC problem is to find a causal ILC updating formula,

$$u_{k+1} = f_{ILC}(e_{k+1}, e_k, \ldots, e_{k-j}, u_k, u_{k-1}, \ldots, u_{k-l}) \text{ and } j, l \geq 0 \tag{2.30}$$

such that, when $k \to \infty$ *the control signal,* $u_k(t)$*, converges to a signal that minimizes* $\|e_k(t)\|$*.*

□

If it was possible to measure the disturbance $d(t)$ in Figure 2.9, the best possible control signal would be,

$$u(t) = G_D^{-1}d(t) \tag{2.31}$$

i.e. to use disturbance feedforward. A condition is, of course, that the inverse of the system exists and is stable.

With the disturbance rejection approach to ILC we want to simplify the formulation of the problem. The idea is also to try to find connections with other research areas, like system identification. These ideas are not completely worked out yet, however, we have made this a point for further work in Chapter 8. To show that the two ILC approaches solve the same problem, in the linear case, we will now present a comparison of the two methods.

2.3.4 Comparison of the Two ILC Approaches

We will now give an explanation to why the two formulations of the ILC problem are relevant. First we will make the assumption that the systems T and G_D are linear systems.

Consider the system in Figure 2.10. This is the same system as in Figure 2.7 but with the error, $e_k(t)$ as output, instead of $y_k(t)$. If the output of the system in Figure 2.10 is generated as

$$e_k(t) = r(t) - y_k(t) = r(t) - T_r(q)r(t) - T_u(q)u_k(t) \tag{2.32}$$

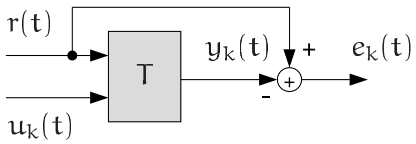


Figure 2.10 *The general system in Definition 2.9 with the error, $e_k(t)$, as output.*

we can define

$$\nu_k(t) = (I - T_r(q))r(t), \quad G_D(q) = T_u(q) \tag{2.33}$$

and we will arrive at the same form as in Definition 2.10 and Figure 2.9.

Remark 2.3
If we consider the realization of the system T in Example 2.3 it might not be obvious that the transfer operator from $u_k(t)$ to $e_k(t)$ is equal to $T_u(q)$, i.e. $G_C(q)$ in the particular realization shown in Figure 2.8. To understand this it is important to note that $e_k(t)$ is not *equal to the control error, i.e. the input to the controller F. The error signal, $e_k(t)$ is instead equal to $r(t) - y_k(t)$. The reference signal, $r(t)$, does not change from iteration to iteration, although the control loop in Figure 2.8 will experience the applied additional control signal, $u_k(t)$ as a change in the reference signal. For the controller the change in the virtual reference signal to the control loop can be interpreted as a* fake target, *leading to a decreased $\|e_k(t)\|$.*

□

In the experiments described in Chapter 7 the idea of *fake target* will be used.

2.4 Stability of ILCF Systems

Stability is a general but very important property for a control system. An unstable system is in practice useless, maybe even dangerous. We are now going to define stability for the class of ILCF systems that will be considered in the thesis. The questions we want to answer are typically:

- Will the error approach zero as the number of iterations grows?
- If the error will not approach zero, will it be bounded?

When defining stability for general nonlinear systems we consider stability of a an equilibrium point [SL91, Lya92]. An equilibrium point has a natural interpretation as a point in the nonlinear state space of the system, cf. Definition 2.2. This direct interpretation is not so easy to incorporate in the ILCF framework. The equilibrium point is not represented by a point in the state space but, rather, as an error trajectory, $e_k(t)$, or a control trajectory, $u_k(t)$, where $k \geq 0$ and $t \in [0, t_f]$.

2.4.1 Basic Assumptions

We have some assumptions on the system that we control using ILCF as well as the ILCF system as a whole. We start by discussing the stability of the controlled system.

Stability of the controlled system

When applying Iterative Learning Control to a system, as shown in Figure 2.6, we always assume that the system T or G_D is stable. We will, in other words, demand stability along the time axis of the system. Hence, the ILCF stability will only concern stability along the k-axis, and, hence, we can consider stability only with respect to the output of the system or a function of this signal. An example of a simulation of a stable ILCF system is shown in Figure 2.5. As is noted in [Be98] the assumption on stability of T and G_D is not necessary. Since the time horizon is finite the state will always be bounded. The case of system stabilization using ILC will, however, not be covered here, interested readers are referred to [Be98].

Structure of the system

In the stability definitions we will define stability with respect to the error signal, $e(t)$. This signal can be defined as the reference signal minus the measured signal or as a function thereof. The state trajectory of the system T or G_D will not be discussed since it is assumed to be stable according to the first assumption. The stability of the ILCF system will instead be characterized by a convergence of the control signal $u_k(t)$ and the error signal $e_k(t)$.

2.4.2 Stability

To be able to describe the notion of stability for ILCF systems we need to give some definitions. First we need an equivalent to the equilibrium point used in the traditional stability analysis. In ILC the equilibrium trajectory will be used instead.

Definition 2.11 (An Equilibrium Trajectory)
An error trajectory $e^*(t)$, $t \in [0, t_f]$, is an equilibrium trajectory of the ILCF system if once the system has reached the trajectory $e^*(t)$ it will follow this trajectory for all future iterations.

For the stable choice of ILC in Example 2.1 the equilibrium trajectory is $e_\infty(t) = e^*(t) \equiv 0$, $t \in [0, t_f]$. Note that we can define an equilibrium output trajectory or control input trajectory in the same way. Using this definition of equilibrium trajectory we can now define stability of an equilibrium trajectory.

Definition 2.12 (Stability of an Equilibrium Trajectory)
Given an ILCF system, the equilibrium trajectory $e^*(t)$, $t \in [0, t_f]$ is said to be stable if $\forall R > 0$, $\exists r > 0$ such that

$$\|e_0(t) - e^*(t)\| < r \implies \forall k \geq 0,\ \|e_k(t) - e^*(t)\| < R \tag{2.34}$$

Otherwise the equilibrium trajectory is unstable.

□

The definition says that given a region R around the equilibrium trajectory, we can always find a small region r with the property that; if we start in the region given by r we will never leave the region R. This stability definition does not imply that the equilibrium trajectory will ever be reached but it is possible to be arbitrarily close by starting sufficiently close to it.

The difference between this definition and most of the previous ones found in the literature, e.g. [Hid92], is that we do not explicitly assume convergence to zero error.

We can now continue by defining asymptotic stability, exponential stability and global stability.

Definition 2.13 (Asymptotic Stability)
Given an ILCF system, an equilibrium trajectory is said to be asymptotically stable if it is stable and,

$$\exists r > 0,\ \|e_0(t) - e^*(t)\| < r \implies \lim_{k\to\infty} \|e_k(t) - e^*(t)\| = 0 \tag{2.35}$$

□

Definition 2.14 (Exponential Stability)
Given an ILCF system, an equilibrium trajectory is said to be exponentially stable if $\exists r > 0$ such that $\|e_0(t) - e^*(t)\| < r$ and,

$$\exists \alpha,\ \lambda > 0 \implies \forall k > 0,\quad \|e_k(t) - e^*(t)\| \le \alpha \|e_0(t) - e^*(t)\| e^{-\lambda t} \tag{2.36}$$

□

Definition 2.15 (Global stability)
If asymptotic (or exponential) stability holds for any initial error trajectory, $e_0(t)$, the equilibrium trajectory $e^*(t)$, $t \in [0, t_f]$, is said to be globally asymptotically (or exponentially) stable.

□

For linear systems there will only exist one equilibrium trajectory, asymptotic stability will therefore automatically imply global asymptotic stability.

Now when the basic terminology is defined and the abstract problem is formulated, different types of learning controllers will be discussed.

2.5 The ILC updating formula

It is obvious that the properties of the ILCF system depend completely upon the ILC updating formula. In this section some different approaches to updating the signal u_k are discussed based on [GN97b]. The ILC can be categorized in two groups according to how the errors from previous trials are utilized. We will first define these groups.

Definition 2.16 (First order ILC)
An ILC that only uses the error in the previous iteration is called a first order ILC .

□

An example of a first order ILC is the one used in Example 2.1. Note that the order of the filter L is not specified. Now we can define high order ILC.

Definition 2.17 (High order ILC)
When the ILC uses errors from more than the previous iteration it is a high order ILC . Of course the term second order, third order, etc., can be used when the order of the ILC is to be emphasized.

□

An ILC is not only characterized by whether it is a *first order* or a *high order* ILC. In the next two sections we will look at the ILC from a linear and nonlinear perspective. We start by considering the linear ILC.

2.5.1 Linear ILC

The class of linear ILC updating formulas are naturally divided into first order and higher order ILCs. We will first discuss the first order ILC.

First order ILC

Many different types of Learning Controllers have been presented up to now, all having different properties. In the original paper on ILC [AKM84a] by Arimoto et al. the following continuous time ILC is used,

$$u_{k+1}(t) = u_k(t) + \Gamma \frac{d}{dt} e_k(t) \tag{2.37}$$

The adjustable parameter in this ILC is the gain matrix, or in the SISO case, the scalar, Γ. This kind of updating equation can be generalized to the following form, discussed in, e.g. [Ari85],

$$u_{k+1}(t) = u_k(t) + \{\Gamma \frac{d}{dt} + \Phi + \Psi \int dt\} e_k(t) \tag{2.38}$$

This is an attempt to introduce a PID type ILC updating formula. We are not going to examine the convergence properties for the suggested ILC here but in [Ari85] the ILC in (2.38) was applied to a first order system. We will see in the analysis, Chapter 3, why this makes sense. It is obvious that for a class of systems the ILC in (2.38) works well. In general, however, there are other types of controllers that work better and, as will be shown in the subsequent chapters, are more robust. A generalization of equations (2.37) and (2.38) is,

$$u_{k+1}(t) = Q(q)(u_k(t) + L(q)e_k(t)) \tag{2.39}$$

where $Q(q)$ and $L(q)$ are considered to be linear transfer operators or simply discrete time filters. As was noted in Section 2.3 the filters Q and L do not need to be causal[3] in the normal sense. Compared to (2.37) the derivative of the error is generalized to a filter operation. There is also a weighting function or filter for the previous control signal. This general linear first order ILC has been presented in a stepwise procedure, with early contributions by, Togai, Yamano [TY85], and Mita, Kato [MK85]. The use of the Q filter is suggested in [HYON88] and [TTC89].

Other structures have also been presented. The following ILC

$$u_{k+1}(t) = H_1(q)u_k(t) + H_2(q)e_k(t) \tag{2.40}$$

is found in, e.g., [GN97a, GN97c]. It might look more general than (2.39) but since Q and L can be picked arbitrarily we can for example choose $Q = H_1$ and $L = \frac{H_2}{Q}$ in order to get the same form as in (2.40).

High order ILC

In the early development of ILC theory most contributions were on first order ILCs. In a recent paper by Chen et al. [CGW98] the advantages of using high order ILC is discussed. A general linear time invariant high order ILC is given by

$$u_{k+1}(t) = \bar{Q}(q)u_0(t) + \sum_{j=0}^{k} \bar{H}_j(q)e_j(t) \tag{2.41}$$

where $\bar{Q}(q)$ and $\bar{H}_j(q)$, $j = 0, \ldots, k$ represent linear transfer operators and $u_0(t)$ is the initial control signal. The control signal for the next trial is constructed from the control signal in the initial iteration and the errors in all the previous iterations, each of them filtered through some filters, $\bar{H}_j$. The filters do not need to be proper.

To get some insights in the high order ILCF, assume that the filters $\bar{H}_j$ are just constants, i.e.

$$\bar{H}_j = \bar{h}_j, \quad \forall\, j \tag{2.42}$$

Let us also assume that $u_0(t) \equiv 0$. From (2.41) we get,

$$u_{k+1}(t) = \sum_{j=0}^{k} \bar{h}_j e_j(t) \tag{2.43}$$

[3]Recall Definition 2.8 of *causality* in ILC.

The control signal at time t in iteration $k+1$ will be a weighted sum of the errors at time t in the previous iterations. This can be interpreted as filtering of the error in the iteration direction with the filter impulse coefficients $\bar{h}_j$. Applying the same idea to the ILC,

$$u_{k+1}(t) = u_k(t) + \gamma e_k(t) \tag{2.44}$$

we see that this can be interpreted as a filtering of the error e_k by the discrete time filter

$$P(q) = \frac{\gamma}{q-1} \tag{2.45}$$

i.e., a pure integrator. Note that q here works in the k direction. By introducing also e_{k-1} in the update equation we obtain a second-order ILCF, often referred to as a two-step algorithm,

$$u_{k+1}(t) = u_k(t) + \gamma_1 e_k(t) + \gamma_2 e_{k-1}(t) \tag{2.46}$$

This corresponds to the filter

$$P(q) = \frac{\gamma_1 q + \gamma_2}{q(q-1)} \tag{2.47}$$

which is of PI-type.

Remark 2.4 (Aspects of the ILC)
We have now seen two different types of ILC updating formulas using the conventional PI and PID concepts. It is important to understand the difference between these two. In the PID of (2.38) it is only a PID control using the error from the previous iteration. We saw in the Example 2.1 that the P-type ILC gave an unstable system but it is clearly possible to find systems that also give convergence using only the P-type ILC. From the convergence criterion in Example 2.1 we see that in order for the criterion to be fulfilled having only P-type ILC a necessary condition is that the system G_C is SPR (strictly positive real).

The PI(D) controller, in the high order ILC sense, is a PI(D) controller over the iterations, or in the k-direction. It is important to note that introducing a D-part, in this sense, will be the same as introducing feedback in the control loop. The D-term will act on the current error and this is feedback in the traditional interpretation. Assume that $P(q)$ in (2.45) is given by

$$P(q) = \frac{q-1}{q} \tag{2.48}$$

where q works in the iteration direction. Using this ILC we get the following update equation according to (2.46)

$$u_{k+1}(t) = e_{k+1}(t) - e_k(t) \tag{2.49}$$

i.e. a differentiation over the iterations. Using a more advance type of ILC including also the PI-part for improved control has been discussed in the ILC literature, e.g. the CITE[4] feedback [CWXS96, CXL96a, CXL96b].

□

The idea of utilizing the errors from more than the previous iteration has been covered in many articles. In [LL93] 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 [Ari91] the errors from previous iterations are used in an indirect way. Chen et al. has also investigated the use of high order ILC and the main reference is [CGW98], but the issue is also discussed in [CWD97, CXW97].

A general linear ILC

We will now conclude this section about the ILC updating formula by introducing a general linear ILC. Previously we have discussed how the updating formula can be interpreted as a filtering of the error and the control signal in both the t- and the k-direction. We will now formalize this by introducing a two dimensional filter, or updating formula. The general updating formula is with this formalism given by

$$u_k(t) = P(q_k, q_t)e_k(t), \;\; k = 0, 1, 2, \ldots \quad t \in [0, t_f] \tag{2.50}$$

where $P(q_k, q_t)$ is a rational function

$$P(q_k, q_t) = \frac{P_B(q_k, q_t)}{P_A(q_k, q_t)} \tag{2.51}$$

The functions $P_A(\cdot,\cdot)$ and $P_B(\cdot,\cdot)$ are both polynomial functions in q_k and q_t, where the operators q_k and q_t are defined as

$$q_k u_k(t) = u_{k+1}(t), \quad q_t u_k(t) = u_k(t+1) \tag{2.52}$$

We see that q_k works in the iteration direction while q_t works in the time direction. Note that the indices k and t are not evaluated, instead they are

[4]Current Iteration Tracking Error

symbols that help in the interpretation of the operators. The polynomials P_A and P_B can be written as polynomials in q_k with coefficients being polynomials in q_t. This will be shown in an example.

Example 2.4 *The ILC,*

$$u_{k+1}(t) = u_k(t) + H_1(q)e_k(t) + H_2(q)e_{k-1} \tag{2.53}$$

can be expressed using the representation with q_k and q_t as

$$u_{k+1}(t) = \frac{H_1(q_t)q_k + H_2(q_t)}{q_k - 1} e_k(t) \tag{2.54}$$

where P_B and P_A are polynomials in q_k with coefficients being polynomials in q_t.

□

If we apply P_B to $e_k(t)$ this will result in a sum of filtered versions of $e_j(t)$, $j \leq k$. We can see that this captures the different types of linear ILC that we have discussed earlier in this section, also the ILC in (2.41)

2.5.2 Nonlinear ILC

Most of the work in the area of ILC has been done on linear ILC updating formulas. When the ILC problem was formulated in Section 2.3 the description of the ILC was,

$$u_{k+1} = f_{ILC}(r, y_{k+1}, y_k, \dots, y_{k-j}, u_k, u_{k-1}, \dots, u_{k-l}) \tag{2.55}$$

This mapping can be a general mapping from the reference signals, the previous measurements on the system, and the previous control signals. It can be even more general if the measurements from the current iteration are included as well. In this very general framework not so many results are available. There are, however, some results in a survey on ILC by Moore [Moo93] and in a recent book [Be98]. 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 to the topic and with this approach not only the control signal changes over the iterations but also the ILC. For a more thorough discussion of this kind of ILC see [Moo93, Be98] and the references found there. A general discussion of ANN in identification is found in, e.g., [Sjö95].

2.6 Summary

This chapter is mainly devoted to definitions to create a framework for the rest of the thesis. The goal is also to explain the ideas behind ILC and the example in Section 2.2 is meant to give a feeling for the ILC method. We want to stress that the system that we consider being controlled by ILC is a closed loop system in the traditional control theory interpretation. The general ILC achieved with both the tracking approach and the disturbance rejection approach can, however, also include a normal feedback, cf. CITE. We can summarize the main parts in the chapter as follows:

- The definition of iterative systems, and the notation used for this class of systems, Definitions 2.1, and 2.5.
- The terminology used for the mechanism that updates the control signal in the ILC method, *the ILC updating formula*, or simply the ILC, and the definition of the ILCF system, Definitions 2.6 and 2.7.
- The definition of the tracking approach to ILC in Section 2.3.2 and the disturbance rejection approach in Section 2.3.3.
- The approach to define stability using the notion of equilibrium trajectories, Definition 2.11. This is an extension to the stability theory for non-linear state space models.
- The discussion on ILC updating formulas in Section 2.5 and the Definitions 2.16 and 2.17, the first order ILC and the high order ILC, respectively.

3

Analysis

In this chapter, the analysis of the ILCF system will be made for the ILC approaches discussed in the previous chapter, the *tracking approach* and the *disturbance rejection approach*.

3.1 Introduction

Before going into the analysis, some of the tools that will be used in the analysis will be discussed.

3.1.1 Linear Iterative Systems

We will now give a theorem that will be used in the subsequent analysis of linear ILCF system. Consider the following iterative system,

$$z_{k+1}(t) = F_1(q)z_k(t) + u(t) \tag{3.1}$$

where $z_k(t) \in \mathbb{R}^n$, $F_1(q)$ is a matrix of discrete time transfer operators, and $u(t) \in \mathbb{R}^n$. We give the following theorem for stability of linear iterative systems.

Theorem 3.1 (Global Asymptotic Stability)
Consider the iterative system given by

$$z_{k+1}(t) = F_1(q)z_k(t) + u(t) \tag{3.2}$$

where $z_j(t) \in \mathbb{R}^n$, *and* $F_1(q)$ *is a discrete time transfer operator matrix, cf. Definition 2.3. The system is globally asymptotically stable if and only if*

$$\gamma = \|F_1(e^{i\omega t_s})\|_\infty < 1 \tag{3.3}$$

□

Proof. *For the proof see Appendix 3.A.*

■

The limit trajectory can be calculated using the following corollary.

Corollary 3.1
It follows from Theorem 3.1 that if the stability condition is fulfilled the signal $z_k(t)$ *will converge to*

$$\lim_{k \to \infty} z_k(t) = z_\infty(t) = (I - F_1(q))^{-1} u(t) \tag{3.4}$$

□

Proof. *If the stability criterion is fulfilled the signal* $z_k(t)$ *will converge and by letting* $k \to \infty$ *in (3.2) the result follows. See also Appendix 3.A.*

■

These two results have not been explicitly formulated in the ILC literature before. The analysis of linear iterative systems is covered by a book by Rogers and Owens [RO92]. Results corresponding to Theorem 3.1 are presented but the results are limited to the first order iterative processes. An interesting property of the approach presented here is that it also covers higher order ILCF systems, i.e. when $z_{k+1}(t)$ in (3.1) not only depend on a filtered version of $z_k(t)$ but also filtered versions of $z_j(t)$ where $j < k$. We illustrate this by an example.

Example 3.1 *We have a second order iterative system given by*

$$z_{k+1}(t) = F_1(q) z_k(t) + F_2(q) z_{k-1}(t) + u(t) \tag{3.5}$$

This iterative system can be written in the following form

$$\zeta_{k+1}(t) = F(q) \zeta_k(t) + \beta u(t) \tag{3.6}$$

where

$$\zeta_k(t) = \begin{bmatrix} z_k(t) \\ z_{k-1}(t) \end{bmatrix}, \quad F(q) = \begin{bmatrix} F_1(q) & F_2(q) \\ I & 0 \end{bmatrix}, \textit{and} \ \ \beta = \begin{bmatrix} I \\ 0 \end{bmatrix} \tag{3.7}$$

From Theorem 3.1 we get a criterion for stability, $\|F(e^{i\omega t_s})\|_\infty < 1$.

□

We will come back to the results on higher order linear iterative systems later in this chapter.

3.2 The Tracking Approach

We are now going to analyze stability for the ILC *tracking approach*, defined in the previous chapter. Here the analysis will cover mainly *linear time invariant (LTI)* systems but some results on controlling nonlinear systems using ILC will also be presented. We will start by considering the control of LTI systems.

3.2.1 LTI systems

Most of the results in the area of ILC are for LTI systems. It is, in general, much easier to show stability and convergence for this class of systems than, e.g., for nonlinear systems. Here results on both first order ILC and high order ILC will be shown. The system that we will control using ILC is given by,

$$y_k(t) = T_r(q)r(t) + T_u(q)u_k(t) + T_\nu(q)\nu_k(t) \tag{3.8}$$

where $T_r(q)$, $T_u(q)$, and $T_\nu(q)$ are transfer operators from the reference signal, the additional control signal, and the disturbances respectively, cf. Figure 2.7. The system description in (3.8) is inspired by the work of Moore [Moo93, Moo98], but the reference input and the disturbance input are not included there. The properties of the ILCF system will now be analyzed when we use the linear ILC updating formulas defined in the previous chapter. In this analysis we will only consider the discrete time system in (3.8). This follows from the assumption that we use a discrete time ILC.

First order ILC

The ILC updating formula that will be considered here is given by

$$u_{k+1}(t) = Q(q)(u_k(t) + L(q)e_k(t)) \tag{3.9}$$

where $Q(q)$ and $L(q)$ are linear discrete time filters. This kind of ILC has been discussed in Section 2.5.1 and can be considered a general first order ILC updating formula.

Nominal stability

Before considering the control of a MIMO we will look at the SISO case. Using the definition of $u_k(t)$ in (3.9) and the system description in (3.8) the error can be expressed in the frequency domain as

$$\begin{aligned} E_{k+1} &= R - Y_{k+1} = \underbrace{R - T_r R}_{\tilde{E}} - T_u U_{k+1} - T_v V_{k+1} \\ &= \tilde{E} - T_u Q(U_k + LE_k) - T_v V_{k+1} \end{aligned} \tag{3.10}$$

We have skipped the arguments in (3.10) in order to make the expression easier to read. The term $\tilde{E}(e^{i\omega t_s})$ is the error in the first iteration when the disturbances are assumed to be zero. By adding and subtracting $Q(e^{i\omega t_s})E_k(e^{i\omega t_s})$ on the right hand side we arrive at

$$\begin{aligned} E_{k+1} = (I - Q)\tilde{E} + Q(I - T_u L)E_k + \\ T_v(QV_k - V_{k+1}) \end{aligned} \tag{3.11}$$

where the homogeneous part gives the stability criterion. We will use this expression to get some insights when considering the disturbance effect of the method. One thing is possible to see immediately from (3.11); if there are no disturbances, i.e. $V_k \equiv 0$, we cannot reach zero error in general without choosing $Q \equiv 1$.

Remark 3.1
The reason why these calculations do not hold in the MIMO case is that the transfer operators T_u, T_v and Q do not commute in general.

□

We can now continue with the MIMO case and formulate a generalization of the result that we found in the introductory example in Chapter 2.

Theorem 3.2 (Stability criterion)
Given an LTI system of the general type, described by (3.8); cf. Figure 2.7. Applying the linear ILC in (3.9) we will have global asymptotic stability if and only if

$$\|Q(e^{i\omega t_s})(I - L(e^{i\omega t_s})T_u(e^{i\omega t_s}))\|_\infty < 1 \tag{3.12}$$

where Q and L are the filters used in the ILC updating formula and T_u is the transfer function from the applied control signal, $u_k(t)$, to the output of the system.

□

Proof. *Using (3.8) and (3.9) we get*

$$\begin{aligned} u_{k+1}(t) &= Q(q)\big(u_k(t) + L(q)e_k(t)\big) = Q(q)\big(u_k(t) + L(q)(r(t) - y_k(t))\big) \\ &= Q(q)\big((I - L(q)T_u(q))u_k(t) + L(q)(I - T_r(q))r(t)\big) \end{aligned} \tag{3.13}$$

Now, let

$$F_1(q) = Q(q)\big(I - L(q)T_u(q)\big) \tag{3.14}$$

and

$$r_F(t) = Q(q)L(q)(I - T_r(q))r(t) \tag{3.15}$$

we get

$$u_{k+1}(t) = F_1(q)u_k(t) + r_F(t) \tag{3.16}$$

From Theorem 3.1 we have that the iterative system in (3.16) is globally asymptotically stable if and only if $\|F_1(e^{i\omega t_s})\|_\infty < 1$, i.e.

$$\gamma = \|Q(e^{i\omega t_s})(I - L(e^{i\omega t_s})T_u(e^{i\omega t_s}))\|_\infty < 1 \tag{3.17}$$

where t_s is the sample time.

■

We make an important observation based on the stability criterion.

Remark 3.2

In the SISO case we can do the following important interpretation of the effect of the Q filter on the stability of the ILCF system. If Q is invertible, (3.12) *can be rewritten in the following form,*

$$|1 - L(e^{i\omega t_s})T_u(e^{i\omega t_s})| < |Q^{-1}(e^{i\omega t_s})|, \quad \forall \omega \in [-\pi/t_s, \pi/t_s] \tag{3.18}$$

As was mentioned in Section 2.1 the criterion in (3.18) *can be interpreted in a Nyquist diagram. If* $Q \equiv 1$ *the criterion becomes that the Nyquist curve of* $L(q)T_u(q)$ *should be kept within a unit circle centered at* $+1$. *If* Q *is chosen different from one the convergence region can be extended or restricted. By choosing* Q *less than one the region is extended in that frequency range and vice versa. The* Q *filter has, in this sense, a robustness property since we can use it to extend the stability region. The price for the robustness is that the ILC will not bring the system to zero error.*

□

A similar interpretation of the Q filter can be done also in the MIMO case but we have to use the criterion as it is formulated in (3.12). We will return to this in Chapter 4. Based on Corollary 3.1 we can formulate the following result for the iterative system in (3.13).

Corollary 3.2

It follows from Theorem 3.2 that if the condition is fulfilled the control signal will converge to

$$\begin{aligned}\lim_{k\to\infty} u_k(t) &= u_\infty(t) = \\ &\Big(I - Q(q)\big(I - L(q)T_u(q)\big)\Big)^{-1} Q(q)L(q)\big(I - T_r(q)\big)r(t)\end{aligned} \tag{3.19}$$

using this result the equilibrium error trajectory can be expressed as

$$\begin{aligned}e_\infty(t) = r(t) - y_\infty(t) = \\ &\bigg(I - T_r(q) - T_u(q)\Big(I - Q(q)\big(I - L(q)T_u(q)\big)\Big)^{-1} \\ &Q(q)L(q)(I - T_r(q))\bigg)r(t)\end{aligned} \tag{3.20}$$

□

Proof. *If the stability criterion in Theorem 3.2 is fulfilled the control signal will converge to an equilibrium trajectory,* $u_\infty(t)$*. Insert this equilibrium trajectory in the time domain version of* (3.13) *and solve for* $u_\infty(t)$*. Using this result,* (3.20) *follows by substitution.*

■

We can now show an application of Theorem 3.2.

Example 3.2 *Recall Example 2.1 where*

$$\begin{aligned} T_u &= G(q) \\ T_r &\equiv 0 \\ Q &\equiv 1 \\ L &= q \end{aligned} \tag{3.21}$$

The stability criterion according to Theorem 3.2 is

$$\|1 - qG(e^{i\omega t_s})\|_\infty < 1 \tag{3.22}$$

which is equivalent to the result in Example 2.1.

□

There is an additional lemma that can be very useful in the construction of the filters Q and L.

Lemma 3.1 (Zero error convergence)
An asymptotically stable ILCF system described by (3.8) *and the ILC in* (3.9) *has the equilibrium error trajectory* $e_\infty(t) \equiv 0$ *if and only if* $Q \equiv 1$.

□

Proof. *Since the stability condition is met there exists an equilibrium input trajectory,* $u_\infty(t)$*. Show (*$\Leftarrow$*): Consider the ILC when the equilibrium trajectory,* u_∞*,* e_∞*, are obtained*

$$u_\infty(t) = Q(q)(u_\infty(t) + L(q)e_\infty(t)) \tag{3.23}$$

If $Q \equiv 1$ *the error must obviously be zero (if* $L(q)$ *has full rank). (*$\Rightarrow$*) : Let* $Q = I + \epsilon$ *in* (3.23)*. This gives us,*

$$\epsilon u_\infty(t) = \tilde{L} e_\infty(t) \tag{3.24}$$

where $\tilde{L} = (I + \epsilon)L$. *To have* $e_\infty \equiv 0$, *either* $\epsilon = 0$ *or* $u_\infty \equiv 0$. *If the system starts from zero initial conditions, the only way* u_∞ *can be zero is if* $r \equiv 0$ *or* $T_r \equiv 1$. *When* $r \equiv 0$ *we have a trivial case where the system, without disturbances, should be kept in the initial position. The case with* $T_r \equiv 1$ *means that there is no error in the first iteration, and the ILC will never be used.*

■

We make an important observation based on Lemma 3.1.

Remark 3.3
In Lemma 3.1 it was shown that the error will converge to zero only if $Q \equiv 1$. *If we let* $Q \equiv 1$, *the error updating equation in* (3.10) *gives an interesting result*

$$E_1 = \tilde{E} - T_u(U_0 + LE_0) = \tilde{E} - T_u L\tilde{E} = (I - T_u L)\tilde{E} \tag{3.25}$$

where we have used that we have $u_0(t) \equiv 0$ *in the ILC tracking case. We see that we get the best performance when* L *is chosen as the inverse of the transfer function from the additional control signal to the plant output. In the next chapters this will be more developed and it will also result in algorithms for designing the ILC filters.*

□

We will now continue by looking at an example of the ILC tracking approach.

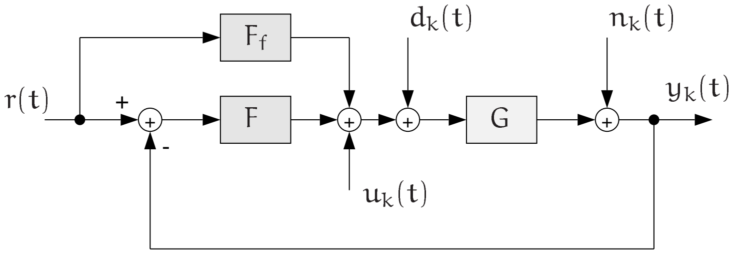


Figure 3.1 *An example of a system* G *controlled by feedforward and feedback controllers.*

Example 3.3 *Consider the SISO system depicted in Figure 3.1. If we compare this with the system description in (3.8) we see that we can express the transfer operators from reference signal and control signal as a function of the feedback controller* F*, the feedforward controller* F_f*, ,and the controlled system* G*. First we introduce the transfer function for the closed loop system,*

$$G_C(q) = \frac{F(q)G(q)}{1 + F(q)G(q)} \tag{3.26}$$

We can now write the transfer operator from the reference to the output,

$$T_r(q) = (F_f(q)F^{-1}(q) + 1)G_C(q) \tag{3.27}$$

and from the control signal to the output,

$$T_u(q) = F^{-1}(q)G_C(q) \tag{3.28}$$

The error that we will minimize is, as usual, given by $e_k(t) = r(t) - y_k(t)$*. We assume that there are no disturbances acting on the system. The convergence criteria becomes*

$$\begin{aligned} &\|Q(e^{i\omega t_s})\big(1 - \tilde{L}(e^{i\omega t_s})T_u(e^{i\omega t_s})\big)\|_\infty \\ &= \|Q(e^{i\omega t_s})\big(1 - \tilde{L}(e^{i\omega t_s})F^{-1}(e^{i\omega t_s})G_C(e^{i\omega t_s})\big)\|_\infty < 1 \end{aligned} \tag{3.29}$$

according to (3.12). By picking the filter $\tilde{L}(q)$ *as* $\tilde{L}(q) = F(q)L(q)$ *we get the following criterion,*

$$\|Q(e^{i\omega t_s})\big(1 - L(e^{i\omega t_s})G_C(e^{i\omega t_s})\big)\|_\infty < 1 \tag{3.30}$$

This is equivalent to choose the error signal to be the output from the controller F *in Figure 3.1 because we get the following updating equation for the ILC control signal,*

$$\begin{aligned} u_{k+1}(t) &= Q(q)\big(u_k + \tilde{L}(q)e_k(t)\big) \\ &= Q(q)\big(u_k + L(q)\underbrace{F(q)e_k(t)}_{e_k^F(t)}\big) \end{aligned} \tag{3.31}$$

where $e_k^F(t)$ *is the output signal from the feedback controller* F *in Figure 3.1.*

□

Consider a controller of PD-type. We see that by choosing the output from the controller as the error signal, e_k^F, we will try to minimize a linear combination of the $e_k(t) = r(t) - y_k(t)$ and $\dot{e}_k(t)$. The convergence criteria will be the same as if we apply the control signal, $u_k(t)$, at the input of the controller F. Of course, considering $e_k(t) = r(t) - y_k(t)$ as error signal.

If the controller has internal states, e.g. the PID-type controller, $e_k^F(t)$ can not be used as error signal. When the output from the PID controller is different from zero, the current error, $e_k(t) = r(t) - y(t)$, does not have to be different from zero. A practical case when this applies is in the control of a robot arm when the arm is moving in the gravity field. In order to keep the position error equal to zero, when the robot stands still, the controller will have to compensate for the gravity disturbance. Minimizing this compensation will, obviously, not lead to a good control strategy.

Disturbance effects

We will now continue by analyzing at the case when we have disturbances acting on the system, and see how this effect the ILCF system. Recall the system equation (3.8),

$$y_k(t) = T_r(q)r(t) + T_u(q)u_k(t) + T_\nu(q)\nu_k(t)$$

Assume that the disturbance can be divided into two terms, the first being a repetitive disturbance and the second acting as a random disturbance. The system equation can now be written in the following form,

$$y_k(t) = T_r(q)r(t) + T_u(q)u_k(t) + T_d(q)d_k(t) + T_n(q)n_k(t) \tag{3.32}$$

where $T_d(q)$ is the filter for the repetitive disturbance and $T_n(q)$ is the filter for the random disturbance. These filters can include parts of the system transfer function. In Figure 3.1, for example, the repetitive disturbance is assumed to be applied at the input of the plant, $G(q)$. The transfer function $T_d(q)$ will therefore include the transfer function of the plant, $G(q)$, and the feedback controller $F(q)$. We get

$$T_d(q) = F^{-1}(q)G_C(q) \tag{3.33}$$

i.e. the same as $T_u(q)$.

Example 3.4 (Disturbances in a Physical System)
Repetitive and random disturbances occur in physical systems mainly because of the fact that we cannot model all the dynamics. However, it is

often possible to tell where the disturbance comes from. Often measurement disturbances are assumed to be of a random character, although also a repetitive disturbance can be present also on the measurements. In the resolvers used in robot applications a repetitive disturbance, called resolver-ripple, is present. The frequency of this disturbance depends of the velocity of the rotation of the motor axis. The load disturbances in the robot system, however, are usually assumed to be of repetitive nature. The repetitive load disturbances come from ripple in the torque applied to the joints (caused by the AC motors), disturbances caused by the transmission, and load disturbances caused by the gravity acting on the robot.

□

The discussion here will not explicitly cover the load disturbances and the measurement disturbances but instead look at repetitive and random disturbances. We will start by considering the repetitive disturbances.

Repetitive disturbances

A number of observations can be made using (3.32). If we first consider the case when $Q \equiv 1$ and $n_k(t) \equiv 0$; this means that there are no random disturbances acting on the system. The update equation for the error becomes,

$$e_{k+1}(t) = (I - L(q)T_u(q))e_k(t) + T_d(q)(d_k(t) - d_{k+1}(t)) \tag{3.34}$$

see also (3.11). We can see that the disturbances contribute to the error equation by their differences between the iterations. If a disturbance is of repetitive nature in the sense that the disturbance signals $d_k(t) = d_{k+1}(t)$ for all k, the contribution to the error difference equation is zero.

This assumption is likely for the load disturbances that occur in a robot due to gravitational forces, since the gravitational forces can be expected to have the same time dependence for each iteration.

Let us also consider the situation with $Q \neq 1$, neglect random disturbances and assume that $d_k(t) = d(t)$ for all k. We have to redo some of the calculations leading to the result in (3.11) since they are not valid in the MIMO case. We have that

$$\begin{aligned} u_{k+1}(t) &= Q(q)(u_k(t) + L(q)e_k(t)) = Q(q)(u_k(t) + L(q)(r(t) - y_k(t))) \\ &= Q(q)(I - L(q)T_u(q))u_k(t) \\ &\qquad + Q(q)L(q)(I - T_r(q))r(t) - Q(q)L(q)T_d(q)d(t) \end{aligned} \tag{3.35}$$

When the stability criterion of 3.12 is fulfilled the equilibrium trajectory in (3.35) can be calculated using Corollary 3.1. This gives us

$$u_\infty(t) = (I - Q(I - LT_u))^{-1} QL\big((I - T_r)r(t) + T_d d(t)\big) \tag{3.36}$$

Inserting this in (3.20), assuming $u_0(t) \equiv 0$ gives us,

$$e_\infty(t) = \ldots + \big(I - T_u(I - Q(I - LT_u))^{-1} QL\big) T_d d(t) \tag{3.37}$$

where the first term, denoted $\ldots$, represents $e_\infty(t)$ in (3.20). We can now illustrate the result in an example.

Example 3.5 *Consider the SISO system depicted in Figure 3.1. If we use the definitions of the different transfer operators that we have made in Example 3.3 and* (3.33) *we get (omitting the dependence of q in all the filters),*

$$\begin{aligned} e_{d\infty}(t) &= (1 - F^{-1}G_C(1 - Q(1 - LF^{-1}G_C))^{-1}QL)F^{-1}G_C d(t) \\ &= \frac{1 - Q}{1 - Q(1 - LF^{-1}G_C)} F^{-1} G_C d(t) \end{aligned} \tag{3.38}$$

where $e_d(t)$ *means that we study the part of the error that originates from* $d(t)$, *here assumed to be a repetitive load disturbance. We see that by choosing* $Q \equiv 1$ *we will be able to remove the error completely. By choosing* L *as* $(F^{-1}G_C)^{-1}$, *i.e.* T_u^{-1} *(cf. Remark 3.3), the resulting error will be* $(1 - Q)T_d d(t)$. *Considering this expression in the frequency domain we can draw the conclusion that if* Q *is a low-pass filter,* $1 - Q$ *will be a high-pass filter. If the repetitive disturbance now has the main part of its energy in the low frequency band the disturbance rejection, also with the filter* Q, *will still be good.*

□

We can now study the effect of random disturbances on the ILCF system.

Random disturbances

We will now consider the case when we have a random disturbance acting on the system. The case that we will study is when the disturbances are assumed to be filtered white noise.

Since we are considering linear systems we can look at the result when we apply ILC on a system having $r(t) \equiv 0$ and where we do not have any repetitive disturbance, i.e. $d(t) \equiv 0$. This means that the system description in (3.8) simplifies to

$$y_k(t) = T_u(q)u_k(t) + T_n(q)n_k(t) \tag{3.39}$$

and because the reference signal equals zero we will have the following error,

$$e_k(t) = r(t) - y_k(t) = -y_k(t) \tag{3.40}$$

Using the ILC in (3.9) gives us the following expression for the updating of the control signal

$$\begin{aligned} u_{k+1}(t) &= Q(q)\big(u_k(t) - L(q)y_k(t)\big) \\ &= Q(q)\big(I - L(q)T_u(q)\big)u_k(t) + Q(q)L(q)T_n(q)n_k(t) \end{aligned} \tag{3.41}$$

where $u_k(t)$ and $n_k(t)$ are uncorrelated. This can now be written using the spectrum formulation

$$\Phi_{u_{k+1}}(\omega) = F_1(e^{i\omega t_s})\Phi_{u_k}(\omega)F_1^*(e^{i\omega t_s}) + F_2(e^{i\omega t_s})\Phi_{n_k}(\omega)F_2^*(e^{i\omega t_s}) \tag{3.42}$$

where $F_1(q) = Q(q)\big(I - L(q)T_u(q)\big)$, $F_2(q) = Q(q)L(q)T_n(q)$, and F^* is the adjoint matrix. We can now formulate the following lemma.

Lemma 3.2
Given the system description in (3.39). Applying the ILC in (3.9) will lead to a stable system if

$$\|F_1(e^{i\omega t_s})\|_\infty < 1 \tag{3.43}$$

□

Proof. *For the proof see Appendix 3.B.*

■

Since $F_1(q) = Q(q)\big(I - L(q)T_u(q)\big)$, this leads to the same criterion as in the nominal case, Theorem 3.2. We can now formulate the following corollary.

Corollary 3.3
It follows from Lemma 3.2 that if the condition is fulfilled the spectra of the control signal will converge to the solution of the following matrix equation,

$$\Phi_{u_\infty}(\omega) = F_1(e^{i\omega t_s})\Phi_{u_\infty}(\omega)F_1^*(e^{i\omega t_s}) + F_2(e^{i\omega t_s})\Phi_n(\omega)F_2^*(e^{i\omega t_s}) \tag{3.44}$$

□

Proof. *Follows from (3.42) and the assumption on convergence.*

■

An explicit solution to (3.44) can be found using the Kronecker products [Kai80]. The solution can be written as

$$(I - F_1(e^{i\omega t_s}) \otimes \bar{F}_1(e^{i\omega t_s}))\mathrm{col}(\Phi_{u_\infty}(\omega)) = \mathrm{col}(\Phi_N(\omega)) \tag{3.45}$$

where $\Phi_N(\omega) = F_2(e^{i\omega t_s})\Phi_n(\omega)F_2^*(e^{i\omega t_s})$. The function col$(\cdot)$ denotes the vector formed by taking the rows of the matrix and put them after each other and finally make the transpose. The Kronecker product is defined as:

Definition 3.1 (Kronecker product)
Let $A \in \mathbb{C}^{m\times n}$ and $B \in C^{j\times l}$, the the Kronecker product of A and B is defined as

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{bmatrix} \in C^{mj\times nl} \tag{3.46}$$

We can now give an example of the result for random disturbances in the SISO case.

Example 3.6 *Consider again the system in Figure 3.1. Let us assume that $r(t) \equiv 0$ and that we have only the random disturbance $n_k(t)$ acting on the system. Using the transfer operators defined in Example 3.3 and*

$$T_n(q) = \frac{1}{1 + F(q)G(q)} \tag{3.47}$$

we get the criteria for convergence from (3.43). If we assume the criteria to be fulfilled we get from Corollary 3.3 that the asymptotic spectrum for u_k is given by

$$\Phi_{u_\infty}(\omega) = \frac{|Q(e^{i\omega t_s})L(e^{i\omega t_s})T_n(e^{i\omega t_s})|^2}{1 - |F_1(e^{i\omega t_s})|^2}\Phi_N(\omega) \tag{3.48}$$

where $F_1(q) = Q(q)(1 - L(q)T_u(q))$. The spectrum for the disturbance is assumed to be $\Phi_N(\omega)$. From expression (3.48) we can now see that, if the random disturbance have the main part of its energy at high frequencies we can reduce the impact of the random disturbance on the asymptotic spectrum by choosing Q as a low pass filter.

□

In Example 3.5 we saw that if the energy of the repetitive disturbance is mainly in the low frequency band we can get a good disturbance rejection by choosing Q as a low pass filter. We have now seen in the example above that for the random disturbance case the choice of Q as a low pass filter will give a good disturbance rejection if the random disturbance has the main part of its energy in the high frequency band.

We will now consider the general linear ILC updating formula applied to the system in (3.8).

3.2.2 General Linear ILC

Finding results for the general linear ILC described in Section 2.5 are still a research topic but we will here give an extension of the stability criterion from the previous section to the general case.

Consider the following general linear ILC ,

$$u_k(t) = \frac{P_B(q_k, q_t)}{P_A(q_k, q_t)} e_k(t), \;\; k = 0, 1, 2, \ldots \quad t \in [0, t_f] \tag{3.49}$$

with the operators q_k and q_t defined by (2.52). The functions $P_A(\cdot, \cdot)$ and $P_B(\cdot, \cdot)$ are both polynomial functions in q_k and q_t. The idea of writing the polynomials P_A and P_B as polynomials in q_k with coefficients being polynomials in q_t is presented in Example 2.4. We will now use this approach to analyze an ILC given by

$$u_k(t) = \frac{H_2(q_t)q_k + H_3(q_t)}{q_k - H_1(q_t)} e_{k-1}(t) \tag{3.50}$$

By using the definition of $e_k(t) = r(t) - y_k(t)$ and applying the q_k operator on all the terms we get,

$$u_{k+1} - H_1(q_t)u_k(t) = H_2(q_t)(r(t) - y_k(t)) + H_3(q_t)(r(t) - y_{k-1}(t)) \tag{3.51}$$

From (3.8) we have in the nominal case (without considering the disturbances) that $y_k(t) = T_r(q_t) + T_u(q_t)u_k(t)$. Using this (3.51) can be written,

$$\begin{aligned} u_{k+1}(t) = {} & (H_1(q_t) - H_2(q_t)T_u(q_t))u_k(t) - H_3(q_t)T_u(q_t)u_{k-1}(t) \\ & + (H_2(q_t) + H_3(q_t))(1 - T_r(q_t))r(t) \end{aligned} \tag{3.52}$$

We can now use the same technique as in Example 3.1 to write (3.52) in a compact form,

$$v_{k+1}(t) = H(q_t)v_k(t) + H_r(q_t)r(t) \tag{3.53}$$

where,

$$\upsilon_k(t) = \begin{bmatrix} u_k(t) \\ u_{k-1}(t) \end{bmatrix} \tag{3.54a}$$

$$H(q_t) = \begin{bmatrix} H_1(q_t) - H_2(q_t)T_u(q_t) & H_3(q_t)T_u(q_t) \\ 1 & 0 \end{bmatrix} \tag{3.54b}$$

$$H_r(q_t) = \begin{bmatrix} (H_2(q_t) + H_3(q_t))(1 - T_r(q_t)) \\ 0 \end{bmatrix} \tag{3.54c}$$

The criterion for global asymptotic stability of the iterative system given by (3.53) can be formulated based on Theorem 3.1,

$$\left\| \begin{bmatrix} H_1(e^{i\omega t_s}) - H_2(e^{i\omega t_s})T_u(e^{i\omega t_s}) & H_3(e^{i\omega t_s})T_u(e^{i\omega t_s}) \\ 1 & 0 \end{bmatrix} \right\|_\infty < 1 \tag{3.55}$$

If the criterion if fulfilled the asymptotic value of the control signal can be calculated using Corollary 3.1. Using (3.53) we can write down the expression for the asymptotic state $\upsilon_\infty(t)$,

$$\upsilon_\infty(t) = H^{-1}(q_t)H_r(q_t)r(t) \tag{3.56}$$

We can now conclude this section.

The idea here is to show that also the general approach to linear ILC, using higher order ILCs, can be analyzed in the framework presented in the thesis. The properties of the general linear ILC applied to linear systems and design rules for this type of updating equation is still an open research area and it will be a topic for future research.

3.2.3 Nonlinear Systems

In this section some results on applying ILC to non-linear systems will be presented. The aim is to give a review of some results from the literature and we do not have the intention to give a complete theory for ILC applied to non-linear system. A very good overview with many references are given in a monograph by Moore [Moo93], in a survey article by the same author [Moo98], and in a book edited by Bien and Xu [Be98]. We refer to these publications for a more extensive treatment of the subject.

For nonlinear systems most of the work has been concentrated to the special structures of systems that is found in robotics. The problem of controlling a robot structure using an ILC has been covered in, e.g., [AKM84a, BCG88, Hor93]. The effect of applying ILC to a system with nonlinear friction is discussed in, e.g. [Liu94, GN97c]. The work presented in [CGW98] gives an analysis of a high order ILC applied to an uncertain nonlinear system.

We will now review a result originally published by Hauser [Hau87], later extended by Heinzinger et al. [HFPM92]. Assume that we want to control a system described by the following non-linear state space model,

$$\dot{x}_k(t) = f(x_k(t), t) + B(x_k(t), t)u_k(t) + \nu_k(t) \tag{3.57a}$$

$$y_k(t) = g(x_k(t), t) \tag{3.57b}$$

where $t \in [0, t_f]$, $x_k(t) \in \mathbb{R}^n$, $u_k(t) \in \mathbb{R}^m$, $y(t) \in \mathbb{R}^p$, and $\nu_k(t) \in \mathbb{R}^n$. Consider the following ILC updating formula

$$u_{k+1}(t) = \rho u_k(t) + (1-\rho)u_0(t) + L(y_k(t), t)\frac{d}{dt}e_k(t) \tag{3.58}$$

where $0 \le \rho < 1$, $e_k(t) = r(t) - y_k(t)$ and $L : \mathbb{R}^p \times [0, t_f] \to \mathbb{R}^{m \times p}$, L bounded. We recognize the first order ILC described in Section 2.5. With some smoothness assumptions on the functions $f(\cdot,\cdot)$, $B(\cdot,\cdot)$, $g(\cdot,\cdot)$, and $L(\cdot,\cdot)$ we can now give a theorem for stability.

Theorem 3.3 (Stability Criterion, Nominal Stability)
Given a non-linear system as in (3.57) *but with* $\nu_k(t) = 0$, $\forall k$ *and* $x_k(0) = x_r(0)$. *Applying the ILC in* (3.58) *will give a stable ILCF system if*

$$\|\rho I - L(g(x,t),t)g_x(x,t)B(x,t)\| \le \gamma < 1 \tag{3.59}$$

Moreover, under the assumption that the desired trajectory is achievable and $\rho = 1$, *the error,* $e_k(t)$, *will converge uniformly to zero on* $[0, t_f]$, *cf. Lemma 3.1.*

□

Proof. *The main idea of the proof is to show that* $u_r(t) - u_k(t)$ *is decreasing (in some norm) with increasing* k, *where* $u_r(t)$ *is the input that gives the desired output signal,* $y(t) = r(t)$. *The technique is similar to the technique used to prove stability in the linear case, see Appendix 3.A. For a complete proof of the theorem see [Hau87, HFPM92].*

■

In [HFPM92] the theorem is extended to the case when we have a state disturbance acting on the system, $\nu_k(t) \neq 0$, and the error in the initial condition is not zero, i.e. $x_k(0) \neq x_0(0)$. The result is a theorem with the same criterion, (3.59), but with some extra assumptions on boundedness of $\nu_k(t)$ and the initial state error.

3.3 The Disturbance Rejection Approach

In the previous chapter we defined an approach to ILC that we called the *disturbance rejection approach*. We will now analyze this approach when the system G_D in Figure 3.2 is a linear system.

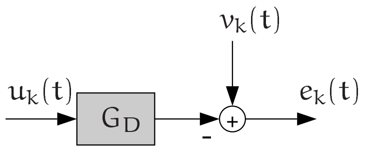


Figure 3.2 *The disturbance rejection problem displayed as a block diagram.*

Consider the system depicted in Figure 3.2. In the problem formulation of Definition 2.10 the disturbance, $\nu_k(t)$, was defined as

$$\nu_k(t) = d(t) + n_k(t) \tag{3.60}$$

where $d(t)$ is a repetitive disturbance and $n_k(t)$ is of random character. The problem is to identify the repetitive disturbance $d(t)$ and compensate this disturbance such that the energy of the output is minimized.

3.3.1 Connections to Adaptive Control

There is a class of adaptive controllers that work with the same goal as we formulated in the disturbance rejection problem, namely to reduce the variance of the output of the system. The adaptive controllers are called *minimum-variance* controllers or *moving average* controllers [AW95]. These types of controllers do not use the fact that the system we control is an iterative system. Only traditional feedback from the current iteration is used. Ideas from adaptive control can however be very useful in the ILC methods.

Comparison between adaptive control and ILC and also some approaches to adaptive ILC updating formulas can be found in , e.g., [Moo93, GS92, Hor93, KL91, SG92, SM98].

3.3.2 Some Stability Results

We are now going to analyze the stability of the ILCF system when we apply ILC. First we will consider the case of first order ILC and then high order ILC will be covered.

First order ILC

The system equation describing the system depicted in Figure 3.2 is given by

$$e_k(t) = -G_D(q)u_k(t) + \nu_k(t) \tag{3.61}$$

with $\nu_k(t)$ as in (3.60). Using (3.9) and (3.61) we get

$$u_{k+1}(t) = Q(q)\big(I - L(q)G_D(q)\big)u_k(t) + Q(q)L(q)\nu_k(t) \tag{3.62}$$

Now we can give a criterion for the nominal stability.

Theorem 3.4 (Stability criterion)
Given a system on the form depicted in Figure 3.2, described as a discrete time LTI system $G_D(q)$ over the time interval $[0, t_f]$. Using the linear first order ILC in (3.9) we get the following stability criterion for global asymptotic stability

$$\|Q(e^{i\omega t_s})\big(I - L(e^{i\omega t_s})G_D(e^{i\omega t_s})\big)\|_\infty < 1 \tag{3.63}$$

where Q and L are the filters used in the ILC updating equation and G_D is the transfer function from $u_k(t)$ to $e_k(t)$.

□

Proof. *Just as in the proof of Theorem 3.2 we can use Theorem 3.1. Using (3.62) and the assumption that $\nu_k(t) = d(t)$, i.e. that the disturbance is the same every iteration, we arrive at*

$$u_{k+1}(t) = Q(q)(I - L(q)G_D(q))u_k(t) + Q(q)L(q)d(t) \tag{3.64}$$

From Theorem 3.1 we have the iterative system described by (3.64) is asymptotically stable if and only if

$$\|Q(e^{i\omega t_s})\big(I - L(e^{i\omega t_s})G_D(e^{i\omega t_s})\big)\|_\infty < 1 \tag{3.65}$$

where t_s is the sample time.

■

We can see some similarities with the stability criterion for the ILC tracking approach, Theorem 3.2. The transfer function from the control signal to the output, T_u, is here changed to G_D, cf. (3.12). In Remark 3.2 it is mentioned that the criterion for the stability of the tracking approach can be interpreted in a Nyquist diagram. This is also possible for this criterion. The interpretation is that the Nyquist curve for $L(e^{i\omega t_s})G_D(e^{i\omega t_s})$ should be kept within the stability region, a unit circle centered at $+1$. The stability region can be extended just as noted in Remark 3.2, using the filter Q.

Corollary 3.4
As a consequence of Theorem 3.4 we have that the control signal will converge to

$$\lim_{k\to\infty} u_k(t) = u_\infty(t) = \big(I - Q(q)(I - L(q)G_D(q))\big)^{-1}Q(q)L(q)d(t) \tag{3.66}$$

Using the result in (3.66) we can get an expression for the asymptotic error

$$\begin{aligned}\lim_{k\to\infty} e_k(t) &= e_\infty(t) = G(q)u_\infty(t) + d(t) = \\ &\Big(I - G_D(q)\big(I - Q(q)(I - L(q)G_D(q))\big)^{-1}Q(q)L(q)\Big)d(t)\end{aligned} \tag{3.67}$$

□

Proof. *If the stability criterion in Theorem 3.4 is fulfilled the control signal will converge to an equilibrium trajectory, $u_\infty(t)$. Insert this equilibrium trajectory in (3.62) and solve for $u_\infty(t)$. Using this result it is easy to arrive at (3.67).*

■

In the case when there is a random disturbance $n_k(t)$ we get the following expression for the spectrum of the control signal,

$$\Phi_{u_{k+1}}(\omega) = F_1(e^{i\omega t_s})\Phi_{u_k}(\omega)F_1^*(e^{i\omega t_s}) + \Phi_{n_k}(\omega) \tag{3.68}$$

where $F_1(q) = Q(q)(I - L(q)G_D(q))$. The following lemma gives a condition for the convergence when we only have the disturbance $n_k(t)$.

Lemma 3.3
Assume that the system G *in Figure 3.2 is described as an LTI system,* $\mathsf{G}(q)$, *and that the disturbance* $\nu_k(t) = n_k(t)$ *and that the spectrum* $\Phi_{n_k}(\omega) = \Phi_N(\omega)$. *Applying the ILC in (3.9) will give a stable ILCF system if*

$$\|Q(e^{i\omega t_s})\big(I - L(e^{i\omega t_s})G(e^{i\omega t_s})\big)\|_\infty < 1 \tag{3.69}$$

□

Proof. *The proof is equivalent to the proof of Lemma 3.2.*

■

Which is the same criterion as we had in the nominal case, Theorem 3.4. Since we have convergence if the criterion is fulfilled the following corollary can be formulated.

Corollary 3.5
If the condition in Lemma 3.3 is fulfilled the spectra of the control signal will converge to the solution of the following matrix equation,

$$\Phi_{u_\infty}(\omega) = F_1(e^{i\omega t_s})\Phi_{u_\infty}(\omega)F_1^*(e^{i\omega t_s}) + \Phi_N(\omega) \tag{3.70}$$

□

Proof. *Follows from (3.68) and the assumption of convergence. See also the proof of Corollary 3.3.*

■

3.4 Summary

This chapter contains a lot of details on the theory of stability and asymptotic behavior of the ILCF system when we use different ILC updating formulas. We are now going to summarize the major results that we have presented.

- **Tracking approach:** Using the first order ILC given by

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

on a system described by

$$y_k(t) = T_r(q)r(t) + T_u(q)u_k(t) + T_v(q)v_k(t)$$

We have asymptotic stability iff

$$\|Q(e^{i\omega t_s})(I - L(e^{i\omega t_s})T_u(e^{i\omega t_s}))\|_\infty < 1$$

The asymptotic error is given by

$$\begin{aligned} e_\infty(t) = r(t) - y_\infty(t) = \\ &\left(I - T_u(q)\left(I - Q(q)\big(I - L(q)T_u(q)\big)\right)^{-1}\right) \cdot \\ &\quad \left(Q(q)L(q)(I - T_r(q))r(t) + T_d(q)d(t)\right) - T_r(q)r(t) \end{aligned}$$

where $d(t)$ is a repetitive disturbance. The result for random disturbances is found in Corollary 3.3.

- **Disturbance rejection approach:** With the same ILC as we used in the tracking approach above and with the system described by,

$$e_k(t) = -G_D(q)u_k(t) + v_k(t)$$

we get a stability criterion for global asymptotic stability according to,

$$\|Q(e^{i\omega t_s})\big(I - L(e^{i\omega t_s})G_D(e^{i\omega t_s})\big)\|_\infty < 1$$

The asymptotic error is given by,

$$e_\infty(t) = \left(I - G_D(q)\big(I - Q(q)(I - L(q)G_D(q))\big)^{-1}Q(q)L(q)\right)d(t)$$

where $d(t)$ is repetitive. The result for random disturbances is found in Corollary 3.5.

Appendix

3.A Proof of Theorem 3.1

We want to prove asymptotic stability of the following iterative system,

$$z_{k+1}(t) = F_1(q)z_k(t) + u(t) \tag{3.71}$$

where $z_j(t) \in \mathbb{R}^n$, and $F_1(q)$ is a $n \times n$ matrix of discrete time transfer operators, see Definition 2.3.

We will proof the actual converge to a equilibrium trajectory using the frequency domain representation of (3.71)

$$Z_{k+1}(e^{i\omega t_s}) = F_1(e^{i\omega t_s})Z_k(e^{i\omega t_s}) + U(e^{i\omega t_s}) \tag{3.72}$$

where, for fixed $\omega \in [-\pi/t_s, \pi/t_s]$, $Z_j(e^{i\omega t_s}) \in \mathbb{C}^n$ and $F_1(e^{i\omega t_s}) \in \mathbb{C}^{n\times n}$. The idea is now to prove that given

$$\gamma = \|F_1(e^{i\omega t_s})\|_\infty < 1 \tag{3.73}$$

the "state" vector $Z_k(e^{i\omega t_s})$ will converge uniformly to an equilibrium vector, $Z_\infty(e^{i\omega t_s})$ for $\omega \in [-\pi/t_s, \pi/t_s]$. To do this we use the following theorem from [Rud76].

Theorem 3.5 (Uniform Convergence)
The sequence of functions $\{f_n\}$, bounded on $\mathbb{C}^n$, converges uniformly on $\mathbb{C}^n$ if and only if for every $\epsilon > 0$ there exists an integer N such that $m \geq N$, $n \geq N$, $x \in \mathbb{C}^n$ implies

$$|f_n(x) - f_m(x)| \leq \epsilon \tag{3.74}$$

□

Proof. *See, e.g., [Rud76].*

■

Now consider $Z_n(e^{i\omega t_s})$ and $Z_m(e^{i\omega t_s})$ with $m \geq N$, $n \geq N$ for some integer N. We can write the difference according to (3.74) as

$$\begin{aligned} |Z_n(e^{i\omega t_s}) - Z_m(e^{i\omega t_s})| &= |F_1(e^{i\omega t_s})Z_{n-1}(e^{i\omega t_s}) - F_1(e^{i\omega t_s})Z_{m-1}(e^{i\omega t_s})| \\ = \ldots &= \left| (F_1^n(e^{i\omega t_s}) - F_1^m(e^{i\omega t_s}))Z_0(e^{i\omega t_s})\right| \end{aligned} \tag{3.75}$$

we can use the fact that $m \geq N$, $n \geq N$,

$$\begin{aligned} &\left| (F_1^n(e^{i\omega t_s}) - F_1^m(e^{i\omega t_s}))Z_0(e^{i\omega t_s})\right| \leq \\ &\qquad \|F_1(e^{i\omega t_s})\|_\infty^N \left| (F_1^{n-N}(e^{i\omega t_s}) - F_1^{m-N}(e^{i\omega t_s}))Z_0(e^{i\omega t_s})\right| \\ &\qquad = \gamma^N \cdot K \end{aligned} \tag{3.76}$$

where γ is defined by (3.73) and

$$K = \left| (F_1^{n-N}(e^{i\omega t_s}) - F_1^{m-N}(e^{i\omega t_s}))Z_0(e^{i\omega t_s})\right| < \infty \tag{3.77}$$

an M is bounded since it is a difference of two bounded functions. From 3.73 we have that $\gamma < 1$, hence, we can always find N such that $\gamma^N \cdot K < \epsilon$. This concludes the proof on uniform stability for the function $Z_k(e^{i\omega t_s})$ to $Z_\infty(e^{i\omega t_s})$. The equilibrium trajectory can be calculated from (3.72) by letting $k \to \infty$ and solving the algebraic equation

$$Z_\infty(e^{i\omega t_s}) = F_1(e^{i\omega t_s})Z_\infty(e^{i\omega t_s}) + U(e^{i\omega t_s}) \tag{3.78}$$

giving

$$Z_\infty(e^{i\omega t_s}) = \left(I - F_1(e^{i\omega t_s})\right)^{-1}U(e^{i\omega t_s}) \tag{3.79}$$

We have now proved asymptotic stability if $\|F_1(e^{i\omega t_s})\|_\infty < 1$. To compete the proof we will show that we will not have asymptotic stability if $\|F_1(e^{i\omega t_s})\|_\infty \geq 1$. If we start by assuming that $\gamma = \|F_1(e^{i\omega t_s})\|_\infty > 1$. Consider the iterative system in (3.72) but with $U(e^{i\omega t_s}) \equiv 0$,

$$Z_{k+1}(e^{i\omega t_s}) = F_1(e^{i\omega t_s})Z_k(e^{i\omega t_s}) = F_1^k(e^{i\omega t_s})Z_0(e^{i\omega t_s}) \tag{3.80}$$

By using that $F_1(e^{i\omega t_s})$ in the discrete time case is a periodic function of ω there will be an $\omega \in [0, \pi/t_s]$ where the maximum singular value, $\bar{\sigma}$, of $F(e^{i\omega t_s})$ actually takes the value γ. We denote this angular frequency ω_γ. If we choose $Z_0(e^{i\omega_\gamma t_s})$ as the *right singular vector*, $v_{\bar{\sigma}}$,corresponding to the maximum singular value we get,

$$|Z_{k+1}(e^{i\omega_\gamma t_s})| = |F_1^k(e^{i\omega_\gamma t_s})v_{\bar{\sigma}}| = \bar{\sigma}^k|u_{\bar{\sigma}}| = \gamma^k|u_{\bar{\sigma}}| \tag{3.81}$$

where $u_{\bar{\sigma}}$ is the *left singular vector*, corresponding $\bar{\sigma}$. Since $\gamma \geq 1$ we will not have convergence to zero for this choice of initial function of $Z_0(e^{i\omega t_s})$. Note that for $\gamma > 1$ we will not only have that the function $Z_k(e^{i\omega t_s})$ will grow unlimited in one point ω_γ. The singular values of $F_1(e^{i\omega t_s})$ are continuous functions of ω and, hence, there will be a region around ω_γ where we the function will grow. When $\gamma = 1$ the system will have a marginally stable behavior. Having zero input to the iterative system, as in (3.80), the system state will not change, With a constant input, , equal to $v_{\bar{\sigma}}$, the function $|Z_k(e^{i\omega t_s})|$ will grow unlimited. This concludes the proof.

3.B Proof of Lemma 3.2

We have that

$$\Phi_{u_{k+1}}(\omega) = F_1(e^{i\omega t_s})\Phi_{u_k}(\omega)F_1^*(e^{i\omega t_s}) + \Phi_N(\omega) \tag{3.82}$$

We can use Theorem 3.5 to prove uniform convergence to $\Phi_{u_\infty}(\omega)$. Consider $\Phi_{u_n}(\omega)$ and $\Phi_{u_m}(\omega)$ with $m \geq N$, $n \geq N$ for some integer N. We can write the difference according to (3.74) as

$$\begin{aligned}
&\|\Phi_{u_n}(\omega) - \Phi_{u_m}(\omega)\|_\infty \\
&= \|(F_1^n(e^{i\omega t_s}) - F_1^m(e^{i\omega t_s}))\Phi_{u_0}(\omega)(F_1^{*(n)}(e^{i\omega t_s}) - F_1^{*(m)}(e^{i\omega t_s}))\|_\infty \\
&\leq \left(\|F_1(e^{i\omega t_s})\|_\infty^n + \|F_1(e^{i\omega t_s})\|_\infty^m\right)\|\Phi_{u_0}(\omega)\|_\infty \cdot \\
&\qquad \left(\|F_1^*(e^{i\omega t_s})\|_\infty^n + \|F_1^*(e^{i\omega t_s})\|_\infty^m\right) \\
&\leq 4\gamma^{\min(m,n)}\|\Phi_{u_0}(\omega)\|_\infty
\end{aligned} \tag{3.83}$$

with $\|\Phi_{u_0}(\omega)\|_\infty < K < \infty$ we can make (3.83) arbitrarily small by choosing N large enough. We conclude that $\|F_1(e^{i\omega t_s})\|_\infty < 1$ is a sufficient criterion for convergence to $\Phi_{u_\infty}(\omega)$.

4

Synthesis

In this chapter an introduction to synthesis of the ILC updating formula is presented. The base for the design algorithms is the criterion for global asymptotic stability, Theorem 3.2, given in the previous chapter. We will present a heuristic approach that we also have used in the experiments in Chapter 7 and two model based approaches, one of them based on H_∞ control theory methods. Some various approaches found in the ILC literature will also be briefly discussed. In this chapter we will only consider first order ILC applied to the type of LTI systems discussed in Section 3.2.1.

4.1 Introduction

The synthesis problem is the problem of finding the ILC given a system, T, to control, cf. Section 3.2. In the ILC literature the synthesis of ILC updating formulas has not been addressed very much. Some work has been made using H_2 and H_∞ optimal controllers. The systematic synthesis has so far only been dealing with linear systems, mostly LTI systems. Within H_2 optimal synthesis Prof. Owens' group have made some contributions [AO94, AOR95b, AOR95a, AOR97], an early contribution is also by Togai and Yamano [TY85]. ILC synthesis based on H_∞ methods have been covered in contributions by, e.g., Park and Hesketh [PH93], Liang and Looze [LL93], and de Roover [dR96a, dR96b]. The work in [PH93] does not involve the

Q filter in the ILC, but this is included in the work of de Roover, and also to some extent in [LL93]. All of the work referenced here is on first order ILC, synthesis methods for higher order ILC is still an open field.

4.2 Algorithms for ILC Synthesis

We will now give a review of some algorithms showing ways to choose the ILC in a systematic way. The algorithms produce first order ILCs on the following form,

$$u_{k+1}(t) = Q(q)(u_k(t) + L(q)e_k(t)) \tag{4.1}$$

where the result of the synthesis algorithms is the filters Q and L. The system model, T, is assumed to be an LTI system. We will start by explaining the method used in some of the ILC experiments in Chapter 7.

4.2.1 A Heuristic Approach

The method used in the experiments is a heuristic approach to the synthesis of the filters Q and L in the ILC updating formula. We assume the system T to be a SISO servo control system. The algorithm can be formulated like this:

Algorithm 4.1 (A Heuristic Design Procedure)

1. *Choose the Q filter as a low-pass filter with cut-off frequency at the band-width of the system.*
2. *Let $L(q) = \kappa q^{\delta}$ and let $\kappa = 0.8$, $\delta = 0$.*
3. *Run an experiment with the chosen δ. Abort the experiment if $\|e_k(t)\|_2$ starts growing, i.e. if $\|e_k(t)\|_2 > \|e_{k-1}(t)\|_2$.*
4. *If the result from the experiment is not satisfying, increase δ (by one or two) and go back to 3.*

The assumption we have on the system T is that it performs as a closed loop system, i.e. basically as a low pass filter. The value of κ can be adjusted to get a faster convergence, cf Chapter 7. We will now describe two model based approaches.

4.2.2 Model Based Approaches

In the analysis of the stability of the ILCF system we have presented some ideas to the design of a good ILC updating formula. The ideas stem from the observations made when considering the stability criteria, cf. Remark 3.2 and Remark 3.3. We will now consider a synthesis approach based on these ideas.

A heuristic model-based approach

The design procedure presented in this section has also been discussed in [GN97a, GN97c, NG97]. The idea is similar to the heuristic approach by de Roover [dR96a, dR96b] but de Roover uses a model matching approach based on H_∞ methods while we take an algebraic approach. The transfer function $T_u(e^{i\omega t_s})$ is the transfer function from the ILC control input, $u_k(t)$, to the output of the system $y_k(t)$, cf. Section 3.2.1.

Algorithm 4.2 (A Heuristic Model-Based Design Procedure)

1. *Choose a high pass filter* $H_B(q)$ *with cut-off frequency* ω_m. *Up to* ω_m *the model* $T_u(q)$ *is assumed to be a good model of the true system.*
2. *Calculate* L *by* $L(q) = T_u^{-1}(q)(1 - H_B(q))$.
   By using this expression for L *we can shape the convergence rate with the filter* H_B. $H_B(q) = 1 - L(q)T_u(q)$, *which we recognize from the stability criterion. The filter* H_B, *in this way, decides the convergence rate that we want for different frequencies.*
3. *Choose the filter* $Q(e^{i\omega t_s})$ *to be a low-pass filter with cut-off frequency near the frequency* ω_m *where the model of the system starts to be uncertain.*

The algorithm is based on the interpretation found already in Example 2.1. We saw that the convergence criteria can be interpreted in a Nyquist diagram. The requirement for stability becomes that the Nyquist curve for $L(e^{i\omega t_s})T_u(e^{i\omega t_s})$ should be held within a circle with the radius 1 and center in $+1$. This stability region can be extended using the Q filter and according to the algorithm this is done where the model of the system has uncertainties.

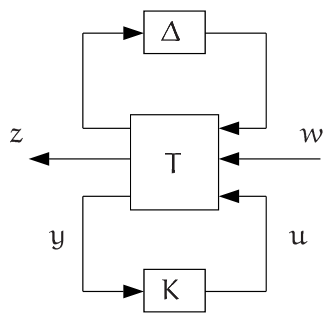
Remark 4.1
The L *that we get from the algorithm can be simplified by using some model reduction technique, like balanced truncation [ZDG96].*

We will now look at a method that uses H_∞ control theory to systematically design the filters Q and L.

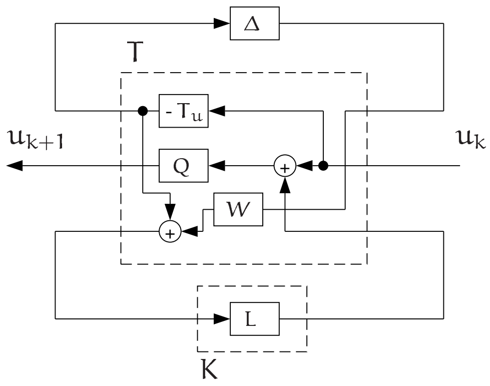
An approach based on μ-synthesis

The design process for the filter Q and L presented here is a review of the algorithm in [dR96a, dR96b]. We have changed the notation to fit in the framework used in the thesis. The algorithm is based on optimal H_∞ control theory [ZDG96].

Before formulating the algorithm we will discuss some prerequisites. We will use the standard plant format, depicted in Figure 4.1a. Within this



a. Standard plant.



b. Standard plant interpretation for the ILC synthesis problem.

Figure 4.1 *Standard plant description for μ-synthesis.*

framework tools (for example in MATLAB™ [BDG+94]) are available for computing a stabilizing K that minimizes $\|T_{zw}\|_\infty$. We assume that the real system can be described by a nominal model T_u^0 with an output multiplicative uncertainty. The uncertainty is described in the frequency domain by a stable and inversely stable weighting function $W(e^{i\omega t_s})$. We say the true system is found in the following set of systems,

$$T_u(q) = \{(I + W(q)\Delta(q))T_u^0(q) \mid \|\Delta(q)\|_\infty < 1\} \tag{4.2}$$

We can now formulate the ILC design problem in the μ-synthesis framework using the representation shown in Figure 4.1b. We have that

$$K = L(q) \quad \text{and} \quad T = \begin{bmatrix} 0 & -T_u(q) & 0 \\ 0 & Q(q) & Q(q) \\ W(q) & -T_u(q) & 0 \end{bmatrix} \tag{4.3}$$

We can now solve the optimization problem using μ-synthesis. The solution is based on the *D-K iteration* technique, we refer to [ZDG96] for a thorough discussion. Important to note is that the D-K iterations do not necessarily need to converge to the optimum. Many applications has, although, shown that the technique works [BDG$^+$94]. The proposed algorithm is now presented.

Algorithm 4.3 (A μ-synthesis based approach)

1. *Model the transfer function* T_u *as a nominal model together with an upper bound on the model uncertainty, according to* (4.2).
2. *Choose the filter* $Q(e^{i\omega t_s})$ *to be a low-pass weighting filter with cut-off frequency* ω_c.
3. *For given* T_u *and* Q *find an* L *that minimizes* $\|T_{zw}\|_\infty$ *using* μ*-synthesis [BDG$^+$94].*
4. *If an* L *can be found such that* $\|T_{zw}\| < 1$, *increase the bandwidth of the filter* Q *and again perform step 3; else decrease* ω_c. *Perform step 2 and 3 repeatedly until the maximum of* ω_c *is reached.*

The algorithm has been tested in [dR96b] on an $xy\phi$-stage, a type of high accuracy positioning mechanism. The model with uncertainty, according to (4.2), 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 4.2, and one using the robust approach in Algorithm 4.3. Not surprisingly, the performance is better with the ILC based on the nominal model compared to the robust ILC. We refer to [dR96a, dR96b] and the references there for more details. Applying it to the experimental setup described in Chapter 5 is a topic for future research.

Various approaches

In [MM94] Manabe and Miyazaki, a method to update the ILC control 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 update of the learning control signal. This corresponds to the ideas presented in Algorithm 4.2 were the L filter is chosen to be the inverse of the transfer function from u_k to y_k.

Some approaches based on ideas from the unconstrained optimization and minimization is presented by Togai and Yamano [TY85]. Given a discrete time state space representation of the system,

$$x_k(t+1) = Ax_k(t) + Bu_k(t) + w_k(t) \tag{4.4a}$$
$$y_k(t) = Cx_k(t) + n_k(t) \tag{4.4b}$$

using the following definition of the error and the updating equation,

$$e_k(t) = x_r(t) - x_k(t) \tag{4.5a}$$
$$u_{k+1}(t) = u_k(t) + Le_k(t+1) \tag{4.5b}$$

and having a criterion in the form

$$J = \frac{1}{2}e_k^T(t+1)e_k(t+1) = \frac{1}{2}|e_k(t+1)|^2 \tag{4.6}$$

Using different gradient methods to minimize the criterion they find optimal choices of L, interested readers are referred to [TY85]. For a general discussion on unconstrained minimization see, e.g., the book by Dennis and Schnabel, [JDS83].

4.3 Summary

A number of approaches to the synthesis of first order ILC applied to linear time invariant systems are presented in the chapter. There are not so many available design approaches to the synthesis of high order ILC or ILC applied to non-linear systems. The optimization based methods, presented in the last part of Section 4.2.2 can however be extended to linear time varying (LTV) systems. Further work on the synthesis of ILC is needed.

5

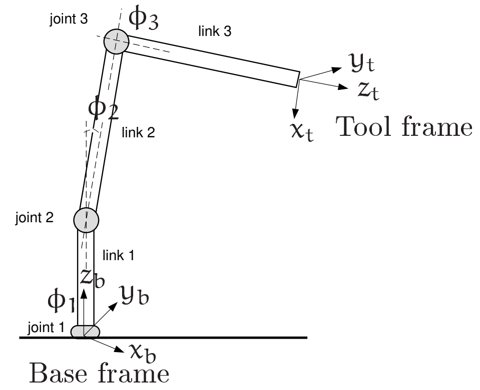
Application, Industrial Robot Control

So far the discussion has been mostly related to theoretical aspects of the method ILC. We will now examine an application where ILC has been successfully applied, namely robot control. In this chapter the robot control and the dynamical equations of the robot will be presented. We will also discuss the specific robot that will be used in the experiments.

5.1 Robot Dynamics and Control

Robot control appeared as a research field in the late 60's and the beginning of the 70's. Robots had been used industrially before that but the mathematical tools used for analysis and design of the control systems were not fully developed until the 70's. Since the term robot is used to describe everything from micro robots, like mechanical bugs, to huge autonomous trucks moving around in mines, we need to define our interpretation of a robot. In Figure 5.1a the type of robot that will be considered in the thesis is shown. This is a classical industrial manipulator, in this case an ABB IRB 1400. In this section some of the important properties of this type of robot will be examined. We will start by looking at the kinematics, i.e. the geometrical description of the manipulator. Many survey articles and books have been written on the kinematics, dynamics and control of robots, e.g., [SV89, SLA92, Cra88]

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



b. Joint space and task space for a three degree of freedom robot.

Figure 5.1 *Two different views of the robot.*

5.1.1 Kinematics

Kinematics of a robot refers to the geometric relationship between the motion of the robot in joint space and the motion of tool frame relative to the base frame of the robot. In Figure 5.1b the two frames are displayed and an idea is given of the relation between the two frames. The joint variables, denoted ϕ_2 and ϕ_3, are relative angles between the links 1-2 and 2-3, respectively, cf. Figure 5.1b. The rotation of the robot around the base frame is denoted ϕ_1. 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. Additional cables put on the robot to serve for example a spot welding device will also reduce the movements drastically. The mechanical limits are for instance, in the ABB IRB1400, $\phi_1 \in [-0.94\pi, 0.94\pi]$ while $\phi_6 \in [-1.67\pi, 1.67\pi]$. In some of the models from ABB axis 6 can actually move without any limitation.

A realization of the vector ϕ is called a *configuration* of the robot. Note that, for a *six degree-of-freedom (DOF)* robot, many 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 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 needed when going 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. We will denote this representation 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.

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 movement on the robot. The *forward kinematic problem* is to determine the mapping

$$X_0 = \begin{bmatrix} x(\phi) \\ r(\phi) \end{bmatrix} = f_0(\phi) \tag{5.1}$$

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 5.1b, the forward kinematic problem has a unique solution.

Velocity kinematics

The *velocity kinematics* define the relationship between the joint velocities, $\dot{\phi}$, and the tool frame translational and angular velocities. Similar to (5.1) we can write the velocity kinematics as

$$V = \begin{bmatrix} \nu \\ \omega \end{bmatrix} = J_0(\phi)\dot{\phi} \tag{5.2}$$

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 (5.1) 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 (5.1) and differentiate with respect to time,

$$\dot{X}_1 = \frac{\partial f_0(\phi)}{\partial \phi}\dot{\phi} = J_1(\phi)\dot{\phi} \tag{5.3}$$

where the Jacobian $J_1(\phi) = \frac{\partial}{\partial \phi} f_0(\phi)$. The points where the Jacobian looses rank are important because these points, called *singular points* 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.

High level planning and control

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

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

Going deeply into these three steps is beyond the scope of the thesis. It is important, however, for the complete understanding to have an idea of what they contain.

Motion planning

The motion planning level contains 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 taken care of 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 program. In the car industry, CAD programs are used that can simulate all the different levels of the motion control of the robots and give very accurate predictions for the cycle time. The car manufacturers use these tools to, offline, optimize the motion of the robot in order to reduce the cycle time or the production time. When the production rate is several units a 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 accelerations 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 to what can be achieved.

Often the trajectory generation is made in a flow 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 planned. Finally, the trajectory is well enough defined to be used for control, i.e. the trajectory tracking algorithms. Often the motion planning and also the trajectory generation are 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 for control of 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, these 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 will follow 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 5.2, for more details on the planning and trajectory generation problems see ,e.g., [SV89].

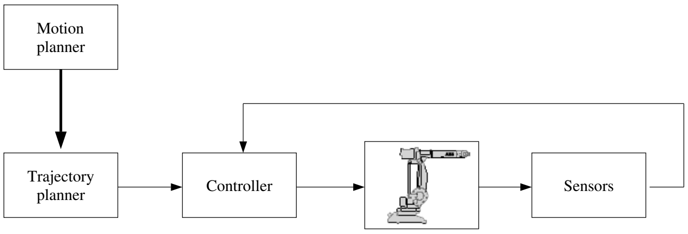


Figure 5.2 *Block diagram showing the components in the robot control problem.*

5.1.2 Dynamics

The mathematical tool that usually is adopted to describe the dynamics of a general n-link manipulator is based on the Lagrangian dynamics. In this approach the joint variables, ϕ, are considered as generalized coordinates. The *kinetic energy* of the manipulator can then be calculated,

$$K(\phi, \dot{\phi}) = \frac{1}{2}\dot{\phi}^T D(\phi)\dot{\phi} \tag{5.4}$$

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. We can now define a function,

$$\mathcal{L}(\phi, \dot{\phi}) = K(\phi, \dot{\phi}) - P(\phi) \tag{5.5}$$

called the *Lagrangian*. The dynamics of the manipulator are then described by Lagrange's equations

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\phi}_k} - \frac{\partial \mathcal{L}}{\partial \phi_k} = \tau_k,\ k = 1, \dots, n \tag{5.6}$$

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,

$$D(\phi)\ddot{\phi} + C(\phi, \dot{\phi})\dot{\phi} + g(\phi) = \tau \tag{5.7}$$

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 (5.7) that are helpful in the analysis and design of the manipulator control system. Among these are [SV89]:

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

$$\mu_1(\phi)I \leq D(\phi) \leq \mu_2(\phi)I \tag{5.8}$$

   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

$$\int_0^T \dot{\phi}^T(u)\tau(u)dt \geq -\beta \tag{5.9}$$

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 (5.7) 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.

5.1.3 Control

In the early days of robot development the manipulators where mainly used for set-point tracking. This means that the robot is programmed to move to a certain point but that the trajectory it follows in order to do that is not well defined. The controllers that were used in these robots where 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, see e.g. [SL91] for a proof.

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 in real control systems. The idea of feedback linearization is to algebraically transform a nonlinear system dynamics 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. We will give an example of the first type.

Given a plant model

$$M(\phi)\ddot{\phi} + C(\phi, \dot{\phi})\dot{\phi} + g(\phi) = \tau \tag{5.10}$$

as shown in (5.7). Here the inertia of the motor and gear has been included in the inertia matrix,

$$M(\phi) = D(\phi) + A(\phi) \tag{5.11}$$

$A(\phi)$ 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

$$\tau = M(\phi)a_\phi + C(\phi, \dot{\phi})\dot{q} + g(\phi) \tag{5.12}$$

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

$$\ddot{\phi} = a_\phi \tag{5.13}$$

If we now have a given reference trajectory that we want to follow, $r(t) = \phi_d(t)$, one possible choice of a_ϕ is

$$a_\phi = \ddot{\phi}_d + K_d(\dot{\phi}_d - \dot{\phi}) + K_p(\phi_d - \phi) \tag{5.14}$$

i.e. a PD controller with feedforward of the reference acceleration. We can now combine (5.13) and (5.14), using

$$\tilde{\phi} = \phi_d - \phi \tag{5.15}$$

arriving at

$$\ddot{\tilde{\phi}} + K_d\dot{\tilde{\phi}} + K_p\tilde{\phi} = 0 \tag{5.16}$$

and by choosing K_d and K_p the poles for the error characteristic equation, (5.16), can be placed arbitrarily. In the real system we will have restrictions on the control signal and this will limit also the speed 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 of the robot. For example, the inner loop controller can take care of the linearization, creating a closed loop system that from the reference to the output gives the impression of being a linear system. The outer loop controller, giving the overall control system the desired properties, can then be designed based on linear techniques. This approach is especially interesting for the ILC method where there are algorithms to design the Iterative Learning Control 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 not possible, of course. Possible solutions are to introduce adaptive or robust controllers [Cra88, SL91] or to use ILC.

Robust and adaptive control

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., 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 actually is 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.

Let us again consider the dynamic equations for the n-link manipulator,

$$M_0(\phi)\ddot{\phi} + C_0(\phi, \dot{\phi})\dot{\phi} + g_0(\phi) = \tau \tag{5.17}$$

with the control input given by

$$\tau = M(\phi)a_\phi + C(\phi, \dot{\phi})\dot{\phi} + g(\phi) \tag{5.18}$$

where M, C, and g represents the nominal values of the true system M_0, C_0, and g_0. We have now introduced a model error, $\tilde{(\cdot)} = (\cdot)_0 - (\cdot)$, 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

$$a_\phi = \ddot{\phi}_d + K_d(\dot{\phi}_d - \dot{\phi}) + K_p(\phi_d - \phi) - \delta a \tag{5.19}$$

and substitute (5.18) and (5.19) into (5.17). After some algebra we obtain

$$\ddot{\tilde{\phi}} + K_d\dot{\tilde{\phi}} + K_p\tilde{\phi} = \delta a + \eta(\phi, \dot{\phi}, \delta a, t) \tag{5.20}$$

where

$$\eta = M_0^{-1}\big(\tilde{M}(\ddot{\phi}_d + K_d\dot{\tilde{\phi}} + K_p\tilde{\phi} - \delta a) + \tilde{C}\dot{\phi} + \tilde{g}\big) \tag{5.21}$$

The result of (5.20) can be formulated as a linear state-space description

$$\dot{x} = Ax + B(\delta a + \eta) \tag{5.22}$$

where

$$x = \begin{bmatrix} \tilde{\phi} \\ \dot{\tilde{\phi}} \end{bmatrix}, \quad A = \begin{bmatrix} 0 & I \\ -K_p & -K_d \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ I \end{bmatrix} \tag{5.23}$$

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

$$\|\eta\| \leq \rho(x,t) \tag{5.24}$$

and to design the additional input term, δa, such that the state trajectory in (5.22) is bounded or, if possible, converges to zero. It is in general difficult to calculate ρ in (5.24) since the term η is a complex expression involving also the additional control term δa. One approach that has been tried is to design δa using sliding mode theory [SV89]. The simplest sliding mode controller results from choosing δa_i according to

$$\delta a_i = \rho_i(x,t)\mathrm{sgn}(s_i), \;\; i = 1, \dots, n \tag{5.25}$$

where ρ_i is the bound on the i-th component of η, $s_i = \dot{\tilde{q}} + \lambda_i \tilde{q}_i$ represent a sliding surface in the state-space and $\mathrm{sgn}(\cdot)$ 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 (5.22) is Hurwitz, which means that $\forall Q > 0, \;\; Q = Q^T \implies \exists P > 0$ with $P = P^T$, satisfying the Lyapunov equation,

$$A^T P + PA = -Q \tag{5.26}$$

This follows from the Lyapunov stability theory for LTI systems, see e.g. [SL91]. Using the matrix P, the term δa can be chosen as

$$\delta a = \begin{cases} -\rho(x,t)\frac{B^T Px}{\|B^T Px\|} & \text{if } \|B^T Px\| \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.27}$$

Using the Lyapunov function $V = x^T Px$ it is possible to show that $\dot{V}$ is negative definite along solution trajectories of the system given by (5.22).

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. There have been many refinements and extensions to the two approaches to robust feedback linearization, mainly to simplify the calculations of the uncertainty bounds, $\rho(x,t)$, and to smooth the chattering in the control signal [SLA92].

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 [SLA92] with references.

Adaptive Feedback Linearization

The work on adaptive control of robot manipulators can be divided into two phases [SLA92], the *approximation* phase (1979 - 1985) and a *linear parameterization* phase (1986-present). In the approximation phase the assumptions were 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 (5.17), but suppose that the parameters in (5.18) are not fixed. Instead, assume that they are time varying estimates of the true parameters. Let

$$a_\phi = \ddot{\phi}_d + K_d(\dot{\phi}_d - \dot{\phi}) + K_p(\phi_d - \phi) \tag{5.28}$$

Substitute (5.18) and (5.28) into (5.17). After some algebra we obtain

$$\ddot{\tilde{\phi}} + K_d\dot{\tilde{\phi}} + K_p\tilde{\phi} = \widehat{M}^{-1}\varphi(\phi, \dot{\phi}, \ddot{\phi})\tilde{\theta} \tag{5.29}$$

where φ is a regressor, and $\tilde{\theta} = \hat{\theta} - \theta$, where $\hat{\theta}$ is a parameter vector estimate. We can now write the system in (5.29) as

$$\dot{x} = Ax + B\Phi\tilde{\theta} \tag{5.30}$$

where

$$x = \begin{bmatrix} \tilde{\phi} \\ \dot{\tilde{\phi}} \end{bmatrix}, \; A = \begin{bmatrix} 0 & I \\ -K_p & -K_d \end{bmatrix}, \; B = \begin{bmatrix} 0 \\ I \end{bmatrix}, \; \Phi = \widehat{M}^{-1}\varphi(\phi, \dot{\phi}, \ddot{\phi}) \tag{5.31}$$

with 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 (5.30), 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

$$A^TP + PA = -Q \tag{5.32a}$$

$$B^TP = C \tag{5.32b}$$

If the parameter update law is chosen as

$$\dot{\hat{\theta}} = -\Gamma^{-1}\Phi^TCx \tag{5.33}$$

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

$$V = x^T P x + \frac{1}{2}\tilde{\theta}^T \Gamma \tilde{\theta} \tag{5.34}$$

This approach has some drawbacks.

- The parameter update 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.

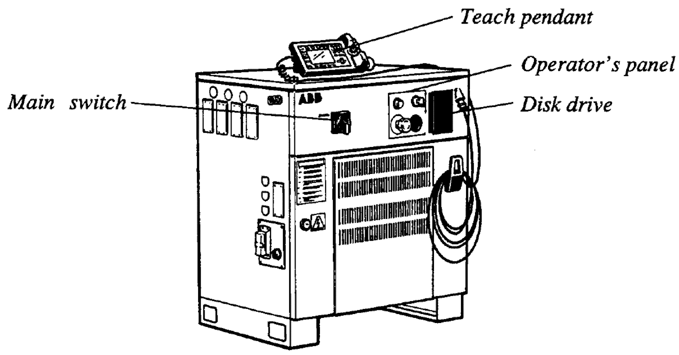
Later work has been devoted to overcome these drawbacks and it has been proved possible by using so-called *indirect* approaches based on a filtered prediction errors [SL91].

Aspects on robot design and control

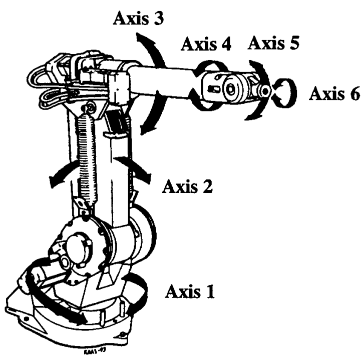
The structure of the manipulator will absolutely have an impact on the control. By using gears for example, it is possible to reduce the nonlinear coupling between the axis of the robot. This, of course, to the price that friction, backlash, and flexibilities are introduced. If the position is measured 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. The dynamic accuracy does not have to be increased in the same way, because of the flexibilities introduced by the gears. 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.

5.2 The ABB IRB Family

After the general description of robot manipulators and control we will now discuss 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 5.2 the two components are depicted.



a. The controller, ABB S4C.



b. The manipulator, ABB IRB 1400.

Figure 5.3 *The two main components in the robot system.*

We see 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. On the rear part of the upper arm, next to joint 3, motors for axes 4, 5 and 6 are mounted. This makes the robot lighter and more balanced, hence, it can move faster with less motor torque. Before describing the robot in more detail we will give some background on the ABB robots.

5.2.1 Background

The first generation robots from Asea/ABB was presented on the market in 1974 [Nil96]. The controller family was called S1 and the only available robots at that time were the IRB-6 and 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 presented. The main contributions in S4 were the new programming language, RAPID$^{\text{TM}}$, and a model-based motion control. The language, RAPID$^{\text{TM}}$, is used by the operator when programming the robot.

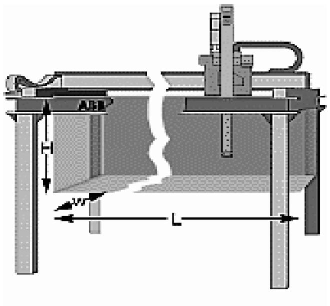


Figure 5.4 *An example of a gantry robot.*

Since 1994 the robots have been developed and today ABB has a full range of robots for different applications, ranging in size from the IRB1400, reaching about 1.5 m, to the IRB6400, reaching up to 3 m from the foot of the robot. Just recently a family of gantry robots were presented and some more specialized manipulators have also been introduced lately, like the IRB640 flex palletizer with only 4-DOF. A gantry robot is a robot that is mounted on a track as displayed in Figure 5.4 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 heavy objects from one production to another. As the number of applications grows the demand on the controller grows. The controller is today the same for all the different robots and this is made 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 the inaccuracy that is always present in the production of mechanical devices.

5.2.2 The Controller, S4C

The controller, depicted in Figure 5.3a, 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 last generation of the cabinet. 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, named 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. Also an optional board for the Ethernet communication is 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 robot has drive units such that it can control 4 or 6 AC motors, depending upon the number of DOF of the robot. Sometimes a cell is equipped with external axes that must be controlled from the cabinet. For this reason it is possible to control up to a total of twelve 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. When more than twelve axes are controlled from the robot the extra drives have to be put in a separate enclosure.

In Figure 5.5 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,
- the orientation of the tool, or

Figure 5.5 *The teach pendant, used for programming the robot.*

- the individual axes of the robot.

On the teach pendant there is also 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 robot.

5.2.3 The Manipulator, IRB1400

In the system that we consider the manipulator is an IRB1400, a 6-DOF manipulator with the structure of the joints according to Figure 5.3b. 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 the axes 4, 5 and 6. The IRB1400 manipulator is the smallest in the family of manipulators from ABB. In Figure 5.6 the measures of the arms are found to give an idea of the size of the manipulator. Note the springs that are mounted in parallel with the lower arm, rotating around joint 2 (cf. Figure 5.3b). These springs are used to balance the robot and to decrease the static load on 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. 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$^{\text{TM}}$ function the ABB robots have become 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

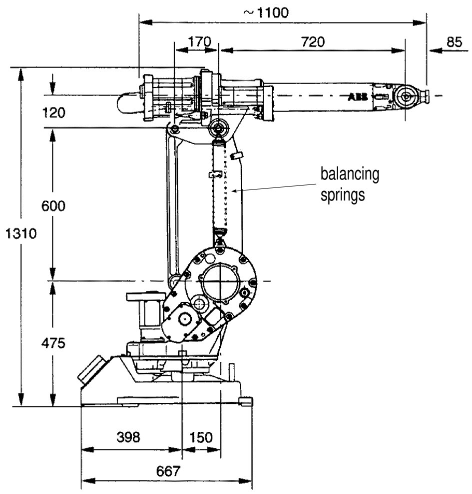


Figure 5.6 *To give an idea of the size of the manipulator.*

acceleration is 15 m/s^2. It is obvious that these measures are not valid for the robot in all the working range. Changing the robot configuration will mean changing the performance of the robot considerably.

6

Implementation

To study how the ILC method works in practice it is implemented and evaluated on an industrial system. We have chosen a commercial robot system from ABB Robotics. The system is described in Section 5.2 and in this chapter the technical aspects of the implementation will be discussed.

6.1 Overview

The system used in the implementation is depicted in Figure 6.1. It consists of a *manipulator*, a *controller* and a *terminal*. The manipulator is of type IRB1400, the controller is an ABB S4C, and the terminal is a standard PC.

6.1.1 Communication

The communication between the controller and the terminal is made on an Ethernet connection. The PC works as a file server for the robot controller. The same configuration is used in many factories where the robots are connected in a network. A reason to connect the robots in a network is to make it possible to do high level control, e.g. production control, of the factory.

During the experiments the robot is controlled from MATLAB™ running on the PC. Remember the ILC updating formula described in Chapters 2

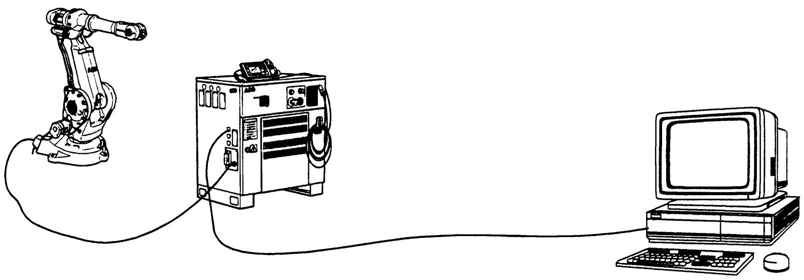


Figure 6.1 *The manipulator and the controller connected to the terminal.*

to 4,

$$u_{k+1}(t) = Q(q)(u_k(t) + L(q)e_k(t)) \tag{6.1}$$

This function is implemented in MATLAB™. The signals e_k and u_k are exchanged between the robot controller and the PC in order to apply ILC to the robot system. In the current version[1] of the implementation the transfer of information between the PC and the robot controller is made using the file system. This is slow and hence in the next version a direct link between MATLAB™ and the controller will be implemented. This link will also make it possible to exchange not only signals, as is possible in the current version, but also controller parameters. Data can also be transfered from the system continously, making it possible to evaluate algorithms for doing, e.g., diagnosis of the system.

6.1.2 Robot Controller Software

The software in the controller is a modified version of the software that ABB Robotics uses in the development of new control functions for the robot. As is common in test environments the number of modules are minimized in order to have as simple test environment as possible. For example, all modules that incorporate the *teach pendant* in the system are excluded in the test environment. Hence it is not possible to program the robot using the joystick. All the programming has to be made off-line.

[1] Version 1.0

In the test environment there are some functions built in that can be used for the evaluation of the ILC method. For example, there is a function that makes is possible to log internal signals in the controller, such as position references and measured positions. This function can be seen as an implementation of an oscilloscope, see Figure 6.2. When using the ILC we will also need to add the ILC control signal in the control loop. The tool that we need for this is basically a signal generator, see Figure 6.2. Using the software implementation of these two tools the ILC algorithm can be evaluated in the robot system.

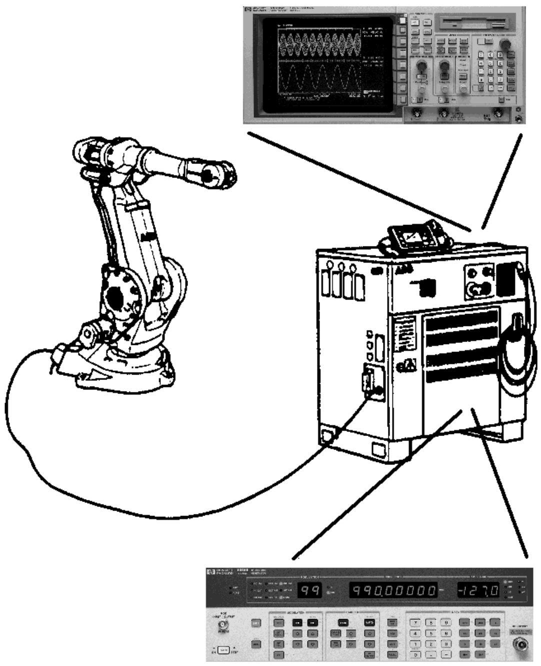


Figure 6.2 *Two familiar tools from electrical engineering, implemented in the robot controller.*

6.1.3 The Tools Used in the Evaluation

To make the following discussion more concise the names of the two functions equivalent to the oscilloscope and the signal generator are defined.

Definition 6.1 (Data-logger)
The software tool in the robot control system that makes it possible to record signals, e.g. position reference and measured position signals, is called the data-logger.

The corresponding software function that makes it possible to feed back signals into the control system is also defined.

Definition 6.2 (Data-input)
The addition to the robot control system that makes is possible to feed back compensation signals into the system is called the data-input function.

Note that the data-logger records the signals during an iteration and, in the current implementation of the function, the data are not available until after the iteration is completed. Thus, it is not possible to evaluate the class of ILC methods that uses the current iteration error in the control, i.e. the CITE Iterative Learning Control method [CWXS96, CXL96a, CXL96b, Be98] mentioned in Remark 2.4. In general, though, the CITE is more of an adaptive control method or robust control method. Using the errors from only previous iterations is the idea of the original ILC and this can be implemented with the current version. Of course, the combination of adaptive control and ILC would be interesting to investigate and the results in [CWXS96, CXL96a, Be98] are promising and interesting. Also in [OM97] the current iteration tracking error is used together with an adaptive feedback gain.

6.2 The Implementation

We are now going to present the implementation from a system theoretic viewpoint. In Figure 6.3 an abstract view of the implementation of the components in Figure 6.2 is depicted. The two functions *data-logger* and *data-input* are shown as dashed arrows going in to, and out of, the system. The system inside the bold rectangle is considered to be the robot control system which, in our case, is not possible to change. We have, however, implemented a function that makes it possible to feed back the control signal $\{u_k(t)\}_0^{t_f}$ to the system. This has to be done in order to make it possible to evaluate the ILC method.

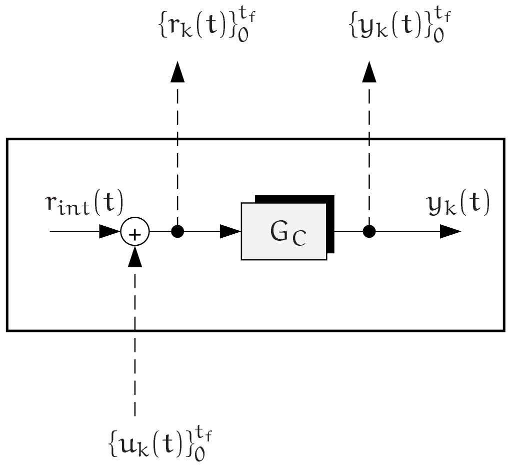


Figure 6.3 *Abstract view of the implementation.*

6.2.1 The Interface

We will now discuss the interface between MATLAB™ and the system, this will include the high level synchronization of the ILC and the robot control system. The data transfer between the robot controller and the PC, the dashed arrows in Figure 6.3, is made using the file system. Since the two systems run completely in parallel some kind of synchronization has to be done. A standard solution when dealing with concurrent systems is to use *semaphores* [Sta92].

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 processing of the compensation signal in the PC using MATLAB™, this process is called *process B*. Since the data is 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 6.4.

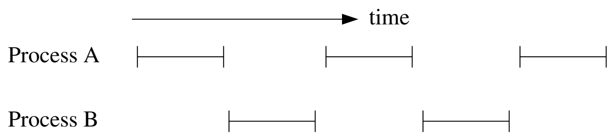


Figure 6.4 *The execution of the processes A and B.*

To show how the problem is solved using semaphores we introduce the primitives, `wait(semaphore)`, and `signal(semaphore)`. The semaphore can be seen as an 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 that is waiting is unblocked. Using the semaphores to synchronize the processes A and B, we get the resulting code shown in Figure 6.5.

Process A:
```
repeat
  wait(semA);
  perform_programmed_motion;
  log_data;
  save_logged_data;
  signal(semB);
until readyA
```

Process B:
```
repeat
  wait(semB);
  load_data;
  process_data;
  save_processed_data;
  signal(semA);
until readyB
```

Figure 6.5 *The resulting processes A and B, using the semaphore for mutual exclusion.*

We see that with only these two processes there will be problems when deciding 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 does the following steps,

1. create `samA` and `samB` 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 that the new correction term can be calculated, based on the error when ILC where not applied.

The implementation of the semaphores in the test environment 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 value 0 or 1 [Sta92], 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 new implementation. This makes the chosen solution of the synchronization and mutual exclusion very general.

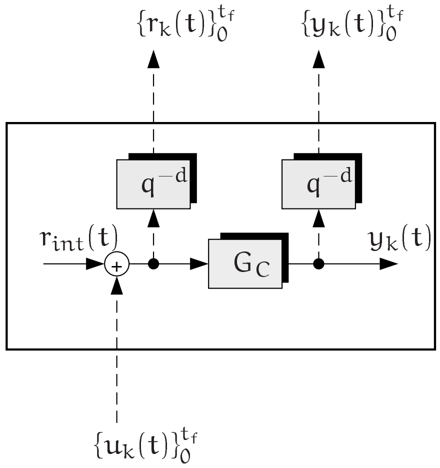
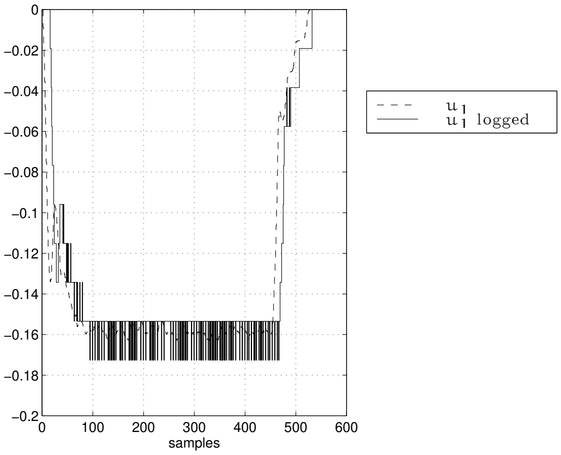


Figure 6.6 *The actual function of the implementation.*

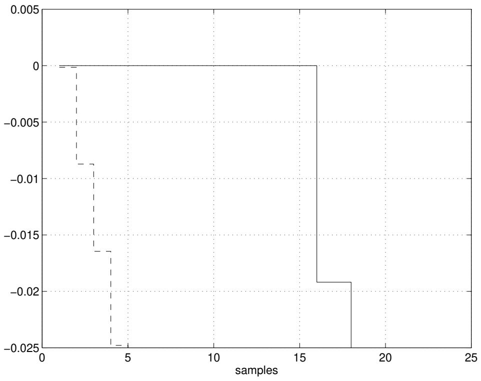
6.2.2 The Robot Controller

In the previous section the synchronization of the terminal and the robot control system was discussed. If we, again, consider the system depicted in Figure 6.3 we can see that we will also, in practice, have to consider the synchronization of the data-input and data-logger function. In the system depicted in Figure 6.6 this problem is shown. The synchronization problem can be seen as the problem of identifying the delay, d, that we get from the data-logger.

The result from an experiment on the system is shown in Figure 6.7 and we can from Figure 6.7a identify the delay, d, to be 14 samples. In practice, however, the delay has been different in different experiments and this is a problem that has to be considered for further work.



a. The complete motion.



b. A piece of the trajectory in Figure 6.7a showing the first 20 samples of the motion.

Figure 6.7 *The synchronization problem.*

6.2.3 The Terminal

The programs that run on the robot are taken from the hard disk of the PC. By altering the programs or creating new programs using, e.g. MATLAB™, it is actually 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, described in Section 6.2.1.

Communication

To be able to evaluate the ILC method, data transfer between the ILC updating formula, implemented in MATLAB™, and the robot controller has to be performed. To do this two functions, called `readlog` and `writelog`, are used. They make it possible to read the data file created by the robot control system and to write back the processed file that eventually is read by the data-input function.

The ILC

The implementation of the ILC is done using the updating law for the control signal discussed in Chapter 2,

$$u_{k+1}(t) = Q(q)(u_k(t) + L(q)e_k(t)) \tag{6.2}$$

where the actual choice of the filters Q and L will be discussed in the next chapter. The expression in MATLAB™ that perform the action of (6.2) is

```
u = filter(Qb,Qa,u + ncfilter(Lb,La, e));
```

where `filter` is the standard filter operator found in the *Signal Processing Toolbox* [LS88]. The variables `u` and `e` are vectors containing time samples of the previous control signal (transferred to the current) and the previous error, respectively. The operator `ncfilter` is based on the `filter` operator but extend the filter operator to take also non-causal filters. The filter L usually has a phase advance term that makes it non-causal in the normal filter sense. To illustrate the use of the different operators and how the filters can look like in MATLAB™ we will now show an example.

Example 6.1 (ILC example in MATLAB™) *To create a low-pass filter Q we can use the function* `butter` *in MATLAB™, creating a Butterworth filter of order n and cut-off frequency $\omega \in [0, 1]$. In MATLAB™ this will look like,*

```
>> [Qb,Qa] = butter(2,0.6)
Qb =
    0.3913    0.7827    0.3913

Qa =
    1.0000    0.3695    0.1958
```

giving a low-pass filter of second order and cut-off frequency 0.6 of the Nyquist frequency. To interpret the vectors we can write the filter Q as

$$Q(q) = \frac{Q_b(q)}{Q_a(q)} = \frac{0.3913 + 0.7827q^{-1} + 0.3913q^{-2}}{1 + 0.3695q^{-1} + 0.1958q^{-2}} \tag{6.3}$$

we see that the elements in the row vectors, `Qa` *and* `Qb`*, represent the coefficients of the polynomials in* q*. If we now use the filter* $L(q) = q$ *we get the following representation in MATLAB™,*

```
>> Lb = [1 0]; La = [0 1];
```

and it is possible to use the updating formula given by,

```
>> u = filter(Qb,Qa,u + ncfilter(Lb,La, e));
```

□

Note that since the signals $u_k(t)$ and $e_k(t)$ are completely defined over the interval $[0, t_f]$ we can use also a non-causal Q, if we find that convenient. For example it is possible to use a zero phase filter Q. This can be realized in MATLAB™ using the operator `filtfilt`, that have the same arguments as `filter` but filters the signal first in the direction of time and then also in the reverse direction. By doing these two operations a zero-phase filter is realized. As we will see in the next chapter this idea has been applied to the ILC evaluated on the robot system.

6.2.4 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 Iterative Learning Control Feedback 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 in 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 is implemented in MATLAB$^{\text{TM}}$. Using these functions, we have an environment where it is possible to try and compare different ILC updating formulas and this will be more explored in Chapter 7.

6.3 Some Concluding Remarks

To conclude this chapter, describing the implementation, we will say some words about possible future extensions of the software. Finally we will discuss the particular challenge of doing research in an industrial system.

6.3.1 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. We will now in a few words summarize some of the things that could be done in the implementation to significantly improve the functionality.

The interface

Today the file system is used as the communication interface between the robot system and the terminal. It would be a major improvement to connect the two system directly by, e.g., using 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 of data from the data-logger to the terminal and to the data-input function from the terminal should also be much faster.

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 have to be removed and a more secure synchronization of the data-input function and the data-logger has to be found.

The terminal

The major extension and improvement needed for the software in the terminal is a more user friendly interface to the robot. 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 only use MATLAB™ on the PC. 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. Using a graphical interface to run the test environment would also give a much more stable environment. In the terminal environment used today a very small mistake makes the robot control software crash, leading to a long reboot process of the robot, which is not acceptable.

6.3.2 Working in a Commercial System

One of the challenges of the work presented in the thesis is that it is implemented in a commercial system. The modifications that we have done to the system can easily be made a part of the commercial software that will be delivered to the customers of the robots. The fact that the system is a commercial system puts also some limitations on what can be done, at least with a descent amount of work.

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 do not have time for this it is impossible to complete the exercise and the project would most certainly fail.

For a company the idea to have researchers doing changes in the software would perhaps seem impossible in many cases. Sometimes the code in itself is a company secret and moving the source code to the research community would, basically, mean to give the source code away. One solution, that we have adopted, is that all the programming work is made inside the walls of the company. This has proved to be a very good way to transfer knowledge both from the research community to the company and vice versa.

7

Experiments

In this chapter the ILC method, described in Chapters 2 to 4, is applied to the industrial robot control system described in Chapter 5. The foundation for the experiments is the implementation described in the previous chapter.

7.1 Introduction

The goal of this chapter is to examine the ILC method applied to an industrial system and investigate the possibility to improve the position tracking performance of the system.

7.1.1 Prerequisites

Before applying the ILC method to the robot control problem we will discuss some of the prerequisites that we have for applying ILC. We can describe the most important as

1. the system starts from the same initial condition, and
2. the dynamics of the system is repetitive.

The two conditions are very crucial and we will now look at two different repeated motions for the robot to see if the prerequisites are fulfilled. We will study the position following or position tracking of the robot. In Figure 7.1

the resulting position errors from 20 iterations of the two different motions are depicted. Figure 7.1a shows the position errors on the motor side for axes 2 and 3 using a combined motion of axes 1, 2, and 3. The robot to the right in Figure 7.1a shows how the three axes are moved in the combined motion. There is a rotation clockwise of axis 1, backwards of axis 2, and upwards of axis 3. In Figure 7.1b the position errors of another motion using axes 2 and 3 are shown. It is obvious that the errors in the different iterations of the two motions are highly repeatable and hence the two conditions are fulfilled.

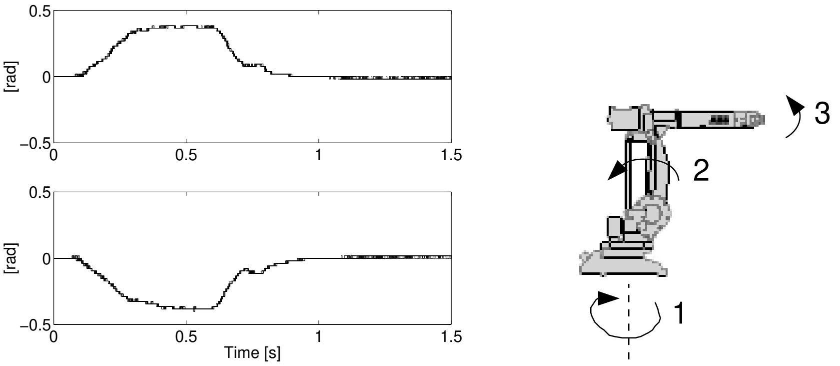
Remark 7.1 (Path error)
There are several things that can be said about the result presented in Figure 7.1. First of all the errors are very small already before applying the ILC. The errors shown are on the motor side, i.e. before the transmission. This means that the errors on the arm side are reduced by a factor of about 100, and since the errors are not bigger than 0.6 rad the errors on the arm side will only be about 0.006 rad. Of course not considering the flexibilities in the arm and transmission. To give an idea of the size of the error in millimeters assume that we have an arm of 1 m length. Using the case with the joint motion, e.g. only axis 3, we get an error in the path following of 6 mm (1000×0.006 mm). In many applications this will not be important, especially since the error will be a delay on the path and not actually a path error. In a more complicated motion the result, however, can be a deviation from the programmed path.

□

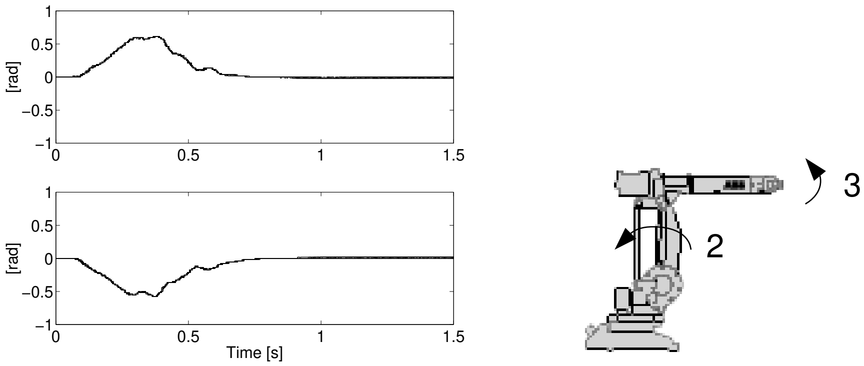
We see that the prerequisites for applying ILC are fulfilled and that the controller already implemented in the control system performs a very good tracking on the motor side. It is important to note that, in order to have a good path following also on the arm side a very stiff robot is required or a compensation for the flexibilities found in the manipulator structure has to be made.

7.1.2 Overview of the Experiments

The experiments have all been made in the research lab of the *Division of Automatic Control* using the robot described in Chapter 5 and the software described in the previous chapter, Chapter 6. The idea of the experiments is to verify the ILC method and to see if it is applicable to the robot control case. We have also examined some of the properties of the filters in the



a. Combined motion axis 1, 2, and 3. Position error on the motor side for axis 2 (upper), axis 3 (lower).



b. Combined motion axis 2 and 3. Position error on the motor side for axis 2 (upper), axis 3 (lower).

Figure 7.1 *Twenty iterations without ILC on two different motions using a standard ABB IRB 1400. The plots show the position errors in motor coordinates for axis 2 and 3 respectively. In (a) axis 1 is also moved, but the error on that axis is not shown.*

ILC updating formula to see how changes of the filters affect the overall performance of the system.

Problems with the implementation

In the previous chapter it was noted that the implementation of the data-logging function and the data-input function have some synchronization problems. We also identified the nominal time delay that we have between the feedback and that the actual data logging starts. We saw that the delay is considerable, about 15 samples at 4 ms sampling time. This delay is compensated for in the experiments.

The Experiments

The experiments that we have made on the robot starts with the most simple case, control of axis 1 only. This case is simple because it means controlling a SISO system without the gravity disturbance present in most of the other possible configurations. In some of the experiments we have used two different speeds for the motion. One is a programmed speed of 0.6 m/s and the other is a programmed speed of 3 m/s, both refer to the speed of the TCP. Since the maximum speed for the robot is less than 3 m/s the programmed speed will never be reached. We can however treat the 3 m/s motion as a high speed motion and the 0.6 m/s motion as a slow speed motion.

In the second experiment ILC is used in a combined motion of axes 1 and 3. This experiment shows that it is possible to apply the method on a combined axes motion, resulting in good properties of the position tracking of the system. The third experiment involves a zero-crossing in velocity of axis 1 and the idea is to show the ability of the method to compensate for joint friction, although the friction in this case showed not to be very significant.

The fourth experiment includes examining changes of the filters Q and L and how this will give different behavior of the ILCF system with respect to the convergence speed and the resulting error.

7.2 Links to the Analysis Results of ILC

We will now give some ideas to how the analysis results of Chapter 3 apply in the robot case. The complete analysis, with a thorough consideration of

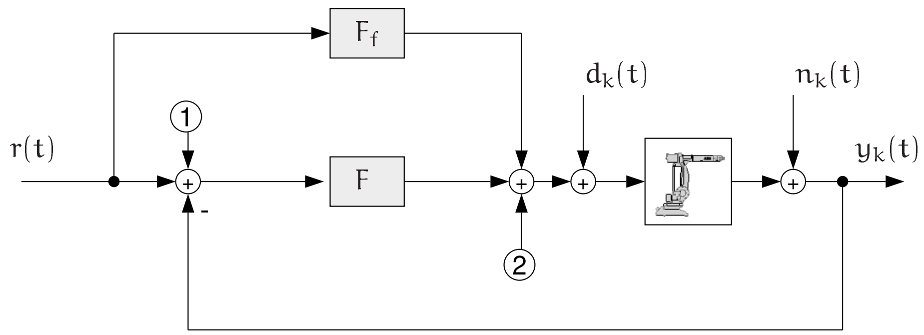


Figure 7.2 *A simplified view of the robot system used in the experiment.*

the robustness aspects of the applied ILC, has not been performed. Ideas how this should be done are however presented.

7.2.1 A Simplified View of the Robot Control System

In Figure 7.2 a simplified view of the robot control system is shown. The blocks represents the robot and the controllers F_f and F. The robot is a nonlinear 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 not known to us but in principal the control scheme of Figure 7.2 gives an idea of how it works.

The input to the system in Figure 7.2 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 principal is discussed in Chapter 5. 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. We will not discuss these disturbances more in the thesis but their nature is a question for future research.

Now we will give some links from the experiments to the theory in Chapters 2 to 4.

7.2.2 Applying ILC

In Chapter 3 we have done a general analysis of the convergence properties of a system where ILC is applied. Using the data-input function, implemented in the robot control system, we are limited to applying the ILC control signal at the points marked ① and ② in Figure 7.2, i.e. at the position reference and at the torque applied to the robot. We have chosen to apply the ILC control signal at ① using the error defined by

$$e_k(t) = r(t) - y_k(t) \tag{7.1}$$

Using the data-logger we sample the reference signal, $r(t)$, and the motor position signal, $y_k(t)$, cf. Figure 7.2. With the definition of the error signal as in (7.1) we can calculate the ILC control signal using the updating formula

$$u_{k+1}(t) = Q(q)(u_k(t) + L(q)e_k(t)) \tag{7.2}$$

where $Q(q)$ and $L(q)$ are linear filters that we will discuss later in this section. In the analysis chapter we presented two different views of ILC,

- the tracking approach, and
- the disturbance rejection approach.

Here the tracking approach will be used and we will assume that the closed loop system is completely decoupled and also linear. Hence, it is enough to consider only the SISO case. Since we are only considering the control errors on the motor side the pre-filter will not have to be considered in the analysis. Using Theorem 3.2 with

$$T_u(q) = G_{yu_k}(q) = \frac{F(q)G(q)}{1 + (1 + F_b(q))F(q)G(q)} \tag{7.3a}$$

$$T_r(q) = (F_f(q)F^{-1}(q) + 1 + F_d(q))G_{yu_k}(q) \tag{7.3b}$$

where $G(q)$ is assumed to be a linearized model of a robot joint. We get the stability criterion

$$|1 - L(e^{i\omega t_s})G_{yu_k}(e^{i\omega t_s})| < |Q^{-1}(e^{i\omega t_s})|, \ \forall\, \omega \tag{7.4}$$

where $L(q)$ and $Q(q)$ have to be chosen.

Choices of Q and L in the experiments

The filter Q used in the experiments has been chosen somewhat ad-hoc. We have seen in the analysis chapter, Chapter 3, that Q has a robustifying effect on the ILC updating formula and we have also seen from simulations that we will have difficulties with the convergence at high frequencies, cf Example 2.1. Using Q as a low-pass filter is therefore a convenient choice. In the experiments we have used the following filter,

$$Q^{1/2}(q) = \frac{0.2 + 0.1956q^{-1} - 0.0476q^{-2}}{1 - 0.8172q^{-1} + 0.1653q^{-2}} \tag{7.5}$$

which we have used as a zero-phase filter by applying the `filtfilt` operator in MATLAB™. The amplitude plot for the filter in (7.5) is depicted in Figure 7.3. The actual optimal choice for the robot is something that we have to examine more thoroughly in future research.

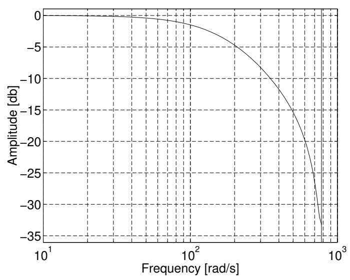


Figure 7.3 *The amplitude plot for the filter* $Q^{1/2}$.

7.2.3 Modeling of the Closed Loop System

We will now discuss an identification experiment that we have performed on the robot. Consider again the system depicted in Figure 7.2. We have chosen to apply the additional control signal, $u_k(t)$ at the point marked by ①.

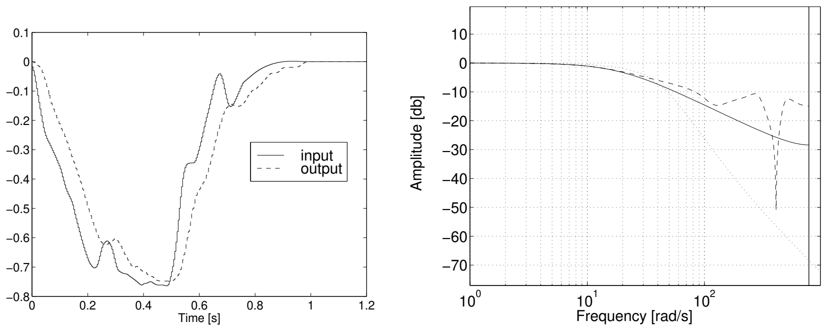
Identification

In order to identify G_{yu_k} we have applied a signal $u_k(t)$ at the point ①. From the schematic view of the robot control system in Figure 7.2 we can see that there are feedforward of both speed and torque in the system. To identify the system seen from point ① we need to do an experiment where we remove the effect of the feedforward.

We have performed the identification experiment on axis 1 using the data from one of the ILC experiments. In the ILC experiment we first determine a trajectory for the robot to follow and then we run the system while monitoring the reference signal $r(t)$, and the position signal $y_0(t)$. These signals are then kept in the memory of the PC running MATLAB™, cf. the setup we have when running the experiments in Chapter 6. The ILC experiment have run for 10 iterations and by considering $u_{10}(t)$ and the resulting $y_{10}(t)$ we can calculate the output that comes from the additional control signal, u_k. We get the following expression

$$\Delta y(t) = y_{10}(t) - y_0(t) = G_{yu_k}(q)u_{10}(t) \tag{7.6}$$

where we have to choose the model structure for the system $G_{yu_k}(q)$. For the modeling task we have used the *System Identification Toolbox* [Lju95] in MATLAB™ to provide a model from the data shown in Figure 7.4a.



a. Input signal, $u_{10}(t)$. 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 7.4 *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.*

The identification procedure will not be covered here, the interested reader is referred to [Lju87], but we will still say a few words about the model structure. We have chosen to model the system using an ARX model structure,

$$\begin{aligned} y(t) + a_1 y(t-1) + \ldots + & a_{n_a} y(t-n_a) \\ = & b_1 u(t-1) + \ldots + b_{n_b} u(t-n_b) + e(t) \end{aligned} \tag{7.7}$$

To make the identification we need to choose the number of parameters that should be estimated, n_a and n_b. There is also a parameter n_k indicating the time delay that we have 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

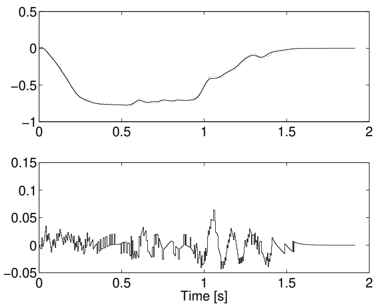
$$\hat{G}_{u_k y}(q) = \frac{0.07q^{-1}}{1 - 0.93q^{-1}} \tag{7.8}$$

In Figure 7.4b the amplitude plot of the model is shown together with a model calculated based on prior knowledge of the physical system 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 7.4b the amplitude curve for this model is shown. The actual parameters for the high order model are not given here.

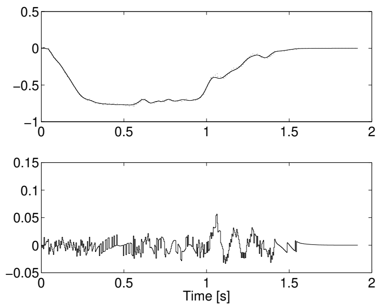
Model validation

The amplitude plots of the different models in Figure 7.4b 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.

We have also made a validation of the model using another set of data from the system. The actual output from the system is compared with the output that is simulated by the ARX models. In Figure 7.5 the result from this validation is shown and we see that the simulation of the models nearly exactly gives the same behavior of the output as we have in the real system. The very fast switching that can be seen in the prediction error is caused by a quantization performed in the DSP. The quantization comes from the fact that the calculation in the DSP is made using integer values only and the steps we can see in the measured output comes from this quantization.



a. Model 1.

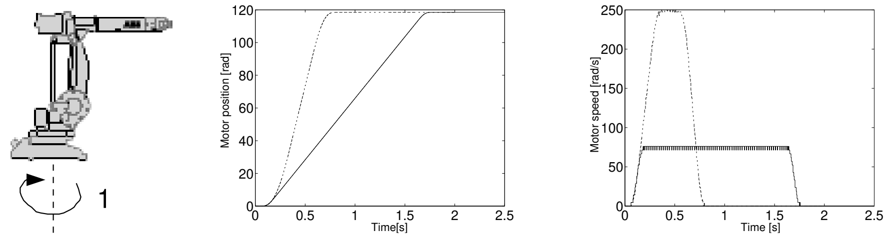


b. Model 2.

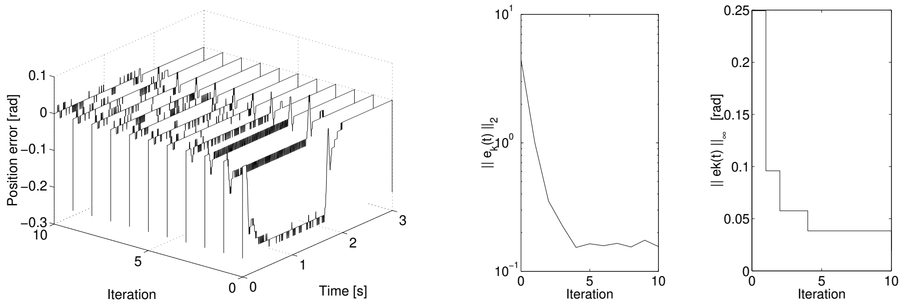
Figure 7.5 *Validation of the two models using validation data from the system. Above: predicted value (solid), measured value (dotted). Below: prediction error.*

7.3 Single Axis Experiment

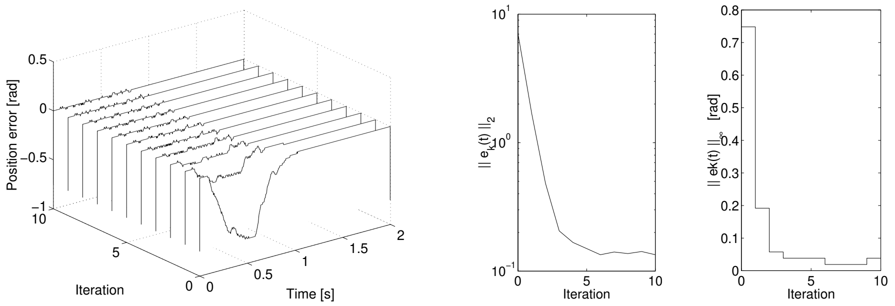
We will start with an experiment involving one of the axes of the robot, shown in Figure 7.6a (cf. Figure 5.3b). The reference trajectory for joint position and joint speed are depicted in the two plots on the right in Figure 7.6a, both the trajectories are given in motor coordinates. The motion on the arm side is a rotation by 1 rad of axis 1. Since the motion on the arm side is 1 rad we can see 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 TCP has traveled in the two motions we can assume that the length of the upper arm, link 3, is 1 m. Moving joint one by 1 rad will mean a length of the trajectory of 1m. This motion is performed in less than one second in the fast motion case, this means a speed of more than 1 m/s. The programmed speed of the TCP was 3 m/s but this speed is not reached in this kind of motion. An approximate measure of the size of the error on the arm side can be achieved in the same way as we did when calculating the distance that the TCP have moved. We have that the gear ratio is about 120:1, the position error on the motor side is less than 0.8 rad. Translated to the arm side this means an error of $0.8/120$ (without considering the dynamical effects in the transmission). Using the arm length of 1 m we get an approximate value of the maximum error on the arm side of 7 mm.



a. Illustration of the motion in experiment 1, the single axis experiment. Axis 1 motion (left), reference trajectories for position (middle) and speed (right) for the slow (solid) and fast (dashed) motion.



b. Slow motion.



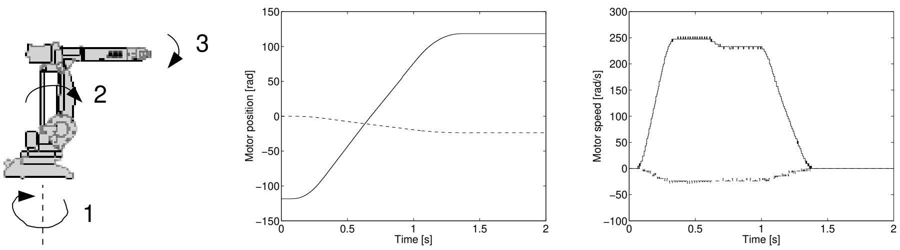
c. Fast motion.

Figure 7.6 *Motion (a) and position errors as a function of iteration and time in the single axis experiment, (b) and (c).*

In Figure 7.6 the resulting position error from 10 iterations of the two single axis experiments are depicted. We see that for both experiments we have been able to reduce the position error down to the quantization noise level in less than 10 iterations. The filter Q used in the two motions is given by (7.5). L is here chosen as $L(q) = 0.8q^5$.

7.4 Combined Axes Motion Experiment

In the combined axes motion experiment three axes are involved in the motion, namely axes 1, 2, and 3. We have, however, only applied ILC to axes 1 and 3. In this experiment we have only performed one motion. This single motion includes, however, a fast motion of axis 1 and a slow motion of axis 3. In Figure 7.7a we see the axes involved in the combined axes motion and how they are moved. The reference signals for axes 1 and 3 are shown in Figure 7.7b where we can see that the difference in speed between the two axes is considerable. The data from this experiment are also used in the validation of the model for axis 1, made in Section 7.2.

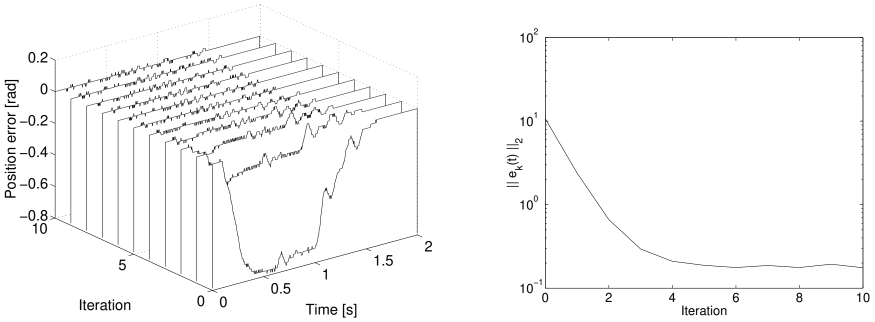


a. Axes 1, 2, and 3 in a combined motion.

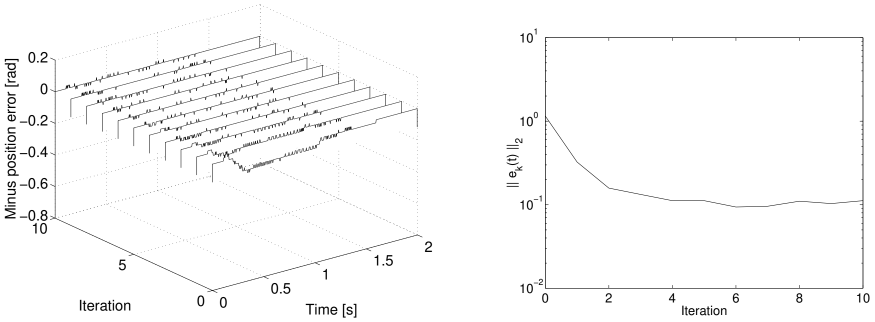
b. Reference trajectories for position and speed of axis 1 (solid) and axis 3 (dashed).

Figure 7.7 *Illustration of the motion in experiment 2, the combined axes motion experiment.*

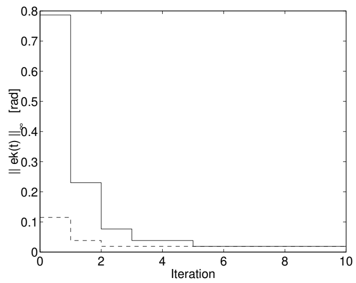
In the ILC used for the second experiment we have the Q filter from (7.5) and the L filter given by $L(q) = 0.8q^5$. The result from the second experiment is depicted in Figure 7.8. Just as we saw in the previous experiment, Section 7.3, we can see that the position error is reduced down to the quantization noise level after less than 10 iterations. For axis 3 the



a. Axis 1.



b. Axis 3. Note that the left diagram shows $-e_k(t)$ as a function of t and k.



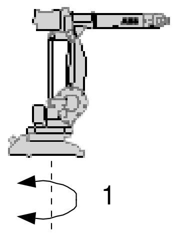
c. Comparison of $\|e_k(t)\|_\infty$ for the two axes, axis 1 (solid) and axis 3 (dashed).

Figure 7.8 *The position error as a function of iteration and time in the combined axes motion experiment.*

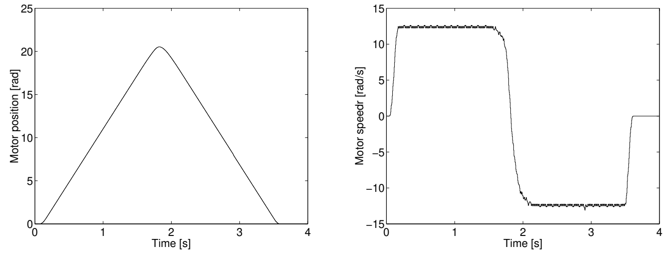
size of the error is, actually, reduced to the quantization noise level after 2 iterations of ILC, as could be seen in Figure 7.8c. In Figure 7.8a and Figure 7.8b the position error and the 2-norm of the error for axis 1 and axis 3, respectively, are shown. We can see that the error reduces very rapidly also in the 2-norm.

7.5 Experiment Involving Friction

We are now going to look at an experiment that will involve a zero crossing in the velocity trajectory. This will mean that the friction acting on the joint will be maximally excited, the change of the direction of movement will introduce a change of sign of the Coulomb friction. In Figure 7.9a the motion performed with the robot is explained. We make a movement of axis 1, first clockwise and then counter clockwise back to the starting position. In Figure 7.9b the joint trajectories are depicted.



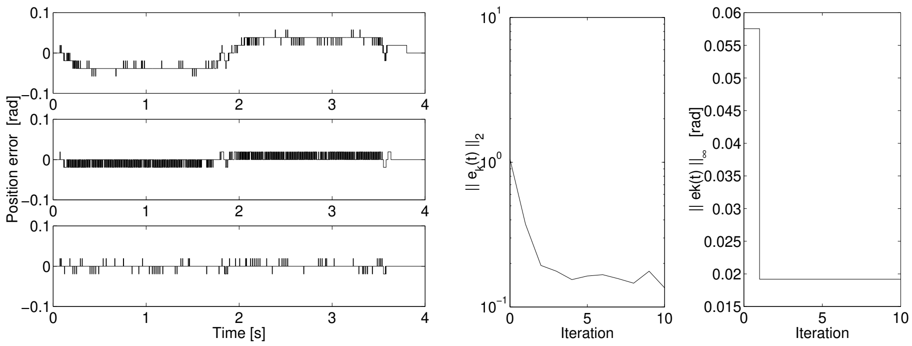
a. Axes 1 forward and backward motion.



b. Reference trajectories for position and speed.

Figure 7.9 *Illustration of the motion in experiment 3.*

The result from the experiment is shown in Figure 7.10. What is important to stress is that already in the first iteration the error is reduced to having an ∞-norm being the size of the quantization. The impact of friction is not seen at all, this can of course also be because the friction for this particular joint is very low. Some simulation results on ILC with friction is found in, e.g., [NG97]. The Q filter used here is the same as described in (7.5) and the L filter is chosen to be $L(q) = 0.8q^5$.



a. Position error in iterations 0, 1, and 3.

b. The 2-norm (left) and the ∞-norm (right) of the position error in the different iterations.

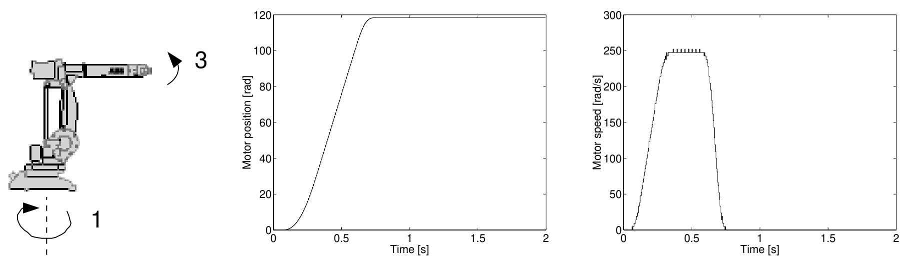
Figure 7.10 *The position error depicted in different ways for experiment 3.*

7.6 Different Choices of Q(q) and L(q)

In this section we will consider what happens with the ILCF system when we change the filters in the ILC updating formula. In the first example we will change the gain of the L filter by letting $L(q) = \kappa q^5$ and change the constant κ. The idea is to show that by reducing the gain in the ILC feedback we will have a slower convergence and by increasing the gain we will finally get an unstable ILCF system. In the second example the Q filter will be set to Q(q) times a factor less than one showing that the error no longer will converge to zero. In the last example of this section an experiment with an L chosen according to the design approach described in Chapter 4 will be shown.

7.6.1 Changing the Gain of L(q)

We will start by considering the effect of reducing the gain of L. We have made an experiment, using the motion shown in Figure 7.11. The gain κ in $L(q) = \kappa q^5$ is reduced from 0.8 down to 0.2 and the result from this experiment is shown in Figure 7.12. We can see that the speed of convergence is highly dependent on the size of the gain in the ILC feedback. If we



a. Axes 1 and 3 motion.

b. Reference trajectory for the position and the speed of axis 3 in the combined motion.

Figure 7.11 *Illustration of the motion in experiment 4, changing the gain of the filter* L.

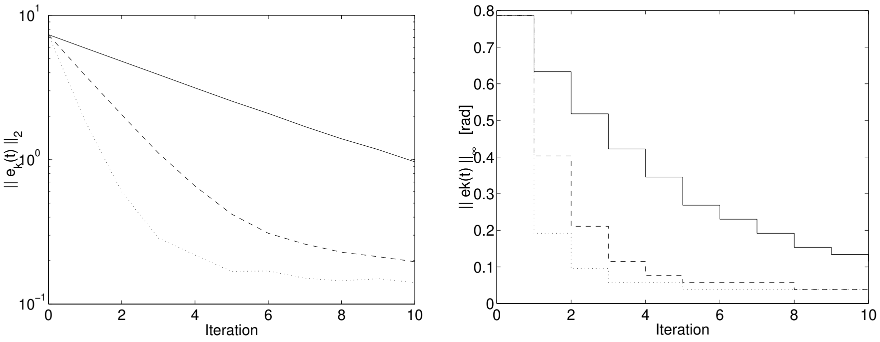
recall the updating equation of the error for SISO systems, given by (3.11) in Chapter 3,

$$E_{k+1}(e^{i\omega t_s}) = (1 - T_u(e^{i\omega t_s})L(e^{i\omega t_s}))E_k(e^{i\omega t_s}) \tag{7.9}$$

where we have assumed that $Q \equiv 1$ and that there are no disturbances acting on the system. We see that the closer $T_u(e^{i\omega t_s})L(e^{i\omega t_s})$ is to one the faster is the convergence of the error. In our case $T_u(e^{i\omega t_s})$ is the system that we described in Section 7.2 as a low pass filter. This means that by decreasing the gain of L as we have done in the experiment shown in Figure 7.12 will reduce the convergence speed.

We are now going to consider the case when the gain κ is made larger than 1. We have made one experiment for this case and it is when $\kappa = 1.6$. The result from this experiment is shown in 7.13. We can see that the increase of the gain introduces a damped oscillation in the iteration direction. Increasing κ even more results in an growing oscillation of the error and a rapidly growing ILC input signal $u_k(t)$. When $u_k(t)$ grows over a certain level the robot control system makes an emergency stop. The reason for this is that the internal supervision system detects the discrepancy between the programmed position and the actual position of the robot to be to large. The supervision is used as a safety function to avoid accidents if the robot starts moving without having an accurate reference trajectory.

After studying the case when we change the gain of L we will now consider the case of changing the δ when we have $L(q) = 0.8q^{\delta}$. We have



a. The 2-norm of the position error in the different iterations.

b. The ∞-norm of the position error in the different iterations.

Figure 7.12 *The 2-norm and the ∞-norm of the position error shown for* $\mathsf{L}(q) = 0.2q^5$ *(solid),* $\mathsf{L}(q) = 0.5q^5$ *(dashed), and* $\mathsf{L}(q) = 0.8q^5$ *(dotted).*

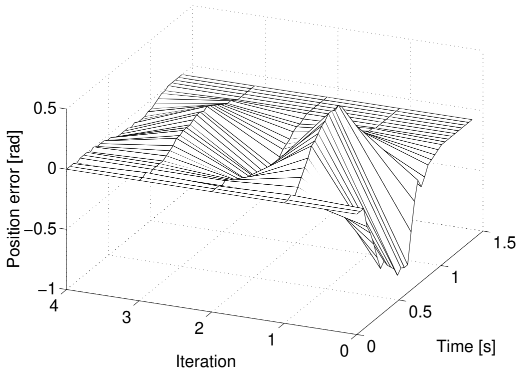
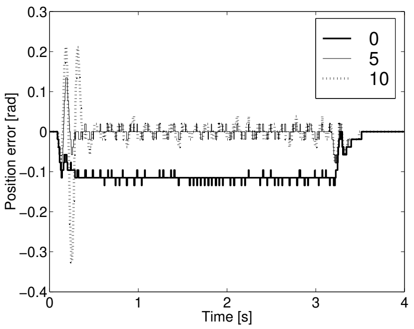
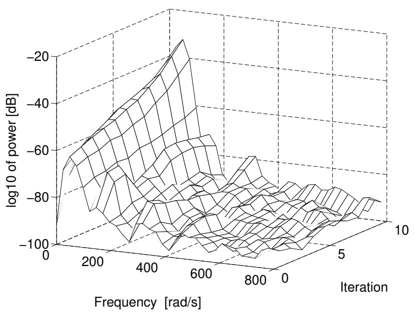


Figure 7.13 *The position error as a function of iteration and time when* $\kappa = 1.6$.

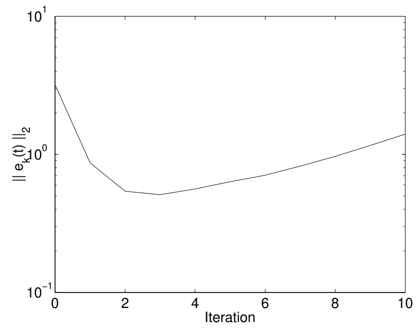
considered a simple case of this in the introductory example of Chapter 2, Example 2.1. In the example we considered $L(q) = 1$ and $L(q) = q$ and showed convergence in the second case and divergence in the first case. We also saw that the error in the 2-norm first was reduced before it started to grow. The reason for this was that in the updating equation for the error, (7.9), we have that for the low frequency band the transfer function between the errors is less than one but for high frequencies the transfer function is larger than 1. The same kind of behavior is seen in Figure 7.14 and we can also see in Figure 7.14a that the steady state error is reduced while the ILC introduces a strong oscillation at higher frequencies.



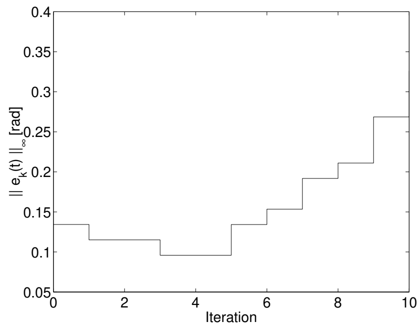
a. Position error for iterations 0, 5 and 10.



b. Spectrum of the position error as a function of iteration.



c. The 2-norm of the position error in the different iterations.

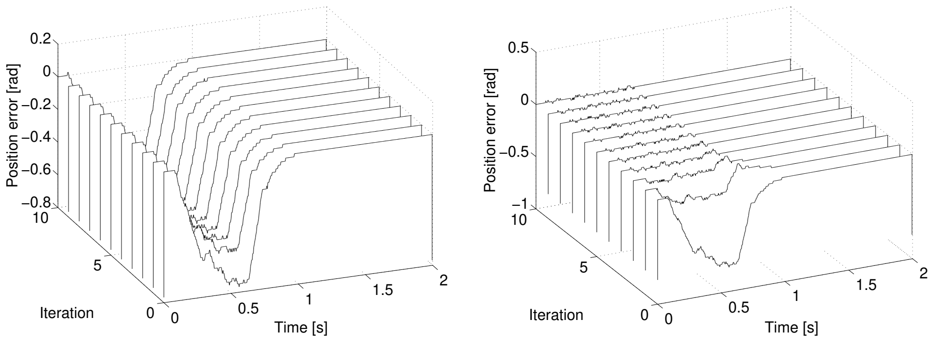


d. The ∞-norm of the position error in the different iterations.

Figure 7.14 *The results from the experiment having* $L(q) = q^{-4}$.

7.6.2 Changing the Filter Q(q)

We will now look at the effect of changing the filter Q. The motion in this experiment is the same as is depicted in Figure 7.11 and the trajectory for axis 1 is given by Figure 7.6a. As we mentioned in the analysis, in Chapter 3, the Q filter has a robustifying property of the ILC updating formula. We said also that introducing the Q filter we will have to pay the price that the resulting error will not be zero (in the linear case). We have made two experiments: First we used a zero phase Q filter given by $0.5Q^{1/2}(q)$, where $Q^{1/2}$ is given by (7.5). The result from this experiment is shown in Figure 7.15a. In the second experiment we have used the Q given by (7.5), without the scaling factor. The result is obviously that the error when using the robustifying effect of the Q filter is that the error will no longer converge to zero.



a. Position error for axis 1 when using a Q filter $0.5Q^{1/2}$.

b. Position error of axis 1 when using $Q^{1/2}$.

Figure 7.15 *The position error from two different experiments of the same motion using different Q filters.*

7.6.3 Experiment with the Model Based L(q)

To conclude the experiments we have made a test with the L filter designed according to Algorithm 4.2. The model that we have used is the low order ARX model found in (7.8). We have assumed that the model is correct and therefore used $H_B(q) = 0$. This means that the filter L is chosen as,

$$L(q) = T_u^{-1}(q) = \hat{G}_{yu}^{-1}(q) = \frac{1 - 0.93q^{-1}}{0.07q^{-1}} \tag{7.10}$$

The robustness has been achieved by using a Q filter. The $Q^{1/2}$ filter used in the experiment is 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 (cf. Example 6.1),

```
>> [Qb,Qa] = butter(2,0.2);
```

A zero-phase filter is achieved by using the function `filtfilt` in MATLAB™. In Figure 7.16 the amplitude diagram for $Q^{1/2}$ is depicted.

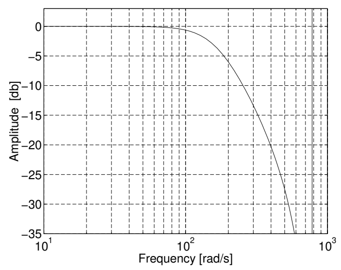
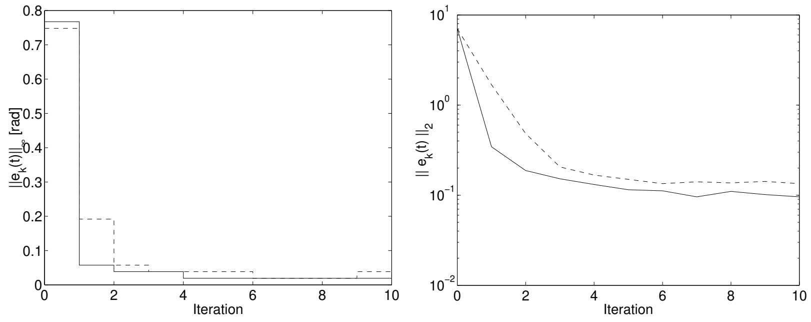


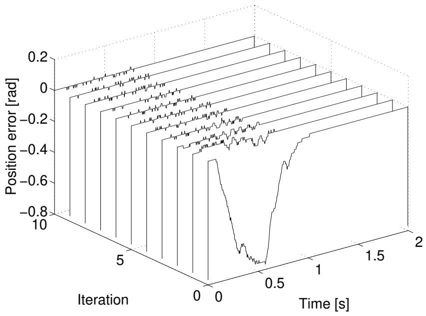
Figure 7.16 *The amplitude plot for the filter* $Q^{1/2}$.

The results from using the model based L filter are depicted in Figure 7.17. Figure 7.17a shows the norm of the error. To compare we show the result from two different experiments, the first using the model based L filter, and the second using the heuristically derived L, cf. Algorithm 4.1. We can see that the reduction of the error is better in both the ∞-norm and the 2-norm when using the model based L. Especially in the first two iterations we can see a gain in the convergence rate. What is important to note, however, is that the ILC with the model based L filter becomes very sensitive to delays in the ILC feedback. We have tried to change the L filter, just multiplying with delay operators, and the result is divergence when there is a delay (or inverse delay) of 2 samples. In the current implementation this can be a problem, cf. Section 6.3.

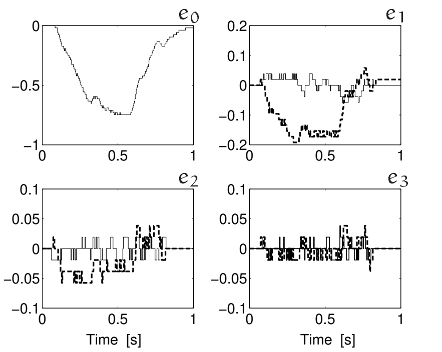
In Figure 7.17b the position error is shown as a function of iteration, the result should be compared with the result shown in Figure 7.6c. The small



a. Comparison of ∞-norm and 2-norm for $L(q) = T_u^{-1}(q)$ (solid) and $L(q) = q^5$ (dashed).



b. Position error as a function of iteration for $L(q) = T_u^{-1}(q)$, cf. Figure 7.6c.



c. Comparison of the position error in iteration 0, 1, 2, and 3 when $L(q) = T_u^{-1}(q)$ (solid) and $L(q) = q^5$ (dotted).

Figure 7.17 *Comparison between the experiments with* $L(q) = T_u^{-1}(q)$ *and* $L(q) = q^5$.

diagrams in Figure 7.17c show the position error in iterations 0, 1, 2 and 3, for the experiments with L according to (7.10) and $L = q^5$. For the error in the first iteration, e_1, the gain by using the model based L is obvious. After three iterations the error is practically the same and we can see the quantization of the signals.

7.7 Summary

We have made a number of experiments showing different properties of the filters in the ILC updating formula. Links to the analysis of ILC are also presented in Section 7.2. The experiments involve, single axis experiment, combined axes experiment, an experiment where one axis has a zero crossing in velocity, and experiments where we change the filters Q and L. The choice of the L filter in most of the experiments is $L(q) = 0.8q^5$. This filter has been found using the design algorithm, Algorithm 4.1, presented in Chapter 4. To explain why the choice of $\delta = 5$ is optimal we have calculated,

$$\|Q(e^{i\omega t_s})(1 - e^{i\omega t_s \delta} T_u(e^{i\omega t_s}))\|_\infty \tag{7.11}$$

where T_u is the model estimated from the system in Section 7.2.3, (7.8). The norm is calculated using the `dhfnorm` command in MATLAB™, cf. Example 2.1. We can see that the minimum is obtained for $\delta = 5$ and this is also

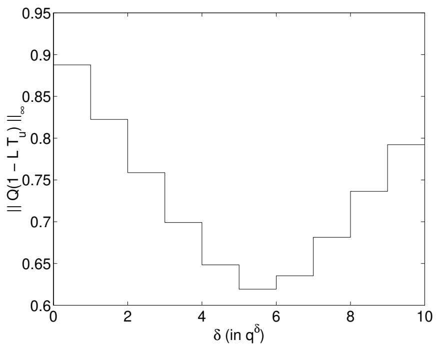


Figure 7.18 *The value of* $\|Q(e^{i\omega t_s})(1 - e^{i\omega t_s \delta} T_u(e^{i\omega t_s}))\|_\infty$ *as a function of* δ.

what we have got from the experiments, using the approach presented in Algorithm 4.1.

The experiment give evidence that the ILC method can be applied to an industrial control system with successful results. In the last experiment the gain in using a model for the design of the ILC is shown. Examining the validity of the linear model in a more global sense is an item for further work.

8

Conclusions

We are now going to summarize, and reflect upon, some of the results and conclusions that we have presented in the thesis. We will also point out some parts in the thesis where further work is needed.

8.1 Summary and Conclusions

In the introduction we stated the goal of the thesis, "to investigate how the properties of an industrial control system can be improved". We have used the *Iterative Learning Control (ILC)* method and applied it to an industrial robot control system from ABB. This has given evidence that it is possible to improve the tracking performance for an industrial system, also when we have limited knowledge on the actual implementation of the controller.

The ILC method is a well known method covered in a vast series of conference and journal articles (some of them are listed in the Bibliography). In the thesis we have given a theory for stability of ILC applied to *linear time invariant (LTI)* systems, the theory includes both definitions and analysis.

In much of the research on ILC, researchers have considered the problem of controlling systems using only the ILC method. We do not have this approach, instead we assume that the system we want to improve is behaving as a servo system. This is one of the key issues and it also explains why the results are as good as they are. The closed loop is well behaved if the controller, already implemented in the industrial control system, is well tuned and well designed. With these assumptions, applying the ILC method is a reasonable task. We can, e.g., use one of the design algorithms presented in Chapter 4.

We have presented two approaches to ILC, the *tracking approach* and the *disturbance rejection approach*. The idea behind the disturbance rejection approach is to reduce the complexity of the ILC problem. The idea stems from an attempt to consider the ILC problem from a system identification perspective and this idea is a topic for more research.

In the implementation of the functions needed to try ILC on the robot control system we have gained a lot of experience. We have used a function that makes it possible to log signals in the DSP, i.e. the lowest level of the control system, and implemented a function called the data-input function that makes it possible to feed back a compensation signal to the system. The feedback is made on a high level, adding a compensation to the position reference signal. The synchronization of the two functions, the data-logger and the data-input has also been implemented in the system. We have, still, some problems with the synchronization and this stems from the fact that the functions altogether contains three processes, two running on the main processor and one on the DSP.

In the experiments we have shown that ILC can be applied successfully to a commercial industrial controller and that we, with the particular robot (an ABB IRB1400), can achieve perfect tracking. An interesting observation is that the ILC designed by the heuristic design algorithm in Algorithm 4.1 works nearly as good as the model based ILC. In the situations where we can afford to run the system three iterations instead of one or two, the heuristic approach gives sufficiently good results, at least for the control of the IRB1400. If the system that we want to control does not have feedback and feedforward controllers that perform as good as they do for the ABB robot, the model based approach might give better results. Important to stress is that with the model we can argue about the robustness of the proposed ILC and this is, of course, in practice a very important aspect.

8.2 Further Work

During the writing of this thesis a lot of new ideas have emerged and there are many open questions for the future. In this section we will point out some of these questions, and ideas, that we want to explore in future work:

- The analysis of the disturbance rejection approach and the applicability to a real problem, like the the robot control system, should be examined. It would also be interesting to extend the class of systems that can be controlled using the disturbance rejection approach. In the thesis we have only considered linear time invariant systems. A possible extension could be to consider systems with static non-linearities on the input or output (Hammerstein and Wiener models respectively). The possibility to compensate for friction could also be investigated more thoroughly.

- The attempt to create a general theory for iterative systems should be more developed. A starting point could be to study the work of, e.g., Hideg [Hid92], and Rogers,Owens [RO92], and try to extend the results found there.

- The results for high-order ILC updating formulas presented in the thesis are very general and do not help very much in the design process for this class of ILC. A theory for high order ILC applied to linear systems would be a major contribution to the area of Iterative Learning Control.

- A challenge is also to compare ILC with what can be achieved with other approaches, e.g., more traditional model based adaptive controllers. What we have seen so far is, of course, that the ILC approach facilitates the ability to improve an already present control system without having to change the internal structure of the controller.

- The synthesis of the ILC updating formula has to be developed. No design algorithms are, e.g., found for high order ILC and for ILC applied to non-linear systems. A major contribution to the synthesis of ILC would be to implement a tool, in for example MATLAB™, where support for the different design algorithms is implemented and where it is possible to try different ILC strategies on models of the true system. With a connection between MATLAB™ and the industrial control system a platform for evaluation of different ILC approaches could be created. This would make it possible to immediately evaluate the chosen ILC on the real system. The first step towards this platform is taken with the implementation discussed in Chapter 6 of the thesis.

- The software in the robot controller and the PC suffers from some problems, discussed in Section 6.2.2. Removing these problems would make it possible to further improve the reliability and testability of the ILC methods on the robot control system.

- The evaluation of the ILC has, in the thesis, only been made on the motor side. It would be interesting to consider the result also on the arm side.
- To really investigate the usefulness of the ILC method, an application, e.g. water cutting, could be chosen. Applying ILC to the system should give experimental results showing the applicability of ILC in a real process.
- An interesting further work would be to make a complete analysis, based on the results in Chapter 3, to explain the results from the experiments in more detail.
- We have applied ILC to the smallest of the robots in the ABB robot family, the IRB1400. Making the same type of experiments on an IRB6400, the largest of the ABB robots, would be an interesting task. New interesting problems might be imposed by the fact that the resonance peak for the large robot is at a lower frequency than for the small robot.
- To be able to use the ILC method in practice the data that we collect from the different iterations have to be compressed. This has not been looked upon yet.

Bibliography

[AKM84a] S. Arimoto, S. Kawamura, and F. Miyazaki. Bettering operation of robots by learning. *Journal of Robotic Systems*, 1(2):123–140, 1984.

[AKM84b] S. Arimoto, S. Kawamura, and F. Miyazaki. Iterative learning control for robot systems. In *Proceedings of IECON*, Tokyo, Japan, October 1984.

[AKMT85] 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, December 1985.

[AN92] S. Arimoto and T. Naniwa. Learning control for motions under geometric endpoint constraint. In *Proc. of the American Control Conference*, pages 2634–2638, 1992.

[AO94] 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.

[AOR95a] 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, 1995.

[AOR95b] 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, 1995.

[AOR97] 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.

[Ari85] 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, Yale University, New Haven, Connecticut, USA, Nov. 1985.

[Ari91] 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.

[AW95] K.J. Aström and B. Wittenmark. *Adaptive Control*. Addison-Wesley Publishing Company, Inc., second edition, 1995.

[BCG88] 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.

[BDG+94] G.J. Balas, J.C. Doyle, K. Glover, A. Packard, and R. Smith. *μ-Analysis and Syntheis Toolbox - For Use with MATLAB*^{TM}. MUSYN Inc. and The MathWorks Inc., 1994.

[Be98] Z. Bien and J.-X. Xu (editors). *Iterative Learning Control: Analysis, Design, Integration and Application*. Kluwer Academic Publishers, 1998.

[BRC97] 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, September 1997.

[CB84] 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.

[CGW98] 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.

[Cra84] J.J. Craig. Adaptive control of manipulators through repeated trials. In *Proc. of ACC*, San Diego, CA, June 1984.

[Cra88] J. J. Craig. *Adaptive Control of Mechanical Manipulators.* Addison-Wesley Publishing Company, 1988.

[CWD97] 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. 10-12 1997.

[CWXS96] Y. Chen, C. Wen, J.-X. Xu, and M. Sun. An initial state learning method for iterative learning control of uncertain time-varying systems. In *Proc. of the 35th Conf. on Decision and Control*, pages 3996–4001, Kobe, Japan, Dec. 1996.

[CXL96a] 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. 1996.

[CXL96b] 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. 1996.

[CXW97] 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. 10-12 1997.

[DB74] G. Dahlquist and Å. Björck. *Numerical Methods.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1974.

[dR96a] 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, 1996.

[dR96b] 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. 1996.

[GK95] G. H. Granlund and H. Knutsson. *Signal Processing for Computer Vision.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1995.

[GN97a] 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 1997.

[GN97b] 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. 1997.

[GN97c] 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, September 1997.

[GS92] K. Guglielmo and N. Sadegh. Motion control of mechanical manipulators. *Journal of Intelligent and Robotic Systems*, 6:17–31, 1992.

[Hau87] J. Hauser. Learning control for a class of nonlinear systems. In *Proc. of the 26-th IEEE Conf. on Decision and Control*, pages 859–860, Los Angles, CA, USA, Dec. 1987.

[HFPM92] G. Heinzinger, D. Fenwick, B. Paden, and F. Miyazaki. Stability of learning control with disturbances and uncertain initial conditions. *IEEE Transactions on Automatic Control*, 37(1):110–114, January 1992.

[Hid92] L.M. Hideg. *Stability of Learning Control Systems.* PhD thesis, Oakland University, 1992.

[HMM91] 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.

[Hor93] R. Horowitz. Learning control of robot manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 115:402–411, June 1993.

[Hor94] R. Horowitz. Learning control applications to mechatronics. *JSME International Journal*, 37(3):421–430, 1994. Series C.

[HYON88] 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.

[JDS83] 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.

[Kai80] T. Kailath. *Linear Systems.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1980.

[KL91] T. Kuc and J. S. Lee. An adaptive learning control of uncertain robotic systems. In *Proc. of the 30th Conf. on Decision and Control*, pages 1206–1211, Brighton, England, December 1991.

[KMA88] S. Kawamura, F. Miyazaki, and S. Arimoto. Realization of robot motion based on a learning method. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):126–134, January/February 1988.

[Liu94] Jing-Sin Liu. Joint stick-slip friction compensation for robotic manipulators by iterative learning. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, volume 1, pages 502–509, 1994.

[Lju87] L. Ljung. *System Identification: Theory for the User*. Prentice-Hall, 1987.

[Lju95] L. Ljung. *System Identification Toolbox - For Use with MATLAB™.* The MathWorks Inc., 1995.

[LL93] 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.

[LS88] J. N Little and L Shure. *Signal Processing Toolbox - For Use with MATLAB^{TM}*. The MathWorks Inc., 1988.

[Lya92] A. M. Lyapunov. *The General Problem of the Stability of Motion*. Taylor & Francis, 1992. An English translation of Lyapunov's major study of the general problem of the stability of motion, first published in Russian in 1892.

[MK85] 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, December 1985.

[MM94] T. Manabe and F. Miyazaki. Learning control based on local linearization by using dft. *Journal of Robotic Systems*, 11(2):129–141, 1994.

[Moo93] K. L. Moore. *Iterative Learning Control for Deterministic Systems*. Advances in Industrial Control. Springer-Verlag, 1993.

[Moo98] K. L. Moore. Iterative learning control - an expository overview. Technical Report T.R. 97/98 002, Measurement and Control Engineering Research Center, College of Engineering, Idaho State University, 1998. To appear in Applied and Computaional Controls, Signal Processing, and Circuits (late 1997 or early 1998).

[NG97] M. Norrlöf and S. Gunnarsson. Using iterative learning control to get better performance of robot control systems. In *Proceedings, robotikdagarna 1997*, Department of Mechanical Engineering, Linköping University, 1997.

[NG98] 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.

[Nil96] K. Nilsson. *Industrial Robot Programming.* PhD thesis, Department of Automatic Control, Lund Institute of Technology, May 1996.

[OM97] 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.

[PH93] 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. IFAC.

[Pla68] J. B. Plant. *Some Iterative Solutions in Optimal Control.* Number 44 in Research monograph. The M.I.T. Press, Cambridge, Massachusetts, 1968.

[RO92] 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.

[Rud76] W. Rudin. *Principles of Mathematical Analysis.* McGRAW-HILL, third edition, 1976.

[SG92] N. Sadegh and K. Guglielmo. Design and implementation of adaptive and repetitive controllers for mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 8(3):395–400, June 1992.

[Sjö95] J. Sjöberg. *Non-Linear System Identification with Neural Networks.* PhD thesis, Linköping University, 1995.

[SL91] J.-J. E. Slotine and W. Li. *Applied Nonlinear Control.* Prentice Hall, Englewood Cliffs, New Jersey 07632, 1991.

[SLA92] M. W. Spong, F. L. Lewis, and C. T. Abdallah, editors. *Robot Control: Dynamics, Motion Planning, and Analysis.* IEEE Control Systems Society, IEEE Press, 1992.

[SM98] D. Sun and J. K. Mills. Adaptive learning control of robotic systems with model uncertainties. In *Proc. of the 1998 IEEE Int. Conf. on Robotics and Automation*, pages 1847–1852, Leuven, Belgium, May 1998.

[Sta92] W. Stallings. *Operating Systems*. Maxwell MacMillan International, 1992.

[SV89] M. W. Spong and M. Vidyasagar. *Robot Dynamics and Control*. Wiley, 1989.

[TTC89] 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.

[TY85] 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.

[Uch78] 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.

[ZDG96] K. ZHou, J.C. Doyle, and K. Glover. *Robust and Optimal Control*. Prentice Hall, Upper Saddle River, New Jersey 07458, 1996.

Subject Index