Linköping studies in science and technology. Dissertations. No. 1358, Linköping 2011

Estimation-based iterative learning control

Johanna Wallén

Linköping University
INSTITUTE OF TECHNOLOGY

Linköping studies in science and technology. Dissertations.
No. 1358

Estimation-based iterative learning control

Johanna Wallén

Department of Electrical Engineering
Linköping University, SE–581 83 Linköping, Sweden

Linköping 2011

Cover illustration: ILC applied to a problem where, for example, a robot tool is supposed to track a circular path. In the beginning, the tracking performance is poor, but as the ILC algorithm "learns", the performance improves and comes very close to a perfect circle. The orange colour represents the connection to the experiments performed on ABB robots.

Linköping studies in science and technology. Dissertations.
No. 1358

**Estimation-based
iterative learning control**

Johanna Wallén

johanna@isy.liu.se
www.control.isy.liu.se
Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE–581 83 Linköping
Sweden

ISBN 978-91-7393-255-4 ISSN 0345-7524

Printed by LiU-Tryck, Linköping, Sweden 2011

Till Daniel, familjen och vännerna

Abstract

In many applications industrial robots perform the same motion repeatedly. One way of compensating the repetitive part of the error is by using iterative learning control (ILC). The ILC algorithm makes use of the measured errors and iteratively calculates a correction signal that is applied to the system.

The main topic of the thesis is to apply an ILC algorithm to a dynamic system where the controlled variable is not measured. A remedy for handling this difficulty is to use additional sensors in combination with signal processing algorithms to obtain estimates of the controlled variable. A framework for analysis of ILC algorithms is proposed for the situation when an ILC algorithm uses an estimate of the controlled variable. This is a relevant research problem in for example industrial robot applications, where normally only the motor angular positions are measured while the control objective is to follow a desired tool path. Additionally, the dynamic model of the flexible robot structure suffers from uncertainties. The behaviour when a system having these difficulties is controlled by an ILC algorithm using measured variables directly is illustrated experimentally, on both a serial and a parallel robot, and in simulations of a flexible two-mass model. It is shown that the correction of the tool-position error is limited by the accuracy of the robot model.

The benefits of estimation-based ILC is illustrated for cases when fusing measurements of the robot motor angular positions with measurements from an additional accelerometer mounted on the robot tool to form a tool-position estimate. Estimation-based ILC is studied in simulations on a flexible two-mass model and on a flexible nonlinear two-link robot model, as well as in experiments on a parallel robot. The results show that it is possible to improve the tool performance when a tool-position estimate is used in the ILC algorithm, compared to when the original measurements available are used directly in the algorithm. Furthermore, the resulting performance relies on the quality of the estimate, as expected.

In the last part of the thesis, some implementation aspects of ILC are discussed. Since the ILC algorithm involves filtering of signals over finite-time intervals, often using non-causal filters, it is important that the boundary effects of the filtering operations are appropriately handled when implementing the algorithm. It is illustrated by theoretical analysis and in simulations that the method of implementation can have large influence over stability and convergence properties of the algorithm.

Populärvetenskaplig sammanfattning

Denna avhandling behandlar reglering genom iterativ inlärning, ILC (från engelskans iterative learning control). Metoden har sitt ursprung i industrirobottillämpningar där en robot utför samma rörelse om och om igen. Ett sätt att kompensera för felen är genom en ILC-algoritm som beräknar en korrektionssignal, som läggs på systemet i nästa iteration. ILC-algoritmen kan ses som ett komplement till det befintliga styrsystemet för att förbättra prestanda.

Det problem som särskilt studeras är då en ILC-algoritm appliceras på ett dynamiskt system där reglerstorheten inte mäts. Ett sätt att hantera dessa svårigheter är att använda ytterligare sensorer i kombination med signalbehandlingsalgoritmer för att beräkna en skattning av reglerstorheten som kan användas i ILC-algoritmen. Ett ramverk för analys av skattningsbaserad ILC föreslås i avhandlingen. Problemet är relevant och motiveras utifrån experiment på både en seriell och en parallel robot. I konventionella robotstyrsystem mäts endast de enskilda motorpositionerna, medan verktygspositionen ska följa en önskad bana. Experimentresultat visar att en ILC-algoritm baserad på motorpositionsfelen kan reducera dessa fel effektivt. Dock behöver detta inte betyda en förbättrad verktygsposition, eftersom robotmotorerna styrs mot felaktiga värden på grund av att modellerna som används för att beräkna dessa referensbanor inte beskriver den verkliga robotdynamiken helt.

Skattningsbaserad ILC studeras både i simulering av en flexibel tvåmassemodell och en olinjär robotmodell med flexibla leder, och i experiment på en parallell robot. I studierna sammanvägs motorpositionsmätningar med mätningar från en accelerometer på robotverktyget till en skattning av verktygspositionen som används i ILC-algoritmen. Resultaten visar att det är möjligt att förbättra verktygspositionen med skattningsbaserad ILC, jämfört med när motorpositionsmätningarna används direkt i ILC-algoritmen. Resultatet beror också på skattningskvaliteten, som förväntat.

Slutligen diskuteras några implementeringsaspekter. Alla värden i uppdateringssignalen läggs på systemet samtidigt, vilket gör det möjligt att använda ickekausal filtering där man utnyttjar framtida signalvärden i filteringen. Detta gör att det är viktigt hur randeffekterna (början och slutet av signalen) hanteras när man implementerar ILC-algoritmen. Genom teoretisk analys och simuleringsexempel illustreras att implementeringsmetoden kan ha stor betydelse för egenskaperna hos ILC-algoritmen.

Acknowledgments

The idea of iterative learning control is appealing and it often works very well in practice. Crucial for the resulting performance is however that the reference is a good one. Mikael Norrlöf and Svante Gunnarsson, you have been my path planners and trajectory generators. Your exceptional patience and efforts to keep me on the right track are much appreciated! You have also been my sensors, measuring the performance, and my ILC algorithm, by calculating a new correction signal so that I could come closer to the desired output in the next iteration.

An important part of being able to track the correct reference is also everyone in the Automatic Control group. The knowledge, curiosity and nice atmosphere have been worth a lot for me, thank you all! My special thanks go to Lennart Ljung, who let me join the group, and to Svante Gunnarsson, Ulla Salaneck and Åsa Karmelind for keeping track of all things in an excellent manner.

Proofreading means many iterations, where mistakes and ambiguities are to be corrected. First of all, I am very grateful for the enormous efforts Mikael Norrlöf and Svante Gunnarsson have made to improve the text. The excellent comments and questions from proofreading by Daniel Ankelhed, Daniel Axehill, Daniel Petersson, Martin Enqvist, Michael Roth, Patrik Axelsson and Ylva Jung have also improved the thesis considerably. Thank you all! Henrik Tidefelt and Gustaf Hendeby, thanks a lot for solving many of my LaTeX problems.

I am also grateful for the funding from VINNOVA, the Swedish Research Council (VR) and from the excellence centers LINK-SIC and ELLIIT, which made it possible for me to perform new iterations to improve the research. During my visits in Lund in the spring 2010 I learned a lot — special thanks to Isolde Dressler and Anders Robertsson for good cooperation! Torgny Brogårdh and the people at ABB Robotics, thank you for bringing an industrial point of view to my reference trajectory.

Finally, family and friends, thank you for the patience while I was too busy following the desired trajectory! And Daniel, without you it would have been much, much harder!

Linköping, January 2011

Johanna Wallén

Contents

Notation

Symbols and operators

Notation	Meaning
$a(t)$	Acceleration
$C(q, \dot{q})$	Vector of Coriolis and centripetal torques
d_b^a	Translation of origin of coordinate frame a relative to origin of coordinate frame b
D	Damping matrix
$e(t)$	Control error
$\epsilon(t)$	Error used in ILC update equation
η	Gear ratio
f_c	Cutoff frequency
$F(\dot{q})$	Friction torque
$g_X(n)$	Pulse-response coefficient n of transfer operator $X(q)$
$G(q)$	Vector of gravity torques
I, $I_{N\times N}$	Identity matrix (with N rows and columns)
J	(Manipulator) Jacobian
k	Iteration number
K	Observer gain
$K(q)$	Stiffness matrix
$L(q)$	Filter in ILC algorithm
$\boldsymbol{L}$	Matrix in ILC algorithm in matrix form
$M(q)$	Inertia matrix

Symbols and operators (continued)

Notation	Meaning
N	Number of samples (batch length)
ω	Angular frequency
p	Differential operator
Π	Matrix of gear ratios
q	Joint variable
q	Time-shift operator, $qx(t) = x(t + T_s)$
q_a	Joint angular position
q_k	Iteration-shift operator, $qx_k(t) = x_{k+1}(t)$
q_m	Motor angular position
$Q(q)$	Filter in ILC algorithm
$\boldsymbol{Q}$	Matrix in ILC algorithm in matrix form
$r(t)$	Reference signal
$r_a(t)$	Joint angular position reference
$r_m(t)$	Motor angular position reference
R_b^a	Rotation matrix from coordinate frame a to coordinate frame b
t	Time
τ	Torque
τ_m	Motor torque
T_s	Sampling interval
$u_k(t)$	ILC input signal at iteration k
$v(t)$	Velocity
$v(t)$	Process noise
$w(t)$	Measurement noise
$x(t)$	State vector
$y(t)$	Measured variable
$z(t)$	Controlled variable
$z_c(t)$	Calculated tool position obtained by transforming motor angular positions by the forward kinematics
$0_{N\times M}$	Matrix of zeros with N and M columns
A^T	Transpose of matrix A
A^{-1}	Inverse of matrix A
A^*	Adjoint of matrix A
$\frac{\partial G(x)}{\partial x}$	Partial derivative of $G(x)$ with respect to x
$G_{ab}(q)$	Transfer operator relating signal $a(t)$ to signal $b(t)$, $b(t) = X_{ab}(q)a(t)$
$G^0(q)$	Transfer operator of true system
$x_k(t)$	Signal $x(t)$ at iteration k
x_∞	Limit of x_k as $k \to \infty$
$\hat{x}(t)$	Estimate of signal x(t)
$\hat{x}(t\|t-1)$	Estimate of x(t) given from measurements up to and including time $t-1$, also denoted $\hat{x}_{t\|t-1}$

Symbols and operators (continued)

Notation	Meaning
$\boldsymbol{x}_k$	Vector $\boldsymbol{x}$ at iteration k of the N-sample sequence of x(t)
x^T	Transpose of vector x
$\dot{x}(t)$	Time derivative of signal x
$\|x\|$	Absolute value of complex variable x
$X(e^{i\omega})$	Discrete-time Fourier transform of x(t)
$\rho(\cdot)$	Spectral radius
$\bar{\sigma}(\cdot)$	Maximum singular value
$\|\cdot\|$	Norm
$\|\cdot\|_2$	2-norm
$\|\cdot\|_\infty$	∞-norm
$\|\cdot\|_A$	Weighted norm with weighting matrix A

Abbreviations

Abbreviation	Meaning
ARX	Autoregressive with external input
CITE-ILC	Current-iteration tracking-error ILC
CNC	Computer numerical controlled (machine)
D	Derivative (controller)
DOF	Degrees of freedom
DTFT	Discrete-time Fourier transform
EKF	Extended Kalman filter
I	Integral (controller)
ILC	Iterative learning control
MIMO	Multiple-input multiple-output (system)
NC	Numerically controlled (machine)
P	Proportional (controller)
PID	Proportional, integral, derivative (controller)
PD	Proportional, derivative (controller)
PRBS	Pseudo random binary sequence
RC	Repetitive control
RMS	Root mean square (error)
SISO	Single-input single-output (system)
TCP	Tool center point

Part I

Background

1 Introduction

IN THIS WORK the general framework for iterative learning control (ILC) is extended to cover estimation-based ILC. The extension is motivated by the use of ILC in robotics, where the desired control error cannot be computed from measured variables. Instead it is necessary to use estimation techniques to find an estimate of the error, which then can be used in the ILC algorithm.

1.1 Motivation and problem statement

The method of iterative learning control, or ILC for short, originates from industrial robotics, where in many applications the same trajectory is repeated. The idea in ILC is to improve the performance of a system, machine or process by adding a correction signal derived from the errors at previous repetitions. The word 'iterative' reflects the recursive nature of the system operation, while 'learning' denotes that the input signal is updated, or learned, from the system performance in the past [Moore, 1998a].

In particular, systems containing mechanical flexibilities are considered in the thesis. When applying an ILC algorithm to a system, it is often already stabilised by feedback control. Still, it may be difficult to follow the desired trajectory due to unmodelled effects or disturbances. One way to understand some of the difficulties of a flexible mechanical system is to first think of the system to be controlled as a stiff metal bar. Consider the case when it is important that the tip follows a desired trajectory as closely as possible with ideally no oscillations or other disturbing phenomena. The only way to control the tip is by the movements of the hand (the "motor") holding the bar. For the metal bar, the movement of the tip is only a scaled version of the movement of the hand. Therefore the desired

trajectory can be followed satisfactorily. Now, instead consider the system as a very flexible fishing rod. To be able to follow the desired trajectory, one has to derive a complete mental model of how the tip of the rod moves for every single movement of the hand in every single part of the trajectory. In reality, the tip can by appropriate hand movements only come fairly close to the desired trajectory, since the model is only partially known. The tip of the rod will however still oscillate, especially for rapid movements, due to the flexible structure of the rod.

The problems when controlling an industrial robot has similarities to the control problem of the fishing rod. The sensor measurements normally available in commercial industrial robot systems are only the motor angular positions, while it is the resulting tool position and orientation that are of interest for the user. The cost- and performance-driven development of the robots results in less rigid mechanical robot structures. For example, the robot weight to payload ratio for large-size ABB robots has been reduced by a factor of three since the mid 1980s from 16:1 to around 5.5:1 of today [Moberg, 2010]. This results in robots having a larger number of mechanical vibration modes and also lower resonance frequencies. Due to the complex structure of the robot, there are uncertainties in both kinematic and dynamic models of the robot. Therefore, when programming the robot to follow a desired tool trajectory, some tool-path errors will still remain.

Despite the robot control challenges described above, there are constantly increasing demands on the desirable robot performance and functionality of the robot motion control, as is discussed in Brogårdh [2007, 2009]. A remedy for handling this situation could be to use ILC when performing the same movement repeatedly, to be able to correct the repetitive part of the remaining errors. Applying an ILC algorithm to the robot using the measurements normally available in the robot system, the motor angular positions, will however not entirely solve the problem due to the model uncertainties. A solution to this problem is discussed in this thesis, where estimates of the controlled variable (the tool position in the case of an industrial robot) are obtained from estimation algorithms. These estimates are thereafter used in an ILC algorithm to improve the system performance. The usage of estimation-based ILC is the main theme in this thesis, and the following aspects are analysed:

- The resulting error of the controlled variable when using different types of estimates in the ILC algorithm.
- The benefits of using estimation-based ILC instead of relying only on the original measurements available. This is illustrated in both simulations and experiments.

In the thesis some implementation aspects of ILC are also treated, which highlights the importance of correct handling of the boundary effects in the implementation of the non-causal ILC filters. The effects of the filter implementation are analysed by a time-domain matrix formulation of the ILC system.

Throughout this work the ILC algorithm works as a complement to the already existing controller of the system. This setting can be motivated from an appli-

cation point of view, where it is in general not possible for the user to affect the control system, and therefore it has to be considered as given [Longman, 2000]. The focus is on discrete-time systems and ILC algorithms, motivated by practical implementation involving computer-based controllers operating in discrete time and digital storage of the information. First-order, linear, time- and iteration-invariant ILC algorithms are investigated.

1.2 Outline

The thesis consists of two parts. Part I gives an overview of the subjects treated in the thesis and serves as a unified background to the results presented in Part II.

1.2.1 Outline of Part I

First, an introduction to the application, industrial robots, is given in Chapter 2. The field of industrial robots is presented both mathematically, as well as historically, to bring some understanding to the challenges that are discussed in this work. In Chapter 3 some estimation techniques are briefly discussed, with focus on Kalman filtering. It is followed by an introduction to the field of ILC in Chapter 4, with an overview of different types of algorithms, convergence properties, design methods and applications.

1.2.2 Outline of Part II

Part II begins in Chapter 5 with an experimental motivation to the problem of improving the performance of the controlled variable of a flexible mechanical system with ILC using measured variables only. The main problem is illustrated in a simulation study. A framework for analysis of estimation-based ILC is then given in Chapter 6, with focus on the resulting performance of the controlled variable of the system. Thereafter, the possibilities of estimation-based ILC are shown by simulations on a flexible nonlinear two-link robot model in Chapter 7 and by experiments on a large-size parallel robot in Chapter 8. In Chapter 9 some implementation aspects are analysed of how the handling of the boundary effects will affect the convergence properties of the ILC algorithm. Finally, Chapter 10 concludes the work and gives possible directions for future work.

1.3 Contributions

The main contributions of the thesis are:

- Experiments performed on a large-size commercial industrial robot with an ILC algorithm using measured motor angular positions. An approach as simple as possible is used, resulting in a substantial error reduction after only a few iterations. The results are discussed in Chapter 5, and published in Wallén et al. [2007a].
- Illustration of the problem of applying ILC algorithms using the measured

variable to flexible systems, where the controlled variable is not the measured variable. This is the topic for Chapter 5, and is also partly discussed in Wallén et al. [2008b, 2009a,b, 2010a,b].

- A framework for analysis of estimation-based ILC, previously presented in Wallén et al. [2009c, 2011] and covered in Chapter 6.
- Illustration of the benefits of using estimation-based ILC, both in simulations of a flexible nonlinear two-link robot model in Chapter 7, published in Wallén et al. [2009a], and in experiments on a large-size parallel robot in Chapter 8, published in Wallén et al. [2010a,b].
- Discussion of the importance of correct handling of the boundary effects in the implementation of non-causal ILC filters. The effects of the filtering are analysed by the time-domain matrix formulation and put in contrast to the frequency-domain analysis for a case where it is not enough with a frequency-domain approach. This is the topic for Chapter 9, and is previously published in Wallén et al. [2010].

1.3.1 Publications

The thesis is based on the following publications, where the author is the main contributor:

Johanna Wallén, Mikael Norrlöf, and Svante Gunnarsson. Experimental evaluation of ILC applied to a six degrees-of-freedom industrial robot. In *Proceedings of European Control Conference*, pages 4111–4118, Kos, Greece, July 2007a.

Johanna Wallén, Mikael Norrlöf, and Svante Gunnarsson. Arm-side evaluation of ILC applied to a six-degrees-of-freedom industrial robot. In *Proceedings of IFAC World Congress*, pages 13450–13455, Seoul, Korea, July 2008b. Invited paper.

Johanna Wallén, Mikael Norrlöf, and Svante Gunnarsson. Performance of ILC applied to a flexible mechanical system. In *Proceedings of European Control Conference*, pages 1511–1516, Budapest, Hungary, August 2009b.

Johanna Wallén, Svante Gunnarsson, Robert Henriksson, Stig Moberg, and Mikael Norrlöf. ILC applied to a flexible two-link robot model using sensor-fusion-based estimates. In *Proceedings of IEEE Conference on Decision and Control*, pages 458–463, Shanghai, China, December 2009a.

Johanna Wallén, Mikael Norrlöf, and Svante Gunnarsson. A framework for analysis of observer-based ILC. In *Proceedings of Symposium on Learning Control at IEEE Conference on Decision and Control*, Shanghai, China, December 2009c.

Johanna Wallén, Mikael Norrlöf, and Svante Gunnarsson. A framework for analysis of observer-based ILC. *Asian Journal of Control*, 2011. Accepted for publication in special issue of Iterative Learning Control.

Johanna Wallén, Svante Gunnarsson, and Mikael Norrlöf. Some implementation aspects of iterative learning control. Technical Report LiTH-ISY-R-2967,

Department of Electrical Engineering, Linköping University, Linköping, Sweden, September 2010. Submitted to IFAC World Congress 2011, Milano, Italy.

Johanna Wallén, Isolde Dressler, Anders Robertsson, Mikael Norrlöf, and Svante Gunnarsson. Observer-based ILC applied to the Gantry-Tau parallel kinematic robot — modelling, design and experiments. Technical Report LiTH-ISY-R-2968, Department of Electrical Engineering, Linköping University, Linköping, Sweden, October 2010a.

Johanna Wallén, Isolde Dressler, Anders Robertsson, Mikael Norrlöf, and Svante Gunnarsson. Observer-based ILC applied to the Gantry-Tau parallel kinematic robot. Submitted to IFAC World Congress 2011, Milano, Italy, 2010b.

Parts of the material have previously been published in

Johanna Wallén. *On Kinematic Modelling and Iterative Learning Control of Industrial Robots*. Licentiate thesis No. 1343, Department of Electrical Engineering, Linköping University, Linköping, Sweden, January 2008. Available at: `http://www.control.isy.liu.se/research/reports/LicentiateThesis/Lic1343.pdf`.

1.3.2 Related publications

Other publications of related interest, but not included in the thesis are:

Johanna Wallén, Svante Gunnarsson, and Mikael Norrlöf. Derivation of kinematic relations for a robot using MAPLE. In *Proceedings of Reglermöte 2006*, Royal Institute of Technology, Stockholm, Sweden, May 2006.

Johanna Wallén. On robot modelling using MAPLE. Technical Report LiTH-ISY-R-2723, Department of Electrical Engineering, Linköping University, Linköping, Sweden, August 2007.

Johanna Wallén, Mikael Norrlöf, and Svante Gunnarsson. Accelerometer based evaluation of industrial robot kinematics derived in MAPLE. In *Proceedings of Mekatronikmöte 2007*, Lund Institute of Technology, Lund, Sweden, October 2007b.

Johanna Wallén, Mikael Norrlöf, and Svante Gunnarsson. Comparison of performance and robustness for two classical ILC algorithms applied to a flexible system. Technical Report LiTH-ISY-R-2868, Department of Electrical Engineering, Linköping University, Linköping, Sweden, November 2008a.

Johanna Wallén, Mikael Norrlöf, and Svante Gunnarsson. Performance and robustness for ILC applied to flexible systems. In *Proceedings of Reglermöte 2008*, Luleå University of Technology, Luleå, Sweden, June 2008c.

2

Industrial robots

THE AIM OF THIS CHAPTER is twofold. First, to shed some light on the development of industrial robots in order to understand the industrial usage and future challenges. Second, to give an introduction to robot modelling and control. The focus is on industrial robots in general, not on special applications.

2.1 Industrial robots and robotics

The word robota exists in several Slavic languages, meaning work. The Czech playwright Karel Čapek wrote R.U.R. (Rossum's Universal Robots), which had its premier in 1921. By that the word robot was born. In the play the humans are served by robots characterised by super human strength and intelligence, but having no feelings or intellectual life. However, after a while the robots revolt and kill the master Rossum and destroy all life on Earth. The play resulted in people thinking of robots as something negative [Bolmsjö, 1992].

Three fundamental laws of robots were formulated by Isaac Asimov, a Russian writer of science fiction, in 1942:

1. "a robot may not injure a human being, or, through inaction, allow a human being to come to harm."
2. "a robot must obey the orders given it by human beings except where such orders would conflict with the First Law."
3. "a robot must protect its own existence as long as such protection does not conflict with the First or Second Law." [Asimov, 1942]

The perspective here is, in contrast to Čapek's robot, a kind-hearted mechanical creation of human appearance, serving human beings [Bolmsjö, 1992]. The be-

haviour is dictated by a "brain" programmed by human beings, satisfying certain ethical rules. The laws were complemented by Asimov's zeroth law in 1985:

0. "a robot may not injure humanity, or, through inaction, allow humanity to come to harm." [Asimov, 1985]

Recently, the laws have been complemented by two more laws for industrial robots by Stig Moberg [NyTeknik, 2007], were the robot motion is considered:

4. a robot must follow the path specified by its master, as long as it does not conflict with the first three laws.
5. a robot must follow the velocity and acceleration specified by its master, as long as nothing is blocking the path and it does not conflict with the other laws. (Translated from Swedish by the author. [NyTeknik, 2007, p. 8])

The definition of industrial robots, formulated by the International Organization for Standardization (ISO), used by most of the robot organisations is:

> Manipulating industrial robot is an "automatically controlled, reprogrammable, multi-purpose manipulator programmable in three or more axes which may be either fixed in place or mobile for use in industrial automation applications". [ISO, 1996, p.5]

The keyword in the definition is "reprogrammable", which makes the robot adaptive to a variety of types of tasks without physically rebuilding the machine. The robot shall also have memory and logic to be able to work independently and automatically. Industrial robotics is a discipline concerning robot design and control, where the products are reaching the level of a mature technology with many industrial applications [Bolmsjö, 1992, Sciavicco and Siciliano, 2000].

2.2 Development of industrial robots

A historical journey in the footsteps of industrial robots is given, together with an introduction to the field of industrial robotics. The main references are Westerlund [2000] and Bolmsjö [1992], if nothing else is stated.

Automation

Ever since the industrial revolution in the 18th century, automation has been a major force in rationalising the production. Until half a century ago, automation was almost synonymous to mechanisation. The drawbacks with mechanisation are large costs and rigid equipment — for every new product the whole production line has to be rebuilt. Therefore, automation is mainly suited for mass production, like the car industry, with the assembly lines at Ford being a famous historical example. When the technical director at Ford aimed at a completely automatic production line in 1946, it led to a world-wide debate on automation. The critics were afraid that automation was meant to rationalise the workers out of production, leading to mass unemployment. On the other hand, the supporters believed in that automation would lead to a second industrial revolution.

In industry, the hydraulic assembly machines and the numerically controlled (NC) milling machines arrived in the 1950s, from which one can say that the industrial robot originates technically. With the birth of the computers in the 1950s, the computer numerical controlled (CNC) machines could be developed. Machines were equipped with increasingly sophisticated digital control units and some systems became integrated with each other by a central control unit in the late 1960s. With the development of the integrated circuit in the 1970s, the manufacturing production could be more flexible.

The first robots

In 1956 George Devol and Joseph Engelberger met on a party in Connecticut. Devol had a patent on a machine called Programmed Transfer Article, but was however uncertain about how the machine actually could be used. Engelberger was a space industry engineer fond of science fiction and Isaac Asimov's books. They started the company Unimation to produce industrial robots. Devol and Engelberger started by visiting a number of factories to better understand the needs regarding industrial robots, and the first prototype came in 1961. The first robot, installed in one of General Motors' factories to serve a die casting machine, could only perform one task and had an expected life-span of 18 months. The breakthrough came after installing 66 robots in a new factory of General Motors in 1964. The manufacturing industry was still not very interested in robots, but media was. Engelberger and his robots were regular guests on American television, and the robot changed from being a scary object to a "funny toy" to the general public.

In order to increase the robot production, the technical director at Ford copied the Unimation robot specification and sent it to other companies for production. With that movement many American companies saw the potential of robots and entered the robotics industry. However, the industrial robot industry could not be seen as a separate unit until the middle of the 1970s, when research results were ready for commercialisation. With new electronic components, especially the microprocessor, the basis of the control systems of today was formed. From the second half of the 1970s, the sale of industrial robots grew rapidly with a yearly growth exceeding 30 %.

The Swedish company ASEA (later ABB) started their robotic era by using NC machines and advanced production techniques in the 1960s. ASEA had its own development of NC control systems, which became a forerunner to their robot control systems. After loosing the contract of manufacturing Unimation robots on license to the Swedish company Electrolux in 1969, ASEA decided in 1971 to develop robots. An electrically driven robot was developed, where the control program of the first prototype, named IRB6, was run on the new Intel 8008 microprocessor. The first customer was Magnussons i Genarp AB, a small Swedish family firm with 20 employees. By the installation of the robot, the firm was among the first in the world to operate an unmanned factory 24 hours per day, seven days a week. Both IRB6 and the next model IRB60, introduced in 1975, were included in the ASEA product range for as long as 17 years.

Robotic development

In the 1970s, material handling was the main area for robots, which requires sufficient load capacity of the robot. In the end of the 1970s, the focus of the development was turned towards assembly, with needs for robots with higher velocity, acceleration and better repeatability in order to shorten the cycle times. The automotive industry was, and still is, an important customer, but also metal industry with their hazardous working environment became an increasing robot user. In the 1980s, relatively simple tasks, like machine tending, material transfer, painting and welding became economically viable for robotisation [Craig, 1989].

Robots are developed not only for manufacturing, but also for medical tasks, service, search and rescue, which brings new interest into robotics. The development performed from the latter part of the 1980s until now has among many things involved advanced sensors [Spong et al., 2006], as for example vision systems, laser scanners or force sensors. Additionally, model-based control is a key competence for robot manufacturers, giving a drastically increase in the robot performance over the years [Björkman et al., 2008]. As a result, robots can be used in demanding applications, like water jet and laser cutting, gluing and deburring, where accuracy at high speed is crucial [Brogårdh, 2009].

Industrial robots in the future

The discussion about future robot development and possible applications starts with a picture of the situation of today. The operational stock of industrial robots for 2005 and 2009 is presented in Table 2.1 for some selected geographical areas. Different countries can have slightly different definitions of industrial robots, with for example Japan also counting simpler mechanical machines. Despite this, Table 2.1 undoubtedly shows that Japan is the largest robot user in the world followed by Germany. The total worldwide stock of industrial robots in operation was at the end of 2009 approximately 1 300 000 units, based on an average life-span of 15 years. However, Japan has seen a continuous decline in robot investments since 2006. Due to the worldwide economic crisis starting in the autumn of 2008, the total number of robot installations dropped substantially from a continuously increased high level in 2005 to 2008 to the lowest number reported since 1994 [International Federation of Robotics, 2010].

Already today the robot market for many applications in the automotive industry is saturated, and the impact on the robot development is reduced. Since it is difficult to directly use these robot applications in other areas with different challenges, further development is necessary. An important aspect in the development is also the high cost pressure, which forces the robot manufacturers to use robot components with larger individual variations, nonlinearities and noise levels [Brogårdh, 2007, 2009]. The constantly increasing requirements on robot performance is solved by using more complex model-based multivariable control methods. Future products could also include iterative learning control (ILC), with the work in this thesis showing the potential of using robot tool-position estimates in the ILC algorithm to improve the performance of the robot tool.

Table 2.1: *Operational stock of industrial robots for selected countries for 2005 and 2009 [International Federation of Robotics, 2006, 2010]. The numbers within parenthesis refer to the position in the ranking for each continent.*

Geographic area	Units 2005	Units 2009
Africa	**634**	**1 973**
America	**143 203**	**172 141**
North America (Canada, Mexico, USA)	139 553	166 183
Asia/Australia	**481 664**	**501 422**
Japan(1)	373 481	332 720
Republic of Korea(2)	61 576	79 003
China(3)	11 557	37 312
Europe	**297 374**	**343 661**
Germany(1)	126 725	144 133
Italy(2)	56 198	62 242
France(3)	30 434	34 099
Spain(4)	24 081	28 781
Sweden(7)	8 028	9 396

Automation by robots handling small lot series in organisations with lacking infrastructure for robot automation and limited economical resources for robot investment is a challenge to be solved in the future. This requires lower robot cost as well as development of tools for configuration, calibration, programming and maintenance. In order to increase the usage of industrial robots, increased sensor-based control will be necessary both for higher performance, lower robot cost and for automation of new applications. An increasing amount of sensors also means new possibilities to develop safe human/robot interaction. One example is newly developed light-weight robot structures, for example the robot LWR III seen in Figure 2.1a, having a robot weight to payload ratio close to 1:1 and where the robot is designed for mobility and interaction with humans. An increasing interest is also shown to fault detection/isolation and diagnosis in combination with more sensors, for service of robot components before breakdown [Brogårdh, 2009].

Very large components are machined in the aerospace industry, nowadays performed manually or by expensive Cartesian robots. The high accuracy requirements may in the future be satisfied by parallel robots [Brogårdh, 2009]. One example of a mechanical structure designed for such applications can be seen in Figure 2.1d. In Chapter 8 improved performance of a parallel robot is achieved by using ILC based on estimates of the robot tool position.

(a) LWR III

(b) IRB6600

(c) IRB340 FlexPicker

(d) Gantry-Tau prototype

Figure 2.1: *Examples of robots: a) light-weight robot with carbon-fibre links, b) large-size robot for heavy industrial applications, c) parallel robot for pick-and-place applications, d) parallel robot with large workspace. Photo a) from DLR [2010], b), c) from ABB Robotics [2007] and d) from SMErobot [2010].*

2.3 Modelling

Robot control, especially model-based robot motion control, is in Brogårdh [2007, 2009] stated to be a key competence in the development of industrial robots. A prerequisite for design of control laws is accurate kinematic and dynamic models of the robot system. Mathematical models of the robot are also important in mechanical design, performance simulation, offline programming, diagnosis and supervision, among others.

This overview of robot modelling starts with a description of the mechanical structure of the robot, then kinematics modelling is described, followed by rigid and flexible dynamic models. Finally, a short summary of the control system and examples of control strategies are given. General references for the remainder of the chapter are Spong et al. [2006], Sciavicco and Siciliano [2000] and Tsai [1999]. For brevity, the time dependence on the variables is omitted in the chapter.

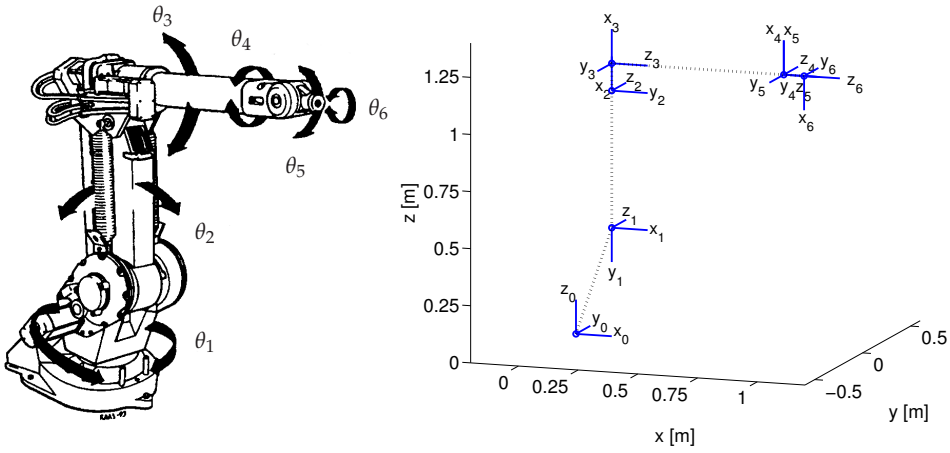
2.3.1 Basic concepts about robots

The individual bodies that together form a robot are called links, connected by joints. The robot links form a kinematic chain, where the base of the robot is defined as link 0, and the other end of the chain is connected to an end effector or tool. When the kinematic chain is open, every link is connected to every other link by only one chain of links. A robot having this type of mechanical structure is called a serial link robot, or serial robot for short. If a sequence of links forms one or more loops, the robot contains closed kinematic chains, resulting in a parallel robot[1]. In Figure 2.1 some examples of serial and parallel robots are shown, with the ABB robot IRB6600 and the light-weight robot LWR III from DLR both having an open kinematic chain. Robots with closed kinematic chains are exemplified by the robot IRB340 and the Gantry/Tau robot prototype in the figure. At present, the serial robot is the most common kinematic structure.

Most industrial robots have six degrees of freedom (DOFs), which makes it possible to control both the tool position and orientation (tool pose). Generally a serial robot with n DOFs has n joints and $n+1$ links, and the convention is that joint i connects link $i-1$ to link i. For a parallel robot, the total DOFs for the robot are equal to the DOFs associated with the moving links minus the number of constraints imposed by the joints.

The links are connected by joints. With no loss of generality, single-DOF joints are assumed, since joints having more DOFs can be expressed as a succession of single-DOF revolute or prismatic joints connected by links of zero length [Denavit and Hartenberg, 1955, Spong et al., 2006]. A prismatic joint is described by a cube with side d, resulting in a translational motion. The commonly used revolute joint has a cylindrical shape, where the motion is a rotation by an angle θ. As an example, the robot IRB1400 from ABB having revolute joints is illustrated in Figure 2.2a. A joint variable q_i is associated to the ith joint, given by

[1] Also called parallel kinematic machine (PKM).



(a) Joint angular positions *(b) Coordinate frames*

Figure 2.2: *The ABB robot IRB1400 with a) joint angular positions θ_i, $i = 1, \ldots, 6$ and b) coordinate frames for each joint of the robot.*

$$q_i = \begin{cases} \theta_i, & \text{revolute joint} \\ d_i, & \text{prismatic joint} \end{cases}$$

For an n-joint robot the joint variables q_i, $i = 1, \ldots, n$ form a set of generalised coordinates. The vector of these joint variables is often denoted q in the literature.

The robot configuration is given by q, and the joint space[2] is the set of all possible values for the joint variables. The workspace is defined by the total volume swept by the end effector when executing all possible motions of the robot[3]. The main disadvantage with parallel robots is that they generally have a smaller workspace compared to serial robots. On the other hand, a closed-kinematic chain structure can give a stiffer mechanical structure compared to an open-chain structure, thereby giving improved accuracy. It is also almost insensitive to scaling, since the same mechanical structure can be used for very small as well as large robots [Merlet, 2006, Merlet and Gosselin, 2008].

For a majority of the industrial robots, the links are actuated by electric motors. The motor positions are measured by sensors, usually encoders or resolvers. A transmission (gearbox) is often used, to amplify the motor torque and reduce the speed. Introduction of a gearbox gives reduced coupling effects, but at the expense of introducing gearbox friction, flexibilities and backlash. If the gearbox dynamics is neglected, the relation between the motor (actuator) position q_m and the joint position q can be expressed by

$$q = \Pi q_m \tag{2.1}$$

[2] Also called configuration space.
[3] Other names are operational space or Cartesian space.

with the matrix Π of gear ratios. The elements in Π are generally small (≈ 0.01) numbers.

2.3.2 Kinematics

Kinematics is a geometric description of the motion of rigid bodies, without considering the forces and torques causing the motion. The concepts of forward kinematics[4] and inverse kinematics of robots are illustrated in Figure 2.3. The aim of forward kinematics is to compute the robot tool pose as a function of the joint variables q. The inverse kinematics problem is the opposite — to compute the joint configuration from a given tool pose.

By attaching coordinate frames to each of the rigid bodies forming a robot and specifying the geometric relationships between the frames, it is possible to represent the relative position and orientation of one rigid body with respect to the other rigid bodies. As an example, the coordinate frames attached to each of the joints of the serial robot IRB1400 are illustrated in Figure 2.2b. The convention of coordinate frames and transformations for serial robots apply without any changes also to a parallel robot. For a point p rigidly attached to a coordinate frame i with local coordinate vector p_i, the coordinates of p are expressed with respect to the coordinate frame $i-1$ according to the rigid motion

$$p_{i-1} = R^i_{i-1} p_i + d^i_{i-1} \tag{2.2}$$

The vector d^i_{i-1} describes the translation of the origin of frame i relative to the origin of frame $i-1$. The rotation matrix R^i_{i-1} describes the orientation of the frame i with respect to the frame $i-1$. There are many ways of representing the orientation, for example by using the axis/angle representation, Euler angles, roll-

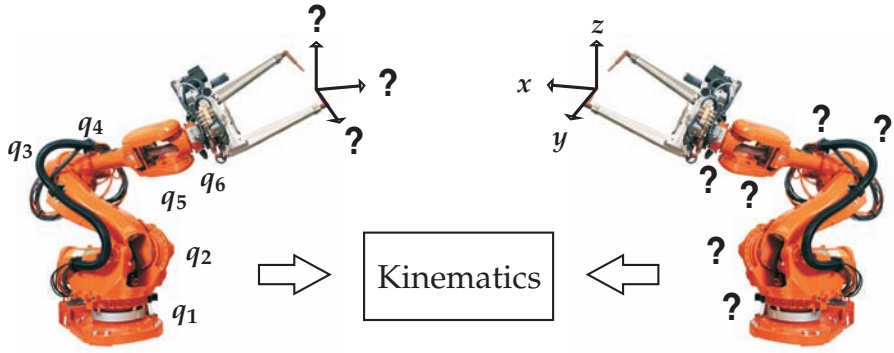


Figure 2.3: *In forward kinematics the joint variable vector q is known and the robot tool pose is sought. Inverse kinematics means to compute the joint configuration q from a given tool pose.*

[4] Sometimes also called direct kinematics.

pitch-yaw angles or quaternions, see for example Spong et al. [2006] or Sciavicco and Siciliano [2000].

The rigid motion (2.2) can be expressed using the homogeneous transformation H as in

$$P_{i-1} = H_{i-1}^{i} P_i = \begin{pmatrix} R_{i-1}^{i} & d_{i-1}^{i} \\ 0 & 1 \end{pmatrix} P_i \tag{2.3}$$

where P_i denotes the homogeneous representation of the vector p_i, given by

$$P_i = \begin{pmatrix} p_i \\ 1 \end{pmatrix}$$

The homogeneous transformation matrix H_{i-1}^{i} is not constant, but varies with the configuration of the robot, and is a function of the joint variable q_i.

Forward kinematics

The forward kinematics aims at deriving the tool pose as a function of the joint position q, see Figure 2.3. This can be described by the transformation

$$P_0 = H_0^n(q) P_n = \begin{pmatrix} R_0^n(q) & d_0^n(q) \\ 0 & 1 \end{pmatrix} P_n \tag{2.4}$$

from the tool frame n to the base frame 0. For a serial robot, the derivation of the forward kinematics is straightforward by direct geometric analysis of the kinematic chain. However, the complexity of the problem increases with the number of joints and the robot structure, and therefore systematic and general procedure is preferable. The Denavit-Hartenberg representation [Denavit and Hartenberg, 1955] is commonly used in robotics. In this convention, each homogeneous transformation matrix $H_{i-1}^{i}(q_i)$ from frame i to frame $i-1$ is represented as a product of four basic transformations, given from the geometric relationships between the frames. The forward kinematics always has a unique solution for serial robots.

For systems with a closed kinematic chain, additional changes have to be made to the Denavit-Hartenberg convention [Paul and Rosa, 1986], unless the kinematic chain can be rewritten to a corresponding serial kinematic chain. Generally, the kinematic description of a parallel robot it fundamentally different compared to the serial robot and requires different methods of analysis. No closed-form solution exists for the forward kinematics of a general parallel robot. Additionally, one set of joint variables generally corresponds to many tool poses. For parallel robots, this makes the forward kinematics problem usually much more complex than the inverse kinematics problem. From an initial guess of the solution, the forward kinematics is usually solved using the Newton-Raphson or the Newton-Gauss iterative scheme [Merlet and Gosselin, 2008]. There is however a risk that the procedure may not converge, or converges to a solution that is not the correct tool pose. Therefore, the correctness of the solution has to be analysed, as is discussed in for instance Merlet [2006] and Merlet and Gosselin [2008].

Inverse kinematics

The inverse kinematics problem is to find the values of the joint positions given the robot tool pose with respect to the base frame, see Figure 2.3. Solving the inverse kinematics problem is important when transforming the motion specifications of the tool to the corresponding joint positions to be able to execute the motion. For a serial robot the inverse kinematics is in general more difficult to solve than the forward kinematics. For the case of a serial robot, it means to solve the set of nonlinear equations given by

$$H_0^1(q_1)\cdots H_{n-1}^n(q_n) = H(q) \tag{2.5}$$

for a given homogeneous transformation $H(q)$, representing the given position and orientation of the tool. The joint variable vector q is to be found so that the relation is satisfied. Generally, the equations are nonlinear. It is not always possible to find a solution in closed form, and numerical methods for solving (2.5) are required. For a solution to exist, the tool pose must lie within the workspace of the robot. There may exist many solutions to the problem (or even infinitely many solutions, which is the case for a kinematically redundant robot). By the design of the robot kinematic structure, some of the problems can be avoided.

For parallel robots, the inverse kinematics problem is usually straightforward. One example is the Gantry-Tau parallel robot, where the inverse kinematics problem can be solved independently for each motor from the closure equations for the links [Johannesson et al., 2004, Dressler et al., 2007b]. For a given tool position, the robot has two solutions for the inverse kinematics, referred to the left- and right-handed configurations [Tyapin et al., 2002].

Velocity kinematics

Velocity kinematics relates the joint velocity vector $\dot{q}$ to the linear velocity v and angular velocity ω of the tool, as in

$$V = \begin{pmatrix} v \\ \omega \end{pmatrix} = J(q)\dot{q} \tag{2.6}$$

with the manipulator Jacobian $J(q)$, or Jacobian for short[5]. It is in general a nonlinear function of the joint variable vector q, computed by a geometric technique where the contribution from each joint velocity to the tool velocity is determined. Assume a minimal representation α of the orientation of the tool frame relative to the base frame, for example Euler angles. Then the analytical Jacobian $J_a(q)$ can be defined from the following expression

$$\dot{X} = \begin{pmatrix} \dot{d} \\ \dot{\alpha} \end{pmatrix} = J_a(q)\dot{q} \tag{2.7}$$

where d represents the origin of the tool frame with respect to the base frame. Generally, the manipulator Jacobian $J(q)$ differs from the analytical Jacobian $J_a(q)$.

[5] It is also called the geometric Jacobian, to emphasise that the derivation is based on the geometry of the robot structure.

For a serial robot, the ith column of the Jacobian is derived from the ith joint velocity only, independently of the velocities of the other joints. For a parallel robot, all other active joints are explicitly kept motionless. A closed-form expression for the inverse Jacobian is usually available for parallel robots, while it is difficult to derive an expression for the Jacobian [Merlet and Gosselin, 2008].

The Jacobian is one of the most important quantities in robot analysis and control, used in for example planning of smooth trajectories, determination of singular configurations and transformation of tool forces to joint torques. Identifying singular configurations is important. For a serial robot in a singular configuration, a nonzero joint velocity results in a constant tool position. This happens when columns of the Jacobian are linearly dependent, giving a decreasing rank of $J(q)$ for the actual configuration. Certain directions of motion may be unreachable, or bounded joint torques may correspond to unbounded tool forces and torques. Near singularities there may be no solution or infinitely many solutions to the inverse kinematics problem for serial robots. For parallel robots the singular configurations can be divided into different types [Gosselin and Angeles, 1990]. In the so-called serial singularity (decreasing rank of Jacobian) the joints can have a nonzero velocity, but the tool is at rest. For the parallel singularity (decreasing rank of inverse Jacobian), there are tool velocities for which the joint velocities are zero. This type of singularity is of great importance for analysis of parallel robots, since it corresponds to configurations where the robot loses controllability. Very large joint forces may also occur, leading to a breakdown of the robot [Merlet and Gosselin, 2008].

2.3.3 Dynamics

The dynamic model of a robot structure describes the relation between the robot motion and the forces causing the motion. First, a rigid-body dynamic model is presented. Second, modelling incorporating some of the mechanical flexibilities is discussed.

Rigid-body dynamic model

There are several methods for deriving a rigid-body dynamic model, all based on classical mechanics [Goldstein et al., 2002], with the Lagrange equations and the Newton-Euler formulation being the two most common approaches [Spong et al., 2006]. The Lagrange equations are based on the Lagrangian of the system, which is the difference between the kinetic and potential energy. The Newton-Euler formulation is a recursive formulation of the dynamic equations and it is often used for numerical calculation online. In this overview only the Lagrange formulation will be covered in some detail.

With the Lagrange approach, a set of generalised coordinates is chosen,

$$q = \begin{pmatrix} q_1 & \dots & q_n \end{pmatrix}^T \tag{2.8}$$

for an n-DOF robot. The Lagrangian $\mathcal{L}(q, \dot{q})$ is defined as the difference between

the total kinetic energy $\mathcal{K}(q, \dot{q})$ and the total potential energy $\mathcal{P}(q)$ of the system,

$$\mathcal{L}(q, \dot{q}) = \mathcal{K}(q, \dot{q}) - \mathcal{P}(q) \tag{2.9}$$

The kinetic energy can be rewritten to a quadratic function of the vector $\dot{q}$ of the form

$$\mathcal{K}(q, \dot{q}) = \frac{1}{2}\dot{q}^T M(q)\dot{q}$$

with the inertia matrix $M(q)$ consisting of the configuration dependent inertia matrix $M_a(q)$ of the robot arm plus the inertia matrix $M_m(q)$ of the rotating motors. Computing the Lagrangian (2.9) and denoting the partial derivative of the potential energy by $g_k(q)$ gives

$$g_k(q) = \frac{\partial \mathcal{P}(q)}{\partial q_k}, \qquad k = 1, \ldots, n$$

where the gravity vector $G(q)$ is

$$G(q) = \begin{pmatrix} g_1(q) & \ldots & g_n(q) \end{pmatrix}^T$$

The equations of motions for the system are given from Lagrange's equations

$$\frac{d}{dt}\frac{\partial \mathcal{L}(q, \dot{q})}{\partial \dot{q}_k} - \frac{\partial \mathcal{L}(q, \dot{q})}{\partial q_k} = \tau_k, \qquad k = 1, \ldots, n$$

where τ_k is the generalised force associated with the generalised coordinate q_k. The terms of the Lagrangian involving products of the type $\dot{q}_i$, called centrifugal terms, and the terms involving products of the type $\dot{q}_i\dot{q}_j$, called Coriolis terms, are grouped into the matrix $C(q, \dot{q})$. To summarise, the dynamic model of a rigid robot can be written as

$$M(q)\ddot{q} + C(q, \dot{q}) + G(q) = \tau \tag{2.10}$$

where τ is the vector of torques applied. Note that the quantities are expressed on the link side of the gearbox. The corresponding motor torques can be computed from $\tau_m = \Pi\tau$. The model (2.10) can be extended with, for example, a nonlinear friction torque $F(\dot{q})$. A classical friction model incorporates the Coulomb friction and viscous friction, while more advanced friction models are the LuGre model and the Stribeck friction model, described in for instance Olsson et al. [1998] and Armstrong-Hélouvry [1991].

For the serial robot, the generalised coordinates (2.8) can be chosen as the joint variables. If the robot contains a closed kinematic chain, an equivalent open kinematic chain of tree structure is derived by virtually cutting the loop at a joint. The generalised coordinates can then be chosen as the actuated joint variables. One possible way to derive the dynamic model of the parallel robot is to first derive the dynamic model of the equivalent open kinematic chain. The torques corresponding to the actuated joints can be computed, resulting in equations of motion in the form (2.10), see Sciavicco and Siciliano [2000].

Flexible joint dynamic model

The flexible joint model is presented as an example of how to extend the rigid dynamic model (2.10), where it is assumed that the rigid bodies are connected by elastic joints (gearboxes) modelled by torsional spring-damper pairs. This is a good description of the robot when the mechanical flexibilities are concentrated to the joints and when the gear ratio is high, thus reducing the inertial couplings between the motors and the links [Spong et al., 2006].

A flexible joint model for a single robot joint rotating in a horizontal plane is illustrated in Figure 2.4. The gearbox is characterised by spring coefficient k, damping coefficient d and gear ratio η. The mass of the motor and the arm have moments of inertia M_m and M_a, respectively, and the viscous friction f_m is approximated as acting only on the motor. The motor angular position and joint angular position are represented by q_m and q_a, respectively, and the motor is driven by the torque τ, which can be modelled by the input voltage times a torque constant, that is, $\tau = k_\tau u$. To summarise, the flexible joint model is described by

$$M_m\ddot{q}_m + f_m\dot{q}_m + \eta k(\eta q_m + q_a) + \eta d(\eta\dot{q}_m + \dot{q}_a) = k_\tau u = \tau \tag{2.11a}$$

$$M_a\ddot{q}_a - k(\eta q_m - q_a) - d(\eta\dot{q}_m - \dot{q}_a) = 0 \tag{2.11b}$$

The flexible joint dynamic model is used for illustrative purposes in Chapters 5, 6 and 9. From the dynamics (2.11) it is straightforward to compute the transfer functions

$$Q_m(s) = G_m(s)\tau(s) \tag{2.12a}$$

$$Q_a(s) = G_a(s)Q_m(s) \tag{2.12b}$$

The zeros of (2.12a) appear as poles of (2.12b), and corresponds to a notch frequency of $\omega \approx \sqrt{\frac{k}{M_a}}$ rad/s. The model (2.11) can be extended to for example include the inertial couplings between the links and the motors, see Tomei [1990]. A detailed survey of models for robots with elastic joints or elastic links can be found in for instance De Luca and Book [2008].

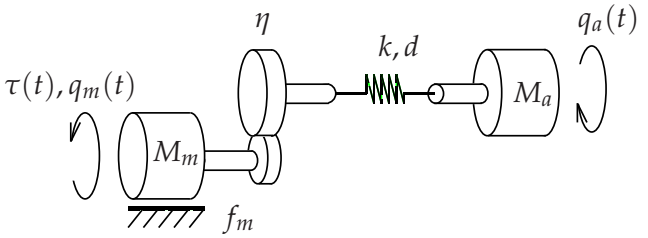


Figure 2.4: *A two-mass model of the dynamics of a single robot joint. It is characterised by spring coefficient k, damping coefficient d, viscous friction f_m, gear ratio η, moments of inertia M_m, M_a, motor angular position q_m and joint angular position q_a. The motor torque τ is generated by a torque constant k_τ times the input voltage.*

Extended flexible joint dynamic model

In Moberg [2010] it is discussed how to describe the dynamics of a modern industrial robot. The idea is to replace the one-dimensional spring-damper pairs in the actuated joints with spring-damper pairs of several dimensions, resulting in the extended flexible joint model originally presented in Moberg and Hanssen [2007]. If necessary, each elastic link is then divided into a suitable number of rigid bodies at proper locations, connected by multi-dimensional spring-damper pairs. By adding non-actuated pseudo-joints to the model, it allows for modelling of the bending out of the plane of rotation, for example the elasticity of bearings, tool and foundation, as well as bending and torsion of the links. As pointed out in Moberg [2010], the number of added non-actuated joints and their locations are not obvious. A proper selection of the model structure is therefore a crucial part of the modelling and identification.

The model equations are in Moberg and Hanssen [2007] given by

$$M(q_a)\ddot{q}_a + C(q_a, \dot{q}_a) + G(q_a) + \begin{pmatrix} f_g(\dot{q}_g) \\ 0 \end{pmatrix} = \begin{pmatrix} \tau_g \\ \tau_e \end{pmatrix} \tag{2.13a}$$

$$K_g(\Pi^{-1} q_m - q_g) + D_g(\Pi^{-1}\dot{q}_m - \dot{q}_g) = \tau_g \tag{2.13b}$$

$$-K_e q_e - D_e \dot{q}_e = \tau_e \tag{2.13c}$$

$$M_m \ddot{q}_m + f_m(\dot{q}_m) = \tau_m - \Pi^{-1}\tau_g \tag{2.13d}$$

where $q_a = \begin{pmatrix} q_g & q_e \end{pmatrix}^T$, with q_g being the vector of actuated joint angular positions, q_e denoting the vector of non-actuated joint angular positions and q_m representing the motor angular position vector. The actuator torques τ_m, τ_g and τ_e are defined analogously. Furthermore, M_m and $M_a(q_a)$ denote the inertia matrices for the motors and joints, where the inertial couplings between the motors and the rigid bodies are neglected due to the assumption of high gear ratios [Spong et al., 2006]. The matrices K_g, K_e, D_g and D_e are the stiffness and damping matrices in the actuated and non-actuated directions, respectively. The Coriolis and centripetal torques are described by $C(q_a, \dot{q}_a)$, the gravity torque is given by $G(q_a)$ and the nonlinear friction in motor bearings and gearboxes are modelled by the terms $f_m(\dot{q}_m)$ and $f_g(\dot{q}_g)$. For the choice of removing all non-actuated joints, the extended flexible joint model equals the flexible joint model.

Flexible nonlinear two-link dynamic robot model

As an example of dynamic robot models, the flexible nonlinear two-link robot model illustrated in Figure 2.5 is studied in some detail. The model corresponds to the second and third link of a large industrial serial robot with six motors, and is validated by experiments made on a commercial industrial robot [Moberg et al., 2008]. In the model the robot joints have nonlinear elasticity and friction, while the links are considered as being rigid. The model is part of the benchmark problem for robust control of a multivariable nonlinear flexible industrial robot, published in Moberg et al. [2008]. The model is also used in a simulation study of estimation-based ILC in Chapter 7.

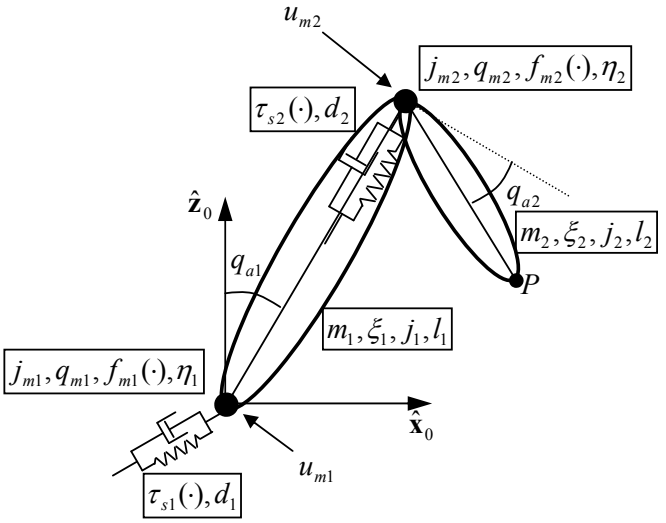


Figure 2.5: *Flexible nonlinear two-link robot model. The rigid links are described by mass m, link length l, center of mass ξ and inertia j with respect to the center of mass. The elastic joints (gear transmissions) are described by gear ratio η, nonlinear spring torque τ_s and linear damping d. The nonlinear friction torque f acts on the motors. The deflection in each joint is described by the motor angular position q_m and joint angular position q_a.*

In the model, each link has rigid-body characteristics described by mass m, link length l, center of mass ξ and inertia j with respect to center of mass. The links are actuated by electric motors, via elastic joints. The deflection in each joint is described by the motor angular position q_m and joint angular position q_a. The joint dynamics is described by nonlinear spring torque τ_s, linear damping d, nonlinear friction torque f and gear ratio η. The angular position vector q and vector u of model inputs are described by

$$q = \begin{pmatrix} q_{a1} \\ q_{a2} \\ q_{m1}/\eta_1 \\ q_{m2}/\eta_2 \end{pmatrix}, \qquad u = \begin{pmatrix} u_{a1} \\ u_{a2} \\ u_{m1}\eta_1 \\ u_{m2}\eta_2 \end{pmatrix}$$

The robot dynamics are given by

$$M(q)\ddot{q} + C(q, \dot{q}) + G(q) + D\dot{q} + K(q) + F(\dot{q}) = u \tag{2.14}$$

with the inertia matrix $M(q)$ as in

$$M(q) = \begin{pmatrix} M_{11}(q) & M_{12}(q) & 0 & 0 \\ M_{21}(q) & M_{22}(q) & 0 & 0 \\ 0 & 0 & j_{m1}\eta_1^2 & 0 \\ 0 & 0 & 0 & j_{m2}\eta_2^2 \end{pmatrix} = \begin{pmatrix} M_a(q_a) & 0 \\ 0 & M_m \end{pmatrix} \tag{2.15}$$

with the terms

$$
\begin{aligned}
M_{11}(q) &= j_1 + m_1\xi_1^2 + j_2 + m_2(l_1^2 + \xi_2^2 - 2l_1\xi_2 \sin q_{a2}) \\
M_{12}(q) &= j_2 + m_2(\xi_2^2 - l_1\xi_2 \sin q_{a2}) \\
M_{21}(q) &= M_{12}(q) \\
M_{22}(q) &= j_2 + m_2\xi_2^2
\end{aligned}
$$

and $M_a(q_a)$ and M_m denoting the inertia matrices for the links and motors, respectively. The vector $C(q, \dot{q})$ of Coriolis and centripetal terms is

$$
C(q, \dot{q}) = \begin{pmatrix} -m_2 l_1 \xi_2 (2\dot{q}_{a1}\dot{q}_{a2} + \dot{q}_{a2}^2) \cos q_{a2} \\ m_2 l_1 \xi_1 \dot{q}_{a1}^2 \cos q_{a2} \\ 0 \\ 0 \end{pmatrix} \tag{2.16}
$$

and the vector $G(q)$ of gravity torques is described as

$$
G(q) = \begin{pmatrix} -g\Big(m_1\xi_1 \sin q_{a1} + m_2\Big(l_1 \sin q_{a1} + \xi_2 \cos(q_{a1} + q_{a2})\Big)\Big) \\ -g m_2 \xi_2 \cos(q_{a1} + q_{a2}) \\ 0 \\ 0 \end{pmatrix} \tag{2.17}
$$

with g denoting the gravitational constant. The linear damping matrix D is

$$
D = \begin{pmatrix} d_1 & 0 & -d_1 & 0 \\ 0 & d_2 & 0 & -d_2 \\ -d_1 & 0 & d_1 & 0 \\ 0 & -d_2 & 0 & d_2 \end{pmatrix} \tag{2.18}
$$

The nonlinear spring torque is given by

$$
K(q) = \begin{pmatrix} \tau_{s1}(\Delta_{q1}) \\ \tau_{s2}(\Delta_{q2}) \\ \tau_{s1}(-\Delta_{q1}) \\ \tau_{s2}(-\Delta_{q2}) \end{pmatrix} \tag{2.19}
$$

with the deflection

$$
\Delta_{qi} = q_{ai} - q_{m1}/\eta_i, \quad i = 1, 2
$$

and where the nonlinear relations are

$$
\begin{aligned}
\tau_{si} &= \begin{cases} k_{i1}\Delta_{qi} + k_{i3}\Delta_{qi}^3, & |\Delta_{qi}| \le \psi_i \\ \operatorname{sign}(\Delta_{qi})\Big(m_{i0} + m_{i1}(|\Delta_{qi}| - \psi_i)\Big), & |\Delta_{qi}| \ge \psi_i \end{cases} \\
k_{i1} &= k_i^{\text{low}} \\
k_{i3} &= (k_i^{\text{high}} - k_i^{\text{low}})/(3\psi_i^2) \\
m_{i0} &= k_{i1}\psi_i + k_{i3}\psi_i^3 \\
m_{i1} &= k_i^{\text{high}}
\end{aligned}
$$

The nonlinear stiffness is specified by the lowest stiffness k_i^{low}, the highest stiffness k_i^{high} and the break-point deflection ψ. For the choice $k_i^{\text{low}} = k_i^{\text{high}} = k_i$, the stiffness is linear, $\tau_{si} = k_i \Delta_{qi}$. Finally, the vector of nonlinear friction torques is approximated as acting only on the motors, and is given by

$$F(\dot{q}) = \begin{pmatrix} 0 \\ 0 \\ f_1(\dot{q}) \\ f_2(\dot{q}) \end{pmatrix} \tag{2.21}$$

with the following friction model

$$f_i(\dot{q}) = \eta_i \Big(f_{vi} \dot{q}_{mi} + f_{ci} \big(\mu_{ki} + (1 - \mu_{ki}) \cosh^{-1}(\beta_i \dot{q}_{mi}) \big) \tanh(\alpha_i \dot{q}_{mi}) \Big), \quad i = 1, 2$$

The robot tool position P in the base frame, see Figure 2.5, is then described by the forward kinematics

$$P = \begin{pmatrix} x(q) \\ z(q) \end{pmatrix} = \begin{pmatrix} l_1 \sin q_{a1} + l_2 \cos(q_{a1} + q_{a2}) \\ l_1 \cos q_{a1} - l_2 \sin(q_{a1} + q_{a2}) \end{pmatrix} \tag{2.22}$$

The model can be extended to, for example, incorporate measurement noise and time delay of the measurements. The control signals (motor torques) can also be subject to disturbance and saturation. See Moberg et al. [2008] for the details regarding the benchmark problem and for suitable model parameters.

2.4 Control

There is a constant need for improved robot models and control methods to be able to satisfy the increasing demands on the robot performance [Brogårdh, 2007, 2009]. In this section a brief overview over the robot control system is given, together with some common control approaches. The reader is referred to for example Spong et al. [2006], Sciavicco and Siciliano [2000] or Siciliano and Khatib [2008] for a more detailed description of different control methods used in robotics. These references are also the main references used in this section. In Brogårdh [2007, 2009] and Moberg [2010] an overview of the current status of robot motion control is given from an industrial point of view.

2.4.1 Robot control system

The robot control problem can be divided into the following parts:

- **Path planning**: a path is determined in task space (or configuration space) in order to move the robot to a desired position without colliding with obstacles. It is a pure geometric description without any time laws specified. This can be done manually by specifying the motion by a series of commands in a robot programming language. The path can also be calculated in an offline-programming system, or be specified on a higher level by task programming.

- **Trajectory generation**: responsible for generating a tool trajectory, such that it is possible to follow the desired tool trajectory without actuator saturation or violating other limitations set by the robot designer (for example motor and gearbox torques and structural forces). The desired tool motion is specified in terms of position, velocity and acceleration as a function of time. To be able to use the generated trajectory for control, it has to be transformed into the configuration space, that is, corresponding joint angular positions. This step relies on the inverse kinematics, and is in general not possible to do analytically.
- **Motion control**: to control the motors such that the tool follows the desired trajectory. The problem could be point-to-point control, where only the start and end points on the path and the travelling time are defined, or trajectory tracking, when the desired trajectory is to be followed at every time step. The focus in this overview will be on trajectory tracking, which is a complex problem to solve due to the multivariable, nonlinear and coupled robot dynamics. Control of robots in interaction with the environment (for example force control) will not be treated here.

The parts of the robot control system is illustrated in Figure 2.6. The kinematic and dynamic robot models are used in path planning and trajectory generation to generate a trajectory that is feasible, that is, a trajectory possible to follow by the robot tool. The robot motion control is often model-based. The actual robot motion is measured by sensors, normally only the motor angular positions.

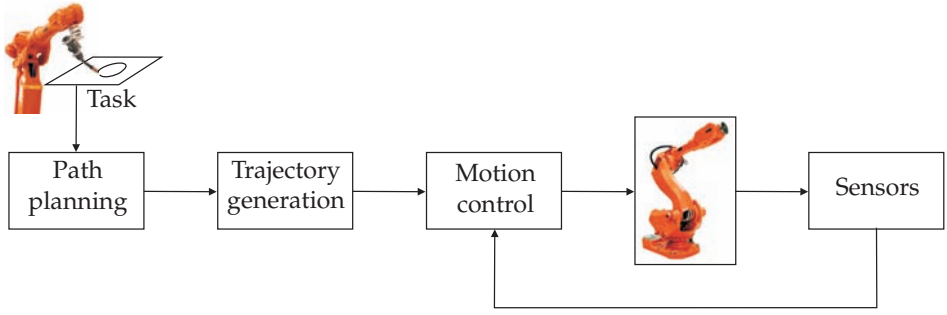


Figure 2.6: *Block diagram of the robot control system.*

2.4.2 Motion control

The mechanical design of the robot influences the type of controller. The robots studied in this thesis are driven by electric motors equipped with gearboxes, thus implying flexible joint dynamics.

Since the reference trajectory is known beforehand, it can be used in a feedforward controller to give the essential performance of the system. The feedback control loop adds stability and reduces sensitivity to disturbances and model uncertainties. It is often assumed that the motor torque control is ideal and can

be seen as a part of the motor, which also has been experimentally proved to be a reasonable assumption [Moberg, 2010]. Therefore, the control signal can be considered as the motor torque reference. The measurements available in commercial robot systems are normally only the motor angular positions [Brogårdh, 2009, Spong et al., 2006]. The motion control approaches can be divided into decentralised control and centralised control. The concepts are described in some more detail below by giving examples of common methods used in robot motion control, illustrated in Figure 2.7.

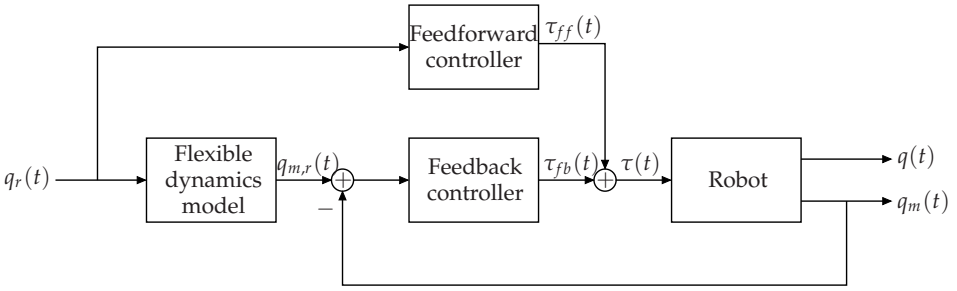


Figure 2.7: *Block diagram of the robot motion control.*

Independent joint control

The basic way of controlling a robot is by independent joint control, which is a decentralised control approach. Independent joint control means that each robot joint is controlled separately and the coupling effects between the links due to varying configurations are treated as disturbances. By relying on only motor angular position measurements, it requires a model of the gearbox dynamics of the single joint, to transform the desired trajectory in the joint space to the actuator space.

Since the main performance is expected to be given from a feedforward controller, a fairly simple structure of the independent joint controller can be chosen, for example proportional-derivative (PD) compensation

$$\tau_{\text{fb}} = K_P(q_{m,d} - q_m) + K_D(\dot{q}_{m,d} - \dot{q}_m) \tag{2.23}$$

where the desired motor position and motor velocity are given by $q_{m,d}$ and $\dot{q}_{m,d}$, respectively.

Feedforward control

Independent joint control is adequate in many point-to-point applications, but not for trajectory tracking. For a complex path, it is not satisfactory to have smooth trajectories in order not to excite the flexibilities in the robot structure, since this would lead to longer cycle times. Therefore, model-based controllers taking into account the robot dynamics should be considered, both for feedback and feedforward control. The model-based approach is typically more computationally intensive than the decentralised approach in independent joint control.

Consider the rigid-body dynamics (2.10) of a rigid robot. The main performance can be achieved by introducing a feedforward term based on the robot dynamics, as in

$$\tau_{\text{ff}} = \widehat{M}(q_d)\ddot{q}_d + \widehat{C}(q_d, \dot{q}_d)\dot{q}_d + \widehat{G}(q_d)$$

where $\widehat{M}$ is a model of the true inertia matrix M. When the true system is exactly known, the feedforward controller results in perfect tracking if no disturbances are present. However, since the model suffers from model uncertainties, the robustness to model uncertainties can be increased by combining the feedforward control action with independent joint control according to

$$\tau = \underbrace{\widehat{M}(q_d)\ddot{q}_d + \widehat{C}(q_d, \dot{q}_d)\dot{q}_d + \widehat{G}(q_d)}_{\tau_{\text{ff}}} + \underbrace{K_P(q_d - q) + K_D(\dot{q}_d - \dot{q})}_{\tau_{\text{fb}}}$$

Feedback linearisation control

The aim of feedback linearisation is to cancel nonlinear terms of the robot system by using the inverse dynamics according to

$$\tau = \widehat{M}(q)a_q + \widehat{C}(q, \dot{q})\dot{q} + \widehat{G}(q)$$

where a_q represents an input to be chosen. When the model is equal to the true system, it results in decoupled linear dynamics for each joint, namely decoupled double integrators

$$\ddot{q} = a_q$$

These decoupled systems can be controlled by a decentralised method, for example a PD-controller, by choosing

$$a_q = \ddot{q}_d + K_P(q_d - q) + K_D(\dot{q}_d - \dot{q})$$

The resulting control action is calculated from an inner loop (feedback linearisation) in combination with an outer loop (linear controller). However, one drawback is that the implementation of the feedback linearisation control requires online computation of the robot inverse dynamics.

2.5 Summary

The prerequisites for the era of industrial robots are mainly the early automation in the industry. The robots have developed from being simple devices performing only one task, to complex mechatronic systems capable of handling a broad spectrum of applications.

Models are important tools for improving robot performance. By kinematic models, the robot structure and for example singular configurations can be analysed. Dynamic models can be divided into rigid-body models and flexible models. The flexible joint dynamic model is used for illustrative purposes in the thesis in simulations presented in Chapters 5, 6 and 9. The simulation study in Chapter 7 is

based on the nonlinear two-link dynamic robot model with flexible joints.

An overview of the robot control system is given, and some methods for robot motion control are summarised. The actual robot motion is normally only measured by the motor angular positions. The control system constitutes a basis on which ILC can be applied to the robot in order to improve the system performance.

3

State estimation

THE IMPORTANCE of additional sensors is expected to increase in future development of industrial robots [Brogårdh, 2009]. Then there is an increased need of fusing different measurements together to obtain estimates of relevant signals. A short overview of some estimation techniques is given in this chapter, with focus on state estimation by using Kalman filtering and estimation by using complementary filtering. The intended usage of the approaches presented here is to derive estimates of the robot tool position to be used in ILC algorithms, as is discussed and illustrated in more detail in Chapters 6 to 8.

3.1 Estimation and sensor fusion in robotics

Due to both practical and economical reasons, the actual robot tool position is not measured in industrial applications. This will be discussed in more detail in Chapter 5. One possible way to handle this difficulty is to fuse measurements of different kinds together by signal processing and estimation algorithms, to form estimates of the tool position.

The Kalman filter is often used for fusing measurements of different kinds together to form an estimate. It has been used in a number of robotic publications. Some examples are the experiments presented in Alder and Rock [1994, 1996], where the states of a single-link flexible robot arm are estimated by a stationary Kalman filter in presence of unknown load dynamics and coloured sensor noise. In Li and Chen [2001], a laser-optical system is developed for measuring the tip deflection of a planar one-link flexible robot arm. The system states are thereafter estimated from the measurements of the tip deflection and motor angular position by a stationary Kalman filter.

Approaches for nonlinear estimation are also investigated in the robotics literature, with a number of publications discussing the usage of the extended Kalman filter (EKF). One example is Lertpiriyasuwat and Berg [2000], where also an overview of Kalman filtering and estimation in robotics are given. In the paper, the performance of the EKF based on models with different quality are compared when applied to a two-axis direct-driven robot arm with very flexible links. The estimation algorithm uses measurements of motor angular positions and tool position. Karlsson and Norrlöf [2005] estimates the robot tool position by using an EKF based on measurements of motor angular positions and tool acceleration. The approach is evaluated on a flexible-joint robot model with three DOFs. In Henriksson et al. [2009], different versions of the EKF as well as the deterministic observer by [De Luca et al., 2007] are evaluated on a commercial industrial robot, based on measurements of motor angular positions and tool acceleration. Experimental results show improved robot motion control.

Many contributions also address the nonlinear estimation problem of robots with structural flexibility, without using Kalman filters. One example is the early publication Nicosia et al. [1988], where approximate observers for robots having elastic joints are investigated. In a paper by Jankovic [1995], a reduced-order high-gain observer for elastic robot joints is derived when the motor angular positions and motor angular velocities are measured. The robustness of the observer is investigated by simulations of a two-DOF robot model with elastic joints. The deterministic observer by De Luca et al. [2007] explicitly uses the model structure of a robot with linear joint elasticity. Measurements of motor angular positions and signals from accelerometers mounted on the robot arm are used in the observer. The approach is validated by experiments on a commercial industrial robot.

3.2 Estimation algorithms

An accurate model is a prerequisite for obtaining a good estimate. The model does not necessarily have to fully describe the system, but has to capture the essential properties of the system and be well suited for being used in the estimation. Since most of the systems studied in practice are nonlinear, it implies that one has to solve a nonlinear estimation problem. Consider therefore a discrete-time nonlinear state-space model given by

$$x(t+1) = f\Big(x(t), u(t), t\Big) + v(t), \qquad v(t) \sim \mathcal{N}\Big(0, R_v(t)\Big) \tag{3.1a}$$

$$y(t) = h\Big(x(t), u(t), t\Big) + w(t), \qquad w(t) \sim \mathcal{N}\Big(0, R_w(t)\Big) \tag{3.1b}$$

with state vector $x(t)$, input $u(t)$ and measured output $y(t)$, see for example Rugh [1996] or Kailath et al. [2000]. The initial state vector $x(0)$ is assumed to be Gaussian with mean $\bar{x}_0$ and covariance $\bar{P}_0$. The process and measurement noise $v(t)$ and $w(t)$ enter additively and are assumed to be white with zero mean and covariance matrices $R_v(t)$ and $R_w(t)$, respectively. A sampling interval of $T_s = 1$ has been used in (3.1) for notational simplicity. The model (3.1a) describes the evolution of the state variables over time, while the measurement model (3.1b)

determines how the measurements are related to the state variables. The problem is now to estimate the state vector $x(t+m)$ by using measurements of the output $y(t)$ up to and including time t, denoted $x(t+m|t)$. The estimation problem can be divided into three categories; filtering ($m = 0$), prediction ($m > 0$) and smoothing ($m < 0$). In this overview, the focus will be on one-step ahead prediction ($m = 1$).

The estimation problem for the nonlinear model (3.1) can be solved by approximating the model (3.1) by a local linear model, and then derive an estimator for the linear model. The first approach, known as the linearised Kalman filter [Kailath et al., 2000], was to linearise the state-space equations around a known nominal trajectory. Later, it was found that relinearisation about the current state estimate might lead to smaller errors than for the former approach, and this procedure is called the (Schmidt) extended Kalman filter [Kailath et al., 2000], or EKF for short.

The Kalman filter was in 1960 presented in the seminal paper by Kalman [1960]. One of the classical references regarding Kalman filtering is the book Anderson and Moore [1979]. A publication thoroughly covering not only Kalman filtering but also many aspects of state-space estimation is Kailath et al. [2000], which together with Grewal and Andrews [2008] and Gustafsson [2000] are the main references for this chapter. An overview of examples of applications can be found in for instance Sorenson [1985].

3.2.1 Kalman filter

A common special case of the nonlinear model (3.1) is the linear time-variant state-space model subject to Gaussian noise, given by

$$x_{t+1} = A_t x_t + B_t u_t + v_t, \quad v_t \sim \mathcal{N}(0, R_{v,t}) \tag{3.2a}$$

$$y_t = C_t x_t + D_t u_t + w_t, \quad w_t \sim \mathcal{N}(0, R_{w,t}) \tag{3.2b}$$

where the subscript t is used from now on in the chapter to denote time dependence. The Kalman filter for this model is given in Algorithm 1, see also Kailath et al. [2000, pp.332–333], Grewal and Andrews [2008, p.138] or Gustafsson [2000, p.278] for a formulation of the equations. The update of the state estimate is recursively alternating between updating the state vector from measurements and one-step simulation. By adding new information from the measurement y_t in the measurement update step, the uncertainty is reduced and the covariance matrix $P_{t|t}$ of the estimation error is decreased. The uncertainty is then increased in the time update due to the one-step ahead prediction. Under the assumption of Gaussian variables x_0, v_t and w_t, the Kalman filter is the best possible estimator among all linear and nonlinear ones, in the sense of minimum variance of the estimation error.

The Kalman filter has been widely used in a broad field of applications. One explanation to its popularity is the usage of the state-space description. This enables easy incorporation of different types of sensor models within the algorithm, together with the possibility to quantitatively assign the quality of different sen-

Algorithm 1 Kalman filter

Consider the linear model (3.2). Initialise by setting $\hat{x}_{0|-1} = \bar{x}_0$ and $P_{0|-1} = \bar{P}_0$. The estimator is recursively computed by the following steps:

1. Measurement update

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(y_t - C_t\hat{x}_{t|t-1} - D_t u_t) \tag{3.3a}$$

$$P_{t|t} = P_{t|t-1} - K_t C_t P_{t|t-1} \tag{3.3b}$$

$$K_t = P_{t|t-1}C_t^T(C_t P_{t|t-1} C_t^T + R_{v,t})^{-1} \tag{3.3c}$$

2. Time update

$$\hat{x}_{t+1|t} = A_t\hat{x}_{t|t} + B_t u_t \tag{3.4a}$$

$$P_{t+1|t} = A_t P_{t|t} A_t^T + R_{w,t} \tag{3.4b}$$

sor measurements by the statistical properties introduced [Durrant-Whyte and Henderson, 2008]. The linear structure of the Kalman filter algorithm also makes it easy to implement and apply to the system.

In the case of applying a Kalman filter to a time-invariant state-space model, given by the relations

$$x_{t+1} = Ax_t + Bu_t + v_t, \quad v_t \sim \mathcal{N}(0, R_v) \tag{3.5a}$$

$$y_t = Cx_t + Du_t + w_t, \quad w_t \sim \mathcal{N}(0, R_w) \tag{3.5b}$$

the covariance matrix will approach the steady-state covariance $\bar{P}$ as $t \to \infty$. The Kalman gain is consequently approaching the steady-state Kalman gain $\bar{K}$, thus resulting in the stationary Kalman filter summarised in Algorithm 2, see also Gustafsson [2000, p.286]. The stationary Kalman filter aims at minimising the variance of the estimation error under the assumption that the initial error has faded away. From Algorithm 2 it can be seen that the Kalman gain $\bar{K}$ and covariance $\bar{P}$ can be computed in advance and stored in a computer. This gives a decreased computational load compared to the KF in Algorithm 1.

Algorithm 2 Stationary Kalman filter

Consider the linear time-invariant model (3.5). The stationary Kalman filter in predictor form is given by

$$\hat{x}_{t+1|t} = (A - A\bar{K}C)\hat{x}_{t|t-1} + A\bar{K}y_t + (B - A\bar{K}D)u_t \tag{3.6a}$$

$$\bar{P} = A\bar{P}A^T - A\bar{P}C^T(C\bar{P}C^T + R_v)^{-1}C\bar{P}A^T + R_w \tag{3.6b}$$

$$\bar{K} = \bar{P}C^T(C\bar{P}C^T + R_v)^{-1} \tag{3.6c}$$

3.2.2 Extended Kalman filter

To be able to apply the Kalman filter to nonlinear systems, the nonlinear system model (3.1) is linearised. A local approximation is obtained by a first-order Taylor

expansion around the current state estimate, giving

$$f(x_t, u_t, t) \approx f(\hat{x}_{t|t}, u_t, t) + F_t(x_t - \hat{x}_{t|t}) \tag{3.7a}$$
$$h(x_t, u_t, t) \approx h(\hat{x}_{t|t-1}, u_t, t) + H_t(x_t - \hat{x}_{t|t-1}) \tag{3.7b}$$

where

$$F_t = \left.\frac{\partial f(x, u_t, t)}{\partial x}\right|_{x=\hat{x}_{t|t}}, \qquad H_t = \left.\frac{\partial h(x, u_t, t)}{\partial x}\right|_{x=\hat{x}_{t|t-1}}$$

Using the linearisation in (3.1) gives

$$x_{t+1} = f(\hat{x}_{t|t}, u_t, t) - F_t\hat{x}_{t|t} + F_t x_t + v_t, \qquad v_t \sim \mathcal{N}(0, R_{v,t}) \tag{3.8a}$$
$$y_t = h(\hat{x}_{t|t-1}, u_t, t) - H_t\hat{x}_{t|t-1} + H_t x_t + w_t, \qquad w_t \sim \mathcal{N}(0, R_{w,t}) \tag{3.8b}$$

The Kalman filter can then be applied to this linear model (3.8), resulting in the EKF algorithm formulated in for example Kailath et al. [2000, p.340] or Grewal and Andrews [2008, p.400] and summarised in Algorithm 3.

Algorithm 3 Extended Kalman filter (EKF)

Consider the nonlinear model (3.1), linearised around the current state estimate to give the model (3.8). Initialise by setting $\hat{x}_{0|-1} = x_0$ and $P_{0|-1} = P_0$. An estimator is then recursively computed by the following steps:

1. Measurement update

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t\big(y_t - h(\hat{x}_{t|t-1}, u_t, t)\big) \tag{3.9a}$$
$$P_{t|t} = P_{t|t-1} - K_t H_t P_{t|t-1} \tag{3.9b}$$
$$K_t = P_{t|t-1}H_t^T(H_t P_{t|t-1} H_t^T + R_{v,t})^{-1} \tag{3.9c}$$

2. Time update

$$\hat{x}_{t+1|t} = f(\hat{x}_{t|t}, u_t, t) \tag{3.10a}$$
$$P_{t+1|t} = F_t P_{t|t} F_t^T + R_{w,t} \tag{3.10b}$$

From the relations (3.9) and (3.10) it can be seen that the calculations depend on the estimated states, due to the matrices F_t and H_t in the linearisation (3.7). The Kalman gain and covariance can therefore not be precomputed, which may result in a large computational load when implemented in a practical application [Kailath et al., 2000]. This is however not a critical issue in ILC applications, where the estimates can be derived offline.

Since the Kalman filter is applied to the linearised model (3.8), the resulting estimator is not optimal and there may be a nonlinear estimator producing better estimates. Also a considerable amount of tuning of the EKF is often required to produce an estimation filter with reasonably performance. Another problem is the risk of divergence. Still, the EKF often performs well in practice and this approach, with variations, is today the most widely used nonlinear state-space estimator [Kailath et al., 2000].

3.2.3 Complementary filtering

A pair of filters is called a complementary filter if the sum of their transfer functions is one over all frequencies [Higgins, 1975]. It is an estimation technique for sensor fusion used in for instance flight control industry and navigation system design and is popular because of its simplicity. Applications of the complementary filter are given in for example Higgins [1975] and the references therein. See also Pascoal et al. [2000] and Corke [2004] for more examples of applications where the method is used. In Higgins [1975] the relation between the complementary filter and the stationary Kalman filter is shown for a certain class of filtering problems. A similar kind of filter pairs is widely used in communication systems, see for example Vaidyanathan [1993]. However, there the sum of the transfer functions does not have zero phase, since many communication systems can handle time delays.

Complementary filtering is used to fuse noisy measurements of the same physical variable from two sensors with different frequency characteristics [Higgins, 1975]. No details about the statistical properties of the noise processes are considered in complementary filtering, and the filters are derived from a simple analysis in the frequency domain. For example, consider a case when measurement $s_1(t)$ from the first sensor has mostly high-frequency noise and measurement $s_2(t)$ from the second sensor suffers from noise mostly at low frequencies. First, the measurement $s_1(t)$ is filtered by using a low-pass filter $G(q)$, with q denoting the time-shift operator. A high-pass filter given by $1 - G(q)$, the complement to $G(q)$, is then used to filter the measurement $s_2(t)$. The filtered signals are then added to form an estimate of the physical variable $x(t)$ as in

$$\hat{x}(t) = G(q)s_1(t) + \Big(1 - G(q)\Big)s_2(t) \tag{3.11}$$

3.3 Summary

A short introduction to state estimation is given in this chapter, with focus on the linear Kalman filter and the extended Kalman filter (EKF). Finally, a model-free estimation approach, the complementary filter, is summarised. Examples of applications of these methods are given. In this thesis the estimation techniques are used to obtain estimates of the robot tool position to be used in ILC algorithms.

4

Iterative learning control

THE FUNDAMENTAL IDEAS and aspects regarding iterative learning control, or ILC for short, are described with focus on discrete-time linear systems. First, the concept of ILC is introduced and a formulation of the system description is given. Then, a general description of ILC updating formulas is presented, focusing on first-order linear algorithms. Stability and convergence properties of a linear system using ILC are then discussed, followed by examples of design methods. The chapter is concluded by discussing various applications of ILC.

4.1 Introduction to the concept of ILC

The concept of ILC is inspired by human learning [Arimoto et al., 1984b], and has its origin in industrial robot applications where the same task is performed repeatedly. It is then a sound idea to try to improve the performance, using information of how the task was performed in previous repetitions by adding a correction signal to the system. The key problem in ILC is what type of algorithm to use for generation of the correction input signal, which should result in a smaller error in some norm. When applying an ILC algorithm to a system, it should ideally result in a large decrease of the error with only limited (a minimum) knowledge of the system dynamics [Moore, 1993, Longman, 2000, Elci et al., 2002].

Another motivation for using ILC can be given by studying a system performing several types of motions [Moore, 1993]. An industrial robot is one example of such a system, since it can be programmed for a variety of tasks. It is a challenging problem to develop a robot motion control system which makes the robot follow all types of trajectories and satisfy the same requirements of high accuracy

everywhere in the workspace. When performing each of these different motions repeatedly, it is possible to improve performance by using ILC.

The inherent property of the signals involved when performing a task repeatedly is the finite time duration. For practical implementation of ILC algorithms, it involves computer-based controllers operating in discrete time together with digital storage of the information. Thus, it is natural to consider discrete-time ILC algorithms applied to systems operating in finite time [Moore, 1993, Longman, 2000]. Therefore, this thesis focuses on ILC algorithms for discrete-time systems.

4.2 Historical background

The origin of ILC can be traced back to the beginning of the 1970s. An application for a United States patent on "Learning control of actuators in control systems" was accepted in 1971, see Garden [1971]. The idea was to store a command signal in a computer memory. A learner iteratively updates the signal by an amount related to the error between the actual response and the desired response of the actuator. In the patent, applications of this learning control law are discussed for electric drive units and pneumatic actuators. This is clearly an implementation of ILC in a general sense, as is discussed in Chen and Moore [2000], although the updating equation is not explicitly formulated in the patent. The first academic contribution to what today is called ILC is the paper by Uchiyama [1978]. Since it was only published in Japanese, the idea was not widely spread until 1984, when the papers by Arimoto et al. [1984b], Casalino and Bartolini [1984], and Craig [1984] were independently published, all describing a method that iteratively could compensate for model errors and disturbances. Thereafter ILC started to become an active research area with a growing number of publications. The idea was especially developed by the group around Arimoto in a number of papers in the middle to late 1980s, as seen in for example the references in Moore [1998a].

The method was first referred to learning control, betterment process, iterative control, repetitive control, training or virtual reference, where virtual reference refers to that a correction signal makes the system follow a "virtual" reference to produce an output closer to the actual reference. The term iterative learning control is introduced in Arimoto et al. [1984a] and has become the standard notion during the latter part of the time span for the research area [Moore, 1993].

The development of ILC grew originally from practical issues in the field of industrial robotics, where repetitive motions appear in many applications. Much of the early work is about convergence of ILC algorithms and robustness analysis. Special attention is directed to linear systems or the special class of nonlinear systems represented by robot dynamic models, see for example Arimoto et al. [1984b], Togai and Yamano [1985], Bondi et al. [1988] and Kawamura et al. [1988]. From the late 1990s the focus has moved from analysis of ILC, to design and performance of algorithms. One typical example is the book by Bien and Xu [1998] that covers both analysis, design and applications. Another commonly referred paper is Longman [2000], discussing practical aspects of design and performance.

One of the open areas is ILC algorithms for nonlinear systems. Algorithms for different classes of nonlinear systems have been developed and analysed, but a unifying theory of ILC is still under development. Other examples of issues pointed out to be important future research areas are formalisation of the tradeoff regarding robustness versus performance, as well as connections to more general learning paradigms [Moore, 1998a, Bristow et al., 2006]. Open problems are also to derive general ILC methods when having not identical, but similar reference signals [Bristow et al., 2006], and how to design algorithms simple enough, robust and with good performance for industrial usage [Longman, 2000]. To conclude, the ILC field is still growing and there are many theoretical as well as practical issues to investigate.

The ILC research field is covered in a number of surveys, for instance Bien and Xu [1998], Chen and Wen [1999] and Bristow et al. [2006]. Moore [1998a] gives a detailed overview over the ILC area and a categorisation of much of the publications up to 1998 and in Ahn et al. [2007] there is a discussion and classification of the literature published between 1998 and 2004. A topological classification of the general results is given in Moore [1998a], which also contains a list of references related to robotics and other applications. A wide range of ILC algorithms has been developed, as can be seen in the survey papers and the references therein. Many of them, particularly those developed from a linear model of the system, are also experimentally evaluated, as is discussed in Section 4.9.

4.3 ILC related to other control approaches

First, ILC is put into the context of conventional feedback control. Then repetitive control is discussed, since it has much in common with ILC. Finally, ILC can be seen as a part of the larger class of intelligent/learning control approaches.

4.3.1 Conventional control

The ILC update signal is based on data from previous iterations, and the algorithm can be regarded as a feedback control law with respect to the iteration domain. In the time domain, on the other hand, the ILC algorithm is acting as a feedforward controller and the ILC input signal is updated only once per iteration. If the system is initially unstable, it should first be stabilised by a conventional feedback control technique, since the focus of the ILC algorithm is to improve performance [Bristow et al., 2006]. Hence, the system is assumed to be stable[1].

A hybrid arrangement with two independent controllers — a stabilising feedback controller in combination with an ILC algorithm — is very common in practical applications [Longman, 2000]. The conventional controller gives good tracking performance and reduces sensitivity to disturbances, while the ILC algorithm improves the performance by reducing the remaining repetitive errors. The ILC

[1] Se the postulated by Arimoto in Section 4.5, for the principles underlying the concept of ILC.

algorithm can therefore be seen as a complement to the conventional controller. Since the ILC algorithm is applied offline, it means that when computing the ILC input signal at the time instant t_1, it is possible to use information from previous iterations at a time instant $t_2 > t_1$. As a result, it is possible to use non-causal filtering techniques. This can be compared to feedback control, where the controller reacts to the error up to the current time step online, and therefore has to be causal.

ILC is categorised as an offline method with a feedforward control action in the time domain. This can be compared to model-based feedforward control, where the control action is computed by using a stable approximate inverse of the model[2]. This can also be done in a non-causal manner, in cases when the reference trajectory is known beforehand. If the dynamics of the true system are known, there is no need of iterative learning of the input in the disturbance-free case. However, the outcome of the model-based feedforward control depends on the quality of the model. ILC requires less information about the true system in order to give the desired system performance, as has been illustrated in a number of publications, see for example Moore [1993], Elci et al. [1994], Longman [2000] and Bien and Xu [1998].

Optimal control relies on an accurate model of the system. If the physical system changes, the optimal controller will no longer be optimal. On the other hand, by using an ILC algorithm one can easily adapt to system and environmental changes by adjusting the ILC input signal at the next iteration. This can be contrasted to the adaptive control technique, where most adaptive control approaches are applied online and adjust the controller parameters.

4.3.2 Repetitive control

The control approach closest to ILC is repetitive control (RC). In the RC situation, the command to be executed is a periodic function of time and disturbances having the same period are compensated. One example is a rotating disc with a disturbance occurring at every revolution. The system does not return to the same initial conditions before the next period starts, which is one of the assumptions in ILC, see Section 4.5. In RC the control changes at the end of one period therefore influence the error at the beginning of the next period and transients can propagate from one period to the next. This makes the stability analysis of ILC and RC very different. Although practical applications of ILC and RC seem very similar, the ILC approach is a special case of a two-dimensional system with a finite time variable and an infinite iteration variable, while RC is not. Among the differences between ILC and RC is also that an undefined amount of time can elapse between the ILC iterations and that the ILC learning does not need to be computed in real time, while the RC scheme acts as a feedback controller. For a deeper understanding of the similarities and differences between the two control approaches, a starting point could for example be the publications by Longman [2000] and Elci et al. [2002].

[2] See Section 2.4.2 for feedforward control in robot motion control.

4.3.3 Intelligent/learning control

Artificial neural networks, fuzzy logic, expert systems and machine learning are examples of frameworks that can be used for control. They all involve learning in some form, and the phrase learning control is sometimes mentioned in connection to these approaches [Moore, 1998a]. Learning control is a special case of learning in general in the context of intelligent systems, as is discussed in for example Albus [1991], where an overview picture of the field of learning is given. In the paper, the concept of learning is regarded as a mechanism for storing knowledge about the external world and to obtain skills and knowledge of how to act. In that sense ILC can be classified into the group of intelligent control strategies. However, ILC is based on a system-theoretic approach, in contrast to the other approaches which are based on artificial intelligence and computer science [Moore, 1993, 1998a].

The two fields of artificial neural networks and ILC are connected in for example Moore [1993]. This is also the topic for a number of contributions in Bien and Xu [1998], where integration of ILC and fuzzy/neural control methods is discussed.

4.4 System description

Consider a system T depicted in Figure 4.1. It has four scalar inputs; a reference signal $r(t)$, an externally generated control signal $u(t)$, and load and measurement disturbances $v(t)$ and $w(t)$, respectively. The measured variable is denoted $y(t)$, while the controlled variable is denoted $z(t)$. The system can have internal feedback, which means that the blocks denoted T_u, T_r, T_v and T_w contains the system to be controlled as well as the controller in operation. All signals are defined on a finite time interval $t = nT_s$, $n \in [0, N-1]$ with N number of samples and sampling interval T_s. In the remainder of this chapter, $T_s = 1$ if nothing else is stated. Finally, T_u, T_r, T_v and T_w are assumed to be stable.

The aim in ILC is to iteratively update the control signal $u(t)$, using information from the measured output $y(t)$, such that the controlled variable $z(t)$ tracks the reference $r(t)$ as well as possible. The system T, illustrated in Figure 4.1, will be used in the analysis of systems using ILC in this chapter. It is a special case of a more general system with a dynamic relationship between the measured variable $y(t)$ and controlled variable $z(t)$, which will be analysed in Chapter 6.

Time-domain description

In a situation where the system T in Figure 4.1 is a linear time- and iteration-invariant system operating in discrete time, it can be described using transfer operators

$$z_k(t) = T_r(q)r(t) + T_u(q)u_k(t) + T_v(q)v_k(t) \tag{4.1a}$$

$$y_k(t) = z_k(t) + T_w(q)w_k(t) \tag{4.1b}$$

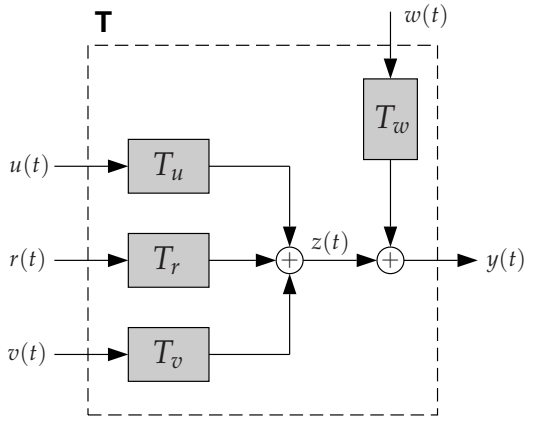


Figure 4.1: *Representation of the system used for analysis of ILC. Inputs are the reference signal $r(t)$, an externally generated ILC control signal $u(t)$, and load and measurement disturbances $v(t)$ and $w(t)$, respectively. Controlled output is $z(t)$ and measured output is $y(t)$. The system T contains the system to be controlled as well as the controller in operation.*

where q denotes the time-shift operator. The subscript k is the iteration[3] index and indicates how many times the iterative motion has been repeated. No repetitions have occurred when the motion is performed for the first time, which corresponds to $k = 0$.

The forthcoming analysis of systems using ILC will be based on the system description (4.1). However, it can be noted that it is not always possible to measure the controlled variable $z(t)$, corrupted by some noise $w(t)$, in practical applications. This issue is the common theme for Chapters 5 to 8. In these chapters the situation of a dynamic relationship between the measured variable $y(t)$ and controlled variable $z(t)$ is discussed and can be handled by introducing an estimate of $z(t)$ in the ILC algorithm.

Now define the vector $\boldsymbol{r}$ of the N-sample sequence of the reference signal $r(t)$ as

$$\boldsymbol{r} = \begin{pmatrix} r(0) & \dots & r(N-1) \end{pmatrix}^T \tag{4.2}$$

The other vectors $\boldsymbol{u}_k$, $\boldsymbol{y}_k$, $\boldsymbol{z}_k$, $\boldsymbol{v}_k$ and $\boldsymbol{w}_k$ are defined analogously. From the pulse-response coefficients $g_{T_r}(t)$, $t \in [0, N-1]$ of the transfer operator $T_r(q)$ in (4.1), the Toeplitz matrix $\boldsymbol{T}_r$ can be formed according to

$$\boldsymbol{T}_r = \begin{pmatrix} g_{T_r}(0) & 0 & \dots & 0 \\ g_{T_r}(1) & g_{T_r}(0) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_{T_r}(N-1) & g_{T_r}(N-2) & \dots & g_{T_r}(0) \end{pmatrix} \tag{4.3}$$

[3] Also called repetition, trial, pass, cycle or batch in the literature.

The system matrices $\boldsymbol{T}_u$, $\boldsymbol{T}_w$ and $\boldsymbol{T}_v$ are defined analogously. They are lower-triangular, since the corresponding transfer operators are causal. Finally, the system description (4.1) can be rewritten in matrix form as

$$\boldsymbol{z}_k = \boldsymbol{T}_r \boldsymbol{r} + \boldsymbol{T}_u \boldsymbol{u}_k + \boldsymbol{T}_v \boldsymbol{v}_k \tag{4.4a}$$

$$\boldsymbol{y}_k = \boldsymbol{z}_k + \boldsymbol{T}_w \boldsymbol{w}_k \tag{4.4b}$$

This matrix description is closely related to the descriptions of systems using ILC in Phan and Longman [1988], Moore [1998b] and Tousain et al. [2001], among others[4]. By writing the system (4.1) in matrix form, the time- and iteration-domain description of the system is rewritten as a MIMO iteration-domain system. The system description (4.4) is more general than the filter description (4.1), since it can handle also time-variant systems. Then, the matrices are not lower-triangular Toeplitz matrices, but instead general lower-triangular matrices that can change from iteration to iteration.

The vectors and matrices describing the system in matrix form are throughout the thesis written in bold face, to distinguish the matrix description (4.4) of the system from the system description (4.1) using transfer operators.

Frequency-domain description

The corresponding frequency-domain representation of $T_r(q)$ is defined from the discrete-time Fourier transform (DTFT) of the pulse-response coefficients $g_{T_r}(t)$,

$$T_r(e^{i\omega}) = \sum_{n=0}^{\infty} g_{T_r}(n) e^{-i\omega n} \tag{4.5}$$

or by replacing q by $e^{i\omega}$ in $T_r(q)$. The frequency-domain representation of the other transfer operators in (4.1) are defined similarly. The signals $z_k(t)$, $r(t)$, $u_k(t)$, $y_k(t)$, $v_k(t)$ and $w_k(t)$ are transformed into the frequency domain by the DTFT,

$$X(e^{i\omega}) = \sum_{n=0}^{\infty} x(n) e^{-i\omega n} \tag{4.6}$$

It is assumed that the sums (4.5) and (4.6) exist and are finite for all ω. Finally, the system (4.1) can be described in the frequency domain as

$$Z_k(e^{i\omega}) = T_r(e^{i\omega}) R(e^{i\omega}) + T_u(e^{i\omega}) U_k(e^{i\omega}) + T_v(e^{i\omega}) V_k(e^{i\omega}) \tag{4.7a}$$

$$Y_k(e^{i\omega}) = Z_k(e^{i\omega}) + T_w(e^{i\omega}) W_k(e^{i\omega}) \tag{4.7b}$$

The matrix formulation (4.4) is more general than the frequency-domain description (4.7), since it can contain both time-variant as well as iteration-variant systems. Also note that the frequency-domain representation uses the assumption of infinite time horizon. This is an approximation in the ILC setting, where a finite time horizon is considered. This will be discussed in Chapter 9, where the influence of boundary effects is analysed when filtering the signals over finite time intervals through possibly non-causal ILC filters.

[4] Also called for example lifted form or system description using supervectors in the literature.

4.5 Postulates by Arimoto

A number of principles that underlie the concept of ILC have been formulated in Arimoto [1990, 1998]. The principles are given as the following postulates, here with the formulation[5] in Arimoto [1998]:

P$_1$: Every iteration ends in a fixed time of duration $T > 0$.

P$_2$: A desired output $r(t)$ is given a priori over that time with duration $t \in [0, T]$.

P$_3$: Repetition of the initial setting is satisfied, that is, the initial state $x_k(0)$ of the objective system can be set the same at the beginning of each iteration:

$$x_k(0) = x_0 \quad \text{for} \quad k = 1, 2, \ldots$$

P$_4$: Invariance of the system dynamics is ensured throughout the repeated iterations.

P$_5$: Every output $y_k(t)$ can be measured and therefore the tracking error signal

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

can be utilized in construction of the next input $u_{k+1}(t)$.

P$_6$: The system dynamics is invertible, that is, for a given desired output $r(t)$ with a piecewise continuous derivative, there exists a unique input $u_d(t)$ that drives the system to produce the output $r(t)$.

Then the problem is to find a recursive control law

$$u_{k+1}(t) = F\Big(u_k(t), e_k(t)\Big)$$

and a function norm $\|\cdot\|$ such that $\|e_k\|$ vanishes as k tends to infinity.

As pointed out in for example Arimoto [1990, 1998], various disturbances related to the initial setting, measuring the output and implementing the control law are present in ILC applications. To meet these considerations, the postulates **P**$_3$, **P**$_4$ and **P**$_5$ are replaced by the following relaxed conditions, here using the formulation in Arimoto [1998]:

P$'_3$: The system is initialised at the beginning of each operation within an error level $\beta_1 > 0$, that is,

$$\|x_k(0) - x_0\| < \beta_1, \quad \text{for} \quad k = 1, 2, \ldots$$

P$'_4$: The norm of the input disturbance $\eta_k(t)$ induced during the repeated iterations is limited to some extent, that is, $\|\eta_k\| \leq \beta_2$.

P$'_5$: The output $y_k(t)$ can be measured with noise, that is, the output error is subject to

$$e_k(t) = r(t) - \Big(y_k(t) + \zeta_k(t)\Big)$$

where the $\mathcal{L}_2$-norm of the noise ζ_k must be small, say, $\|\zeta_k\| \leq \beta_3$.

[5] Reproduced with some minor changes to fit the notation used in the thesis

Although these underlying assumptions about the system controlled using ILC are important for analysis of the resulting ILC system, many contributions discuss how to overcome the difficulties when one or several of the postulates are not satisfied. For example, learning for the case when a system is to follow a reference trajectory $r_1(t)$, and thereafter be able to use the learning action to follow a similar but not identical reference trajectory $r_2(t)$ is discussed in a number of publications under the notion of direct learning control, see for example Xu [1997], Xu and Song [1998]. This is an example of how to overcome the restriction on the reference trajectory set by postulate $\mathbf{P}_2$. Another example is Saab et al. [1997], where tracking of a slowly varying reference is discussed. A number of publications are devoted to the situation with mismatch in initial conditions from iteration to iteration, see for example Lee and Bien [1991] and Xu et al. [2006], that is, violation of postulate $\mathbf{P}_3$. The work in this thesis especially investigates the case with an ILC algorithm applied to a system where postulate $\mathbf{P}_5$, or $\mathbf{P}_5'$, does not hold, that is, when the controlled variable is not measured.

4.6 ILC algorithms

Considering ILC algorithms, the categorisation can be based on a number of properties, such as linear or nonlinear, discrete time or continuous time and frequency domain or time domain. Another categorisation is first-order or high-order ILC algorithms, which are defined below according to the definitions given in Norrlöf [2000].

Definition 4.1 (First-order ILC algorithm). *An ILC updating formula that only uses measurements from the previous iteration,*

$$u_{k+1}(t) = F\Big(u_k(t), e_k(t)\Big)$$

is called a first-order ILC algorithm.

Definition 4.2 (High-order ILC algorithm). *When the ILC updating formula uses measurements from more than the previous iteration,*

$$u_{k+1}(t) = F\Big(u_k(t), u_{k-1}(t), u_{k-2}(t), \dots, e_k(t), e_{k-1}(t), e_{k-2}(t), \dots\Big)$$

it is called a high-order ILC algorithm. The terms second order, third order, etc., are used when the order of the ILC algorithm should be emphasised.

The focus in this thesis is on linear first-order ILC algorithms, which are described in more detail below. These algorithms are put into a more general context by briefly describing high-order linear algorithms and nonlinear algorithms. The reader is referred to for example Moore [1993], Bien and Xu [1998], Moore [1998a] and Ahn et al. [2007] and references therein for a more thorough description of other types of algorithms.

4.6.1 Linear ILC algorithms

First-order ILC algorithms

The ILC scheme proposed in the seminal paper by Arimoto et al. [1984b] is in continuous time, given in the form

$$u_{k+1}(t) = u_k(t) + \Gamma \dot{e}_k(t) \tag{4.8}$$

where $u_k(t)$ is the ILC input signal at iteration k and Γ is a constant gain. The update equation of the ILC algorithm (4.8) of so-called Arimoto type includes the derivative of the error $e_k(t)$, with the error given by the difference between the reference $r(t)$ and the measured output $y_k(t)$ as in

$$e_k(t) = r(t) - y_k(t) \tag{4.9}$$

This algorithm is in Arimoto [1985] generalised to

$$u_{k+1}(t) = u_k(t) + \Phi e_k(t) + \Psi \int_0^t e_k(\tau) d\tau + \Gamma \dot{e}_k(t) \tag{4.10}$$

with constant gains Φ, Ψ and Γ. The ILC update equation has a term proportional (P) to the error, an integral (I) term and a derivative (D) term of the error, thereby forming a PID-like system.

The discrete-time counterpart of the ILC algorithm (4.8) of Arimoto type is

$$u_{k+1}(t) = u_k(t) + \gamma e_k(t+1) \tag{4.11}$$

where the error at next time instant $t+1$ is used in the update equation. This type of algorithm is used in one of the first publications regarding discrete-time ILC algorithms, see Togai and Yamano [1985]. More recent examples are Moore [2001] and Moore et al. [2005], where in the latter publication also a time-varying learning gain $\gamma(t)$ is used. The algorithm (4.11) can be seen as a special case of a more general structure by introducing a filter $L(q)$, where $L(q) = \gamma q$ in (4.11). In the first contributions only the error $e_k(t)$ is filtered, as in (4.11). Later, other structures of ILC updating equations have been suggested, where also filtering of the ILC updating signal $u_k(t)$ is included, or filtering of the computed learning signal $u_k(t) + L(q)e_k(t)$ before applying it to the system. The latter alternative is used in this thesis, resulting in the algorithm

$$u_{k+1}(t) = Q(q)\Big(u_k(t) + L(q)e_k(t)\Big) \tag{4.12}$$

The structure (4.12) gives two degrees of freedom with the filters $Q(q)$ and $L(q)$, both possibly non-causal, and is widely used in the ILC community [Hara et al., 1988, Norrlöf, 2000, Elci et al., 2002, Barton and Alleyne, 2008].

The update equation for a first-order ILC algorithm in matrix form is similar

to (4.12) given by

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}(\boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{e}_k) \tag{4.13a}$$

$$\boldsymbol{e}_k = \boldsymbol{r} - \boldsymbol{y}_k \tag{4.13b}$$

In general, the matrix form (4.13) covers a larger class of algorithms than the filter form (4.12). For time- and iteration-invariant filters, the corresponding matrices are found from the pulse-response coefficients of the filters similarly as in (4.3). The filter form (4.12) can also be described in the frequency domain, derived similarly to the relations (4.5) to (4.6). In the thesis, the ILC algorithm in filter form (4.12) or matrix form (4.13) is considered. There are however possible generalisations, for example:

- **Current-iteration ILC**: The approach to incorporate feedback control in the ILC algorithm is discussed in for example Verwoerd [2005], Goldsmith [2002], Xu et al. [2004] and Norrlöf and Gunnarsson [2005], commonly referred to as current-iteration tracking-error ILC (CITE-ILC). One example of a CITE-ILC algorithm is given by

  $$u_{k+1}(t) = Q(q)\Big(u_k(t) + L(q)e_k(t)\Big) + C(q)e_{k+1}(t) \tag{4.14}$$

  where the errors from both the previous iteration and the current iteration are used to compute the ILC input signal. When separating the terms into a (non-causal) feedforward and a (causal) feedback component, it can be seen that CITE-ILC is equivalent to a combination of an ILC algorithm with a feedback controller [Bristow et al., 2006]. CITE-ILC appears naturally in the norm-optimal ILC design approach, described in some more detail in Section 4.8.
- **Time-varying and iteration-varying ILC filters**: Another generalisation of the algorithm (4.12) is to let the processing of $e_k(t)$ and $u_k(t)$ vary with time, that is, along the trajectory. A possible way to derive a time-varying ILC algorithm is to formulate the algorithm design as an optimisation problem, see Section 4.8. Examples of time-varying ILC filters are given in Hätönen et al. [2004] and Tharayil and Alleyne [2004], while the usage of iteration-varying filters is discussed in for example Norrlöf [2002], among others.

High-order ILC algorithms

Most of the existing ILC schemes are of first order [Chen et al., 1998, Norrlöf, 2000], that is, only the error from the previous iteration is used in the update equation. There are however also publications dealing with high-order ILC algorithms, as for instance Moore and Chen [2002] and Gunnarsson and Norrlöf [2006]. High-order ILC algorithms were first introduced in Bien and Huh [1989], where it is concluded that the rate of convergence can be improved when incorporating errors from previous iterations. Introducing high-order updating laws is also a way of reducing the effect of measurement disturbances, since the error is weighted over the iterations [Moore and Chen, 2002]. In Gunnarsson and Norrlöf [2006] disturbance properties of high-order ILC algorithms are analysed by using statistical models of the load and measurement disturbances.

For a general ILC algorithm of Nth order, the ILC input signal is computed from the ILC input signals and errors from the previous N iterations, see Definition 4.2. The update equation can according to Norrlöf [2000] be written

$$u_{k+1}(t) = \sum_{j=k-N+1}^{k} \Big(Q_{k-j+1}(q)\big(u_j(t) + L_{k-j+1}(q)e_j(t)\big)\Big) \tag{4.15}$$

with transfer operators $Q_j(q)$ and $L_j(q)$ for $j = 1, \ldots, N$. The concept of high-order ILC is intuitively described by an example in Norrlöf [2000], with constant filters $L_j(q) = l_j$ and where $Q_1(q) = 1$ and $Q_j(q) = 0$ for $j > 1$. The update equation (4.15) then results in

$$u_{k+1}(t) = u_k(t) + \sum_{j=k-N+1}^{k} l_{k-j+1} e_j(t) \tag{4.16}$$

where the second term can be interpreted as iteration-domain filtering of the error at time t by a filter with pulse-response coefficients l_j.

4.6.2 Nonlinear ILC algorithms

A general nonlinear ILC algorithm of Nth order is given by

$$u_{k+1}(t) = F\big(u_k(t), \ldots, u_{k-N+1}, e_k(t), \ldots, e_{k-N+1}(t)\big)$$

The class of nonlinear systems is very large, and as is pointed out in Moore [1993], it is not clear what type of nonlinear ILC algorithm structure $F(\cdot)$ that would be most suitable for learning control applications. Artificial neural networks are due to their nonlinear structure a good candidate, resulting in an iteration-varying updating scheme, see for example Moore [1993] and Bien and Xu [1998].

Still, ILC applied to nonlinear systems and nonlinear ILC algorithms is an open area, and the results available are applied on a case-by-case basis [Moore, 1998a, Xu and Bien, 1998].

4.7 Convergence properties

Convergence properties of ILC algorithms are considered in this section. The effects of disturbances will not be discussed in this overview, and therefore an ILC algorithm applied to a disturbance-free system will be studied. A thorough investigation of convergence properties of ILC algorithms is for instance given in Norrlöf and Gunnarsson [2002a], and an overview of these results can be seen in Bristow et al. [2006]. Most of the convergence properties presented here are well-known from linear systems theory, see for example Rugh [1996]. The main reference for this section is Norrlöf and Gunnarsson [2002a].

First, some matrix measures used in the sequel will briefly be recovered. The

spectral radius of an $N \times N$ matrix $\boldsymbol{F}$ is defined as

$$\rho(\boldsymbol{F}) = \max_{i=1,\ldots,N} |\lambda_i(\boldsymbol{F})| \tag{4.17}$$

where λ_i denotes the ith eigenvalue of the matrix $\boldsymbol{F}$. The maximum singular value of the matrix $\boldsymbol{F}$, defined by

$$\bar{\sigma}(\boldsymbol{F}) = \sqrt{\rho(\boldsymbol{F}^T\boldsymbol{F})} \tag{4.18}$$

gives a bound on the matrix gain according to

$$\|\boldsymbol{F}\boldsymbol{x}\|_2 \leq \bar{\sigma}(\boldsymbol{F})\|\boldsymbol{x}\|_2$$

with the 2-norm of a vector $\boldsymbol{x}$ given by

$$\|\boldsymbol{x}\|_2 = \sqrt{\boldsymbol{x}^T\boldsymbol{x}}$$

The analysis will be carried out using linear iterative systems [Norrlöf and Gunnarsson, 2002a], defined on a discrete and limited time interval. The linear iterative system considered in this section is given by

$$\boldsymbol{z}_{k+1} = \boldsymbol{F}\boldsymbol{z}_k + \boldsymbol{F}_r\boldsymbol{r} \tag{4.19}$$

where the vectors $\boldsymbol{r}$, $\boldsymbol{z}_k$ and $\boldsymbol{z}_{k+1}$ are defined from the N-sample sequence of the corresponding signals, similarly as in (4.2).

Applying the ILC algorithm (4.13),

$$\begin{aligned}\boldsymbol{u}_{k+1} &= \boldsymbol{Q}(\boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{e}_k)\\ \boldsymbol{e}_k &= \boldsymbol{r} - \boldsymbol{y}_k\end{aligned}$$

to the system (4.4) with disturbances omitted,

$$\boldsymbol{y}_k = \boldsymbol{T}_r\boldsymbol{r} + \boldsymbol{T}_u\boldsymbol{u}_k \tag{4.20}$$

gives

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_u)\boldsymbol{u}_k + \boldsymbol{Q}\boldsymbol{L}(I - \boldsymbol{T}_r)\boldsymbol{r} \tag{4.21}$$

The expression (4.21) is in the sequel denoted an ILC system in matrix form. The ILC system in filter form is similarly to (4.21) given as

$$u_{k+1}(t) = Q(q)\Big(1 - L(q)T_u(q)\Big)u_k(t) + Q(q)L(q)\Big(1 - T_r(q)\Big)r(t) \tag{4.22}$$

With $\boldsymbol{z}_k = \boldsymbol{u}_k$, $\boldsymbol{F} = \boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_u)$ and $\boldsymbol{F}_r = \boldsymbol{Q}\boldsymbol{L}(I - \boldsymbol{T}_r)$ it can be seen that the ILC system (4.21) is a special case of the linear iterative system (4.19). The system (4.22) is similarly a special case of a linear iterative system in filter form.

4.7.1 Stability

An essential property of a linear iterative system is bounded-input bounded-output (BIBO) stability, which is stated below.

Definition 4.3 (BIBO stability [Norrlöf and Gunnarsson, 2002a, Definition 4]). *The linear iterative system* (4.19) *is BIBO stable if a bounded input* $\|\boldsymbol{r}\| < \infty$ *generates a bounded output* $\|\boldsymbol{z}_k\| < \infty$ *for all* k.

Theorem 4.1 (BIBO stability [Norrlöf and Gunnarsson, 2002a, Theorem 1]). *Consider the linear iterative system* (4.19). *The system is BIBO stable if and only if*

$$\rho(\boldsymbol{F}) < 1 \tag{4.23}$$

Of importance for linear iterative systems is the transient response. From the following theorem a monotonously decreasing 2-norm of the signal is guaranteed when the homogeneous part of the linear iterative system is considered.

Theorem 4.2 (Monotone convergence [Norrlöf and Gunnarsson, 2002a, Theorem 2]). *Consider the linear iterative system*

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

If the maximum singular value satisfies

$$\bar{\sigma}(\boldsymbol{F}) < 1 \tag{4.24}$$

then

$$\|\boldsymbol{z}_{k+1}\|_2 < \|\boldsymbol{z}_k\|_2$$

From Theorem 4.1 it is concluded that the linear iterative system (4.19) is BIBO stable if $\bar{\sigma}(\boldsymbol{F}) < 1$. The condition (4.24) giving monotone convergence is however stronger than the stability condition (4.23).

Theorem 4.3 (BIBO stability, frequency domain [Norrlöf and Gunnarsson, 2002a, Theorem 6]). *The linear iterative system*

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

is BIBO stable if

$$\sup_{\omega\in[-\pi,\pi]} |F(e^{i\omega})| < 1 \tag{4.25}$$

The frequency-domain stability result is only asymptotically valid, that is, under the assumption of infinite time horizon. By the following theorem, a relation between the time-domain stability criterion (4.24) and the frequency-domain stability criterion (4.25) is provided.

Theorem 4.4 ([Norrlöf and Gunnarsson, 2002a, Theorem 8]). *Consider the linear iterative system*

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

Suppose a stable and causal filter $F(q)$ and

$$\sup_{\omega\in[-\pi,\pi]} |F(e^{i\omega})| < 1 \tag{4.26}$$

Then the largest singular value of $\boldsymbol{F}_N \in \mathbb{R}^{N\times N}$ in the matrix representation of the linear iterative system

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

satisfies

$$\bar{\sigma}(\boldsymbol{F}_N) < 1 \tag{4.27}$$

where $\boldsymbol{F}_N$ is the lower-triangular Toeplitz matrix with the first column being the N first pulse-response coefficients of $F(q)$.

4.7.2 Convergence of ILC algorithms

By using the stability results for linear iterative systems, the convergence properties of ILC algorithms can be studied. Stability of the ILC system is defined as follows.

Definition 4.4 (ϵ-convergence, stability [Norrlöf and Gunnarsson, 2002a, Definition 1]). *An ILC system* (4.21) *is ϵ-convergent in some norm $\|\cdot\|$ if*

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

where u_d is the input that drives the system to produce the desired output r.

An ILC system is called stable if it is ϵ-convergent with $\epsilon < \infty$.

If the linear iterative system (4.19) is BIBO stable according to Theorem 4.1, the output is bounded for all iterations k, that is, $\|\boldsymbol{z}_k\| < \infty$. BIBO stability for the ILC system (4.21) is given by the following theorem.

Theorem 4.5 (Stability [Norrlöf and Gunnarsson, 2002a, Corollary 3]). *The ILC system* (4.21) *is stable if and only if*

$$\rho\Big(\boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_u)\Big) < 1 \tag{4.28}$$

The stability property (4.28) of the ILC algorithm given by Theorem 4.5 is discussed by studying the following example.

Example 4.1: Stability of ILC system

Consider a stable SISO system

$$y_k(t) = G(q)u_k(t)$$

described by the transfer operator

$$G(q) = g_1 q^{-1} + g_2 q^{-2} + \ldots$$

from the pulse-response coefficients g_i. The input $u_k(t)$ is generated from an ILC algorithm given by (4.12). The ILC filter $L(q)$ is designed to compensate for the one-sample time delay of the system (g_0 is zero), giving $L(q) = l_1 q^1$. A filter $Q(q) = 1$ is chosen. From N pulse-response coefficients of the filters $G(q)$, $Q(q)$ and $L(q)$, the following Toeplitz matrices can be formed according to

$$\boldsymbol{G} = \begin{pmatrix} 0 & \dots & 0 & 0 \\ g_1 & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ g_{N-1} & \dots & g_1 & 0 \end{pmatrix}, \qquad \boldsymbol{Q} = I_{N\times N}, \qquad \boldsymbol{L} = \begin{pmatrix} 0 & l_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & l_1 \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

Calculating $\boldsymbol{Q}(I - \boldsymbol{LG})$ it results in a lower-triangular matrix with diagonal elements given by $1 - l_1 g_1$, except the last element, as in

$$\boldsymbol{Q}(I - \boldsymbol{LG}) = \begin{pmatrix} 1 - l_1 g_1 & 0 & \dots & 0 & 0 \\ -l_1 g_2 & 1 - l_1 g_1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -l_1 g_{N-1} & -l_1 g_{N-2} & \dots & 1 - l_1 g_1 & 0 \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix} \tag{4.29}$$

The output signal is monitored on the finite time interval $t \in [0, N-1]$ and the ILC input signal is calculated using the error on the same interval. Due to the one-sample time delay of the system, the value of the ILC input signal $u_k(T)$ for the last time instant will not affect the system output $y_k(T)$ and hence the last row of $\boldsymbol{LG}$ contains only zeros. Therefore, the last row of the matrix $\boldsymbol{Q}(I - \boldsymbol{LG})$ has the appearance as given in (4.29), which means that the ILC input signal value $u_k(T)$ is the same for all iterations. Now, considering only the part of the matrix $\boldsymbol{Q}(I - \boldsymbol{LG})$ which is dependent on the choice of the learning gain l_1 in the ILC filter $L(q)$, the stability condition (4.28) from Theorem 4.5, results in

$$|1 - l_1 g_1| < 1$$

The ILC algorithm applied to the system will convergence if and only if

$$0 < l_1 g_1 < 2$$

from which a suitable value of the learning gain l_1 can be chosen.

A number of conclusions can be drawn from this result. First, the condition for stability is independent of the system dynamics, since the first pulse-response coefficient is $g_1 = CB$ for a system described in state-space form with matrices A, B, C and D. Second, it is less useful in practical applications, as is pointed out in Longman [2000]. In Longman [2000] a third-order linear model of the input-output relationship of a single robot joint is studied, with the above ILC algorithm applied to the model. Simulations show a decreasing error for the first seven iterations, followed by an increasing error to a very large value of the root-mean square (RMS) error, and finally decreasing to zero error. Although the ILC algorithm converges, resulting in zero error, it would be impossible to achieve this in practice due to the high error level before reaching zero error.

From this example it can be concluded that it is relatively easy to guarantee stability of the system using ILC with the use of Theorem 4.5. In practical applications it is however not enough with stability, where the important aspect is to guarantee monotone convergence in some norm, as discussed in Elci et al. [2002] and Longman [2000].

Theorem 4.6 (Monotone convergence [Norrlöf and Gunnarsson, 2002a, Theorem 9]). *If the ILC system* (4.21) *satisfies*

$$\bar{\sigma}\Big(\boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_u)\Big) < 1 \tag{4.30}$$

then the system is stable and $\boldsymbol{u}_\infty - \boldsymbol{u}_k$ *will converge according to*

$$\|\boldsymbol{u}_\infty - \boldsymbol{u}_k\|_2 \leq \lambda^k \|\boldsymbol{u}_\infty - \boldsymbol{u}_0\|_2$$

with constant $0 \leq \lambda < 1$ *and with* $\boldsymbol{u}_\infty$ *defined as*

$$\boldsymbol{u}_\infty = \Big(I - \boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_u)\Big)^{-1}\boldsymbol{Q}\boldsymbol{L}(I - \boldsymbol{T}_r)\boldsymbol{r} \tag{4.31}$$

With the asymptotic ILC input signal $\boldsymbol{u}_\infty$ applied to the system (4.20), it gives the asymptotic error

$$\boldsymbol{e}_\infty = \boldsymbol{r} - \boldsymbol{y}_\infty = \Big(I - \boldsymbol{T}_r - \boldsymbol{T}_u\Big(I - \boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_u)\Big)^{-1}\boldsymbol{Q}\boldsymbol{L}(I - \boldsymbol{T}_r)\Big)\boldsymbol{r} \tag{4.32}$$

Generally, monotone convergence of $\boldsymbol{u}_k$ does not imply monotone convergence of the error $\boldsymbol{e}_k$, see for example Bristow et al. [2006]. One example when monotone convergence of the error is also achieved is for causal ILC filters and causal system, which implies lower-triangular Toeplitz matrices that commute. Then the condition on monotone convergence of the error reduces to the condition (4.30) on monotone convergence of the ILC input signal.

From Theorem 4.4, under the condition of the ILC system being causal with scalar input, the condition on the maximum singular value can be replaced by the criterion

$$\sup_{\omega\in[-\pi,\pi]} |Q(e^{i\omega})\Big(1 - L(e^{i\omega})T_u(e^{i\omega})\Big)| < 1 \tag{4.33}$$

under the assumption of infinite time horizon. This coincides with the well-known frequency-domain convergence criterion in the ILC literature, see for example Bristow et al. [2006] or Longman [2000]. The frequency-domain convergence criterion (4.33) offers some intuition to the problem of designing the ILC algorithm (4.12). The criterion (4.33) implies that there are two main routes for the design of the filters $Q(q)$ and $L(q)$. One approach is to choose $L(q)$ as the best possible inverse of the model $T_u(q)$. This requires that the model describes the true system sufficiently well. For a system having time delays, it also requires a non-causal filter $L(q)$. The second approach is to choose the magnitude of the filter $Q(q)$ small enough. If the frequency-domain criterion (4.33) is satisfied for a certain frequency, it means that this frequency component of the error is attenuated. On the contrary, violating the criterion (4.33) for a certain frequency im-

plies growth of this error component. An approach to satisfy the criterion (4.33) for all frequencies is to cut off the learning by the filter $Q(q)$ for the frequencies where the criterion is violated for $1 - L(q)T_u(q)$ [Elci et al., 2002, Longman, 2000]. Cutoff is also needed in practice to introduce robustness to model errors in $T_u(q)$. The price to pay is convergence to a non-zero error level [Elci et al., 2002]. The design is a trade-off between these two routes for the design of $Q(q)$ and $L(q)$.

It shall be noted that the frequency-domain condition (4.33) is an approximate condition for monotone convergence, see for instance Norrlöf and Gunnarsson [2002a] and Longman [2000]. First, the transformation to the frequency domain assumes infinite time horizon. The frequency-domain convergence condition (4.33) then assures that the amplitudes of all frequency components of the signal decay monotonically at every iteration. In applications of ILC, the time horizon is finite, and boundary effects will influence the output from the filtering operations, especially if the filter transients are long compared to the iteration duration. These issues are discussed in more detail in Chapter 9, where analysis of how the boundary effects will influence the convergence properties is performed in the time domain.

In contrast to the frequency-domain analysis, the benefit of the convergence analysis in the time domain is that also time- and iteration-variant systems and ILC algorithms can be considered. However, the disadvantage is that the time-domain approach is in general more computationally demanding. The issue of extending the usage of the time-domain approach to ILC systems with larger dimensions is considered in for instance in Barton and Alleyne [2008], where an upper approximation of the largest singular value is made by introducing submatrices. Thereby the size of systems possible for analysis in matrix form is increased.

4.7.3 Stability using two-dimensional systems theory

The information propagation is both along the trajectory of the current iteration and from iteration to iteration. By emphasising the dependence on the finite time t and infinite iteration k, the ILC algorithm (4.12) can be written as a linear equation in two dimensions,

$$u(k+1,t) = Q(q)\Big(u(k,t) + L(q)e(k,t)\Big) \tag{4.34}$$

Now define the iteration-shift operator, see for instance Norrlöf [2000] and Moore [1998a], by the relation

$$q_k u(k,t) = u(k+1,t) \tag{4.35}$$

For example, consider the ILC update equation (4.34), with the filters $Q(q) = 1$ and $L(q) = \gamma q$. The ILC algorithm can then be written

$$u(k,t) = \frac{\gamma q}{q_k - 1} e(k,t) = P(q, q_k)e(k,t) \tag{4.36}$$

in the variables t and k and transfer operator $P(q, q_k)$ with shift operators q and q_k.

A number of publications address convergence and stability of ILC systems by using two-dimensional analysis, for example Al-Towaim et al. [2004]. One of the first publications is Kurek and Zaremba [1993], where a state-space representation of a two-dimensional linear system is studied. Hladowski et al. [2010] apply two-dimensional systems theory to discrete-time linear repetitive processes to derive robust ILC algorithms. The algorithms are then applied to a gantry robot and evaluated experimentally.

4.8 Design methods

Almost three decades have passed since the start of the ILC research area in 1984. Since then the first algorithms have been modified and extended in a variety of directions, as is seen from the categorisation presented in Moore [1998a] and Ahn et al. [2007]. Following the discussion in Ratcliffe et al. [2005], one can from a wider perspective split the types of algorithms into two different categories; basic and model-based algorithms. The basic type of algorithms, as for example the ILC algorithm of Arimoto-type in (4.11), is very attractive from an industrial point of view [Ratcliffe et al., 2005, Longman, 2000]. This type of algorithms is easy to implement and adjust with a few parameters, and requires very little knowledge of the true system, such as for example static gain and time delay of the system. The model-based algorithms require an explicit model of the system, and the methods tend to be more computationally intensive. On the other hand, utilising more system knowledge could improve the resulting performance. In this section some examples of both basic and model-based design methods are given, with the discussion limited to linear ILC algorithms.

If it is possible to choose the filter $L(q)$ in (4.12) as the inverse of the system dynamics, it results in convergence to zero error in one single iteration. As discussed in for example Elci et al. [1994], there are practical difficulties when implementing such an algorithm. First, the discrete-time equivalent of a continuous system is very often non-minimum phase due to the sampling, and therefore it is not possible to invert the full system resulting in stable dynamics. However, by using causal and anti-causal filtering techniques, the inverse of the non-minimum phase system can be utilised in the design of the filter $L(q)$, as is discussed in Markusson et al. [2001]. The model suffers from uncertainties, especially at higher frequencies, which can violate the stability properties of the system. However, it can be useful to design the ILC algorithms based on inversion of parts of the system dynamics, see for instance Elci et al. [1994].

4.8.1 Basic algorithms

The inherent message for design and implementation of ILC algorithms in practical applications is clearly expressed in Norrlöf and Gunnarsson [2000] by the following words: *Try simple things first*. This is stressed already in the first publications in the field, see for example Arimoto [1990], where the simplicity of the algorithm is emphasised motivated by practical implementation, as well as the

desire to design the ILC algorithm without knowing the entire system dynamics. These issues are also discussed in a number of publications by Longman and co-authors, for example Elci et al. [1994], Longman [1998, 2000] and Elci et al. [2002].

Design methods relying on very little knowledge of the system are sometimes called model-free ILC or heuristic ILC algorithms [Norrlöf and Gunnarsson, 2002b]. One example is when the knowledge of the system consists of only the time delay and the static gain of the system to be controlled. The learning gain γ is then chosen such that γ times the static gain is less than 1. The time delay of the system is compensated by a time shift forwards in time, see Example 4.1. Finally, the bandwidth of the filter $Q(q)$ determines how large part of the error dynamics that should be learned. The heuristic design method is summarised in Algorithm 4 below. See also Norrlöf [2000] for the formulation of the algorithm design. Some examples of algorithms designed according to Algorithm 4 are given in Elci et al. [2002], Abdellatif et al. [2006] and Freeman et al. [2010].

Algorithm 4 Heuristic design

1. Choose the filter $Q(q)$ as a low-pass filter with cutoff frequency such that the bandwidth of the learning algorithm is sufficient.
2. Let $L(q) = \gamma q^{\delta}$. Choose the learning gain γ and time shift δ such that the frequency-domain stability criterion (4.33) is satisfied. Normally it is sufficient to choose δ as the time delay of the system and $0 < \gamma \leq 1$ to get a stable ILC system.

There are also model-free design methods based on self tuning. In these methods the ILC design parameters, for example learning gain, time shift and cutoff frequency, are tuned or adapted along the iteration axis until an algorithm with reasonable learning speed and final error level is obtained. See for example Longman and Wirkander [1998] for a discussion of different self-tuning techniques.

4.8.2 Model-based algorithms

Optimisation-based design

Using the matrix description of the system and ILC algorithm, the algorithm design can be considered in the context of numerical optimisation, see for example Togai and Yamano [1985], Lee and Lee [1998a], Gunnarsson and Norrlöf [2001], Owens and Hätönen [2005] and Barton et al. [2008]. A suitable ILC input signal $\boldsymbol{u}_k$ is derived from minimising a cost function, for example based on the current cycle error, and the change in the ILC input signal, according to

$$J_{k+1} = \|\boldsymbol{e}_{k+1}\|^2 + \|\boldsymbol{u}_{k+1} - \boldsymbol{u}_k\|^2 \tag{4.37}$$

One possible update equation, see Amann et al. [1996a], is given by

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

where $\boldsymbol{L} = \alpha_{k+1}\boldsymbol{T}_u^*$, with $\boldsymbol{T}_u^*$ being the adjoint of $\boldsymbol{T}_u$ and α_{k+1} denoting the step length to be chosen at each iteration. The aim is to achieve optimal correction while still avoiding too high actuator demands by penalising the difference of the ILC input signal between successive iterations. One example is the early reference Togai and Yamano [1985], where the learning gain is derived by minimising a quadratic criterion of the error by using gradient-based methods. The problem of deriving an ILC input signal by minimising the criterion (4.37) is further studied in Amann et al. [1996a,b] under the notion of norm-optimal ILC. In Owens and Hätönen [2005] norm-optimal ILC is used for effectively controlling a gantry robot system.

The cost function (4.37) to be minimised can be generalised to

$$J = \|\boldsymbol{e}_{k+1}\|_{\boldsymbol{Q}_J}^2 + \|\boldsymbol{u}_{k+1}\|_{\boldsymbol{S}_J}^2 + \|\boldsymbol{u}_{k+1} - \boldsymbol{u}_k\|_{\boldsymbol{R}_J}^2 \tag{4.38}$$

with sums of weighted norms with symmetric positive definite matrices $\boldsymbol{Q}_J$, $\boldsymbol{S}_J$ and $\boldsymbol{R}_J$, often chosen as $(\boldsymbol{Q}_J, \boldsymbol{S}_J, \boldsymbol{R}_J) = (qI, sI, rI)$, as seen in Ratcliffe et al. [2006]. Tuning guidelines of the weights are given in Barton et al. [2008], where it is seen that the convergence speed strongly depends on $\boldsymbol{R}_J$.

Design of an ILC algorithm based on optimisation can be summarised by Algorithm 5, see also [Norrlöf and Gunnarsson, 2002b].

Algorithm 5 Model-based time-domain design using optimisation

1. Build a model of the relation between the ILC input and the resulting correction on the output, that is, find a model $\hat{\boldsymbol{T}}_u$ of $\boldsymbol{T}_u$.
2. Choose weighting matrices.
3. Minimise a quadratic criterion in the error and the control signal. The resulting algorithm can be interpreted as matrices $\boldsymbol{Q}$ and $\boldsymbol{L}$.

Frequency-domain design

Another branch of design methods is based on frequency-domain analysis, which implies linear time- and iteration-invariant ILC algorithms. One example is given by Algorithm 6, see also Norrlöf and Gunnarsson [2002b]. Examples of frequency-based ILC design methods are de Roover [1996], where the ILC design is generalised to the synthesis of a sub-optimal H_∞-controller, and Norrlöf and Gunnarsson [2002b], where an algebraic approach is adopted. In Wang and Ye [2004] a multi-channel approach is proposed to extend the bandwidth of the ILC system. The error is divided into different frequency bands, where an ILC algorithm is tuned based on the frequency-domain criterion (4.33) for the frequency range of interest. See also, for example, Longman [2000] for a discussion of design and tuning of the ILC filters based on the frequency-domain convergence criterion (4.33).

Algorithm 6 Model-based frequency-domain design

1. Build a model of the relation between the ILC input and the resulting correction on the output, that is, find a model $\hat{T}_u(q)$ of $T_u(q)$.
2. Choose a filter $H_B(q)$ such that it represents the desired convergence rate for each frequency. Normally this means a high-pass filter.
3. Compute $L(q) = \hat{T}_u^{-1}(q)\big(1 - H_B(q)\big)$.
4. Choose the filter $Q(q)$ as a low-pass filter with cutoff frequency such that the bandwidth of the resulting ILC algorithm is high enough and the desired robustness is achieved.

4.9 Applications of ILC

The patent Garden [1971] is an example of early industrial usage of ILC, as is discussed in Chen and Moore [2000]. Experimental results presented later show improved performance of highly coupled, time-varying or nonlinear systems under repetitive motions. The control specifications met by using ILC may not easily be satisfied by other control methods, as other methods require more prior knowledge of the process in the controller design. ILC requires much less information of the system variations to yield the desired dynamic behaviour [Moore, 1993, Longman, 2000, Bien and Xu, 1998]. For example, experimental results in Elci et al. [1994] show that it is possible to come close to the reproducibility level of the robot system without relying on a complex model of the entire robot dynamics. In Norrlöf and Gunnarsson [2002b] an experimental comparison of ILC design approaches is presented with the ILC algorithm applied to the first three joints of a commercial industrial robot of moderate size. One observation from this comparison is that the heuristic approach presented in Algorithm 4 performs surprisingly well compared to the model-based approaches in Algorithms 5 and 6. In Abdellatif et al. [2006] an ILC algorithm with phase-lead compensation and zero-phase filter $Q(q)$, Algorithm 4, is compared to a more general linear form and to a model-based approach. The same conclusion was found also in Abdellatif et al. [2006], that Algorithm 4 performs well. This is one reason for choosing this design method in the simulations and experiments presented in the thesis. However, usage of more system knowledge could improve performance.

However, ILC is still not yet widely used in engineering applications. In order to increase the impact of ILC in practical usage, the following items could be one way of progress [Longman, 2000]:

- *Linear ILC formulation.* A majority of the nonlinear control problems are addressed by linear control laws in practice. Choose the simplest approach that works.
- *Discrete-time ILC algorithms.* To measure, store and process the information, it requires usage of a digital computer online, and it is natural to consider the ILC problem in discrete time.

- *Existing feedback controllers.* By simultaneously designing feedback controllers and ILC algorithms, it restricts the number of practical applications. A method easily implemented and applied to an existing feedback control system is desirable.
- *Command to the feedback controller adjusted, not the manipulated variable.* The only possible solution in many practical control applications is to adjust the input to the controller, since it is not desirable/possible to go into the existing controller and modify the signals sent to the actuators.
- *Simple ILC algorithms with a small number of parameters to adjust.* PID controllers have only a few parameters to adjust and the tuning is easy to understand. It is therefore the largest class of feedback control laws used in practice. The aim is to have an ILC algorithm in a similar fashion.
- *Typical knowledge about system behaviour.* Do not try to design a universal ILC algorithm that works for every system. Make instead use of information about the system dynamics that is easy to obtain, for example, by frequency-response tests.
- *Monotone convergence.* Universal ILC algorithms that result in convergence to zero error are often of little use in practice, due to increasing error levels before convergence to zero error. The difficult part is to derive an ILC algorithm that guarantees monotone convergence.
- *Long-term stability.* Instability may occur after only many iterations, which can be avoided by giving the user guidelines of how to adjust the ILC design parameters.
- *Knowledge foundation of practicing control engineers.* Make the concepts easily understood to make the control system engineer knowing how to apply them, for example by interpreting the information intuitively in a Bode plot.

4.9.1 Examples of applications

Finally, a number of applications of ILC will be mentioned, in order to show the broad field of potential usage. Due to the quite recent growing interest in design, performance and practical aspects of ILC, studies where an ILC algorithm is applied to a commercial industrial platform are still less common in the ILC literature. This also motivates the choice of applying ILC algorithms to a serial and parallel industrial robot in Chapters 5 and 8. ILC applied to resonant systems is of special interest here, since the systems considered in this thesis suffer from mechanical flexibilities in the frequency region of interest[6].

A summary of publications regarding ILC applied to especially robotics is given in Moore [1998a]. See also Ahn et al. [2007] for a detailed categorisation of recent publications, with applications of ILC in robotics, rotary systems and biomedical

[6] See also Chapter 5 for illustrative experimental and simulation results of ILC algorithms applied to systems containing mechanical flexibilities.

applications, as well as for controlling actuators, semiconductors or power electronics. Regarding the robotics area, ILC is applied to for example direct-drive robots, serial and parallel robots and XY-tables. The focus in this thesis is on ILC algorithms applied to industrial robotics, and therefore the publications mentioned in this section mainly relate to the robotics field. The applications are divided into three categories, covering the range from simulation and laboratory-scale experiments to industrial products.

Simulation studies and laboratory-scale experiments

- Convergence and stability properties of various ILC algorithms, exemplified by simulation studies which naturally involves a large number of iterations. The ILC research community has earlier been very focused on stability, which explains the number of work in this area. Some works are for example Elci et al. [2002] and Wang et al. [2003].
- New approaches to ILC, verified by simulation studies or experiments performed on laboratory-scale testbeds that are representative for the industrial problems. The robot model has at most two or three DOFs. Examples of such studies are De Luca and Ulivi [1992], Gunnarsson et al. [2007] and Tayebi [2004].
- ILC algorithms applied to robot arms with several DOFs. In Elci et al. [2002] a simple first-order ILC algorithm is applied to all axes of a seven-DOF robot arm. The ILC design variables are tuned experimentally, or by using a simple model of the system identified from experiments. The motion is chosen to illustrate theoretical results and is not motivated by any practical application, which also is the case in Longman [2000].
- Different application areas other than robotics, where the system is hard to model accurately and is subject to disturbances, as for example batch chemical processes. The ILC algorithms have to be combined with feedback control to be successful. ILC applied to laboratory-scale batch chemical processes is described in for instance Lee and Lee [1998b].

Industry-motivated experiments

- Different aspects of implementation and practical usage of ILC algorithms. One example is Dijkstra and Bosgra [2002], where an ILC algorithm designed using optimisation and is applied to a wafer stage, which is a system used in production of integrated circuits. Only the observable part of the system in iteration domain is considered in the design, which gives a controller of lower order. The resulting ILC update rule, given in the matrix description, is implemented by interpreting the update as filtering operations. In Ratcliffe et al. [2006] implementation aspects are discussed for norm-optimal ILC, where a faster version of the norm-optimal algorithm is presented and evaluated in experiments.
- Experiments and trajectories clearly motivated by applications. Norm-optimal ILC is experimentally evaluated in Ratcliffe et al. [2006] and Barton et al. [2008], where different tuning aspects are discussed, together with

robustness to initial state errors. In Ratcliffe et al. [2006] the algorithm is applied to a gantry robot for automation applications, where each motor is controlled individually. In Barton et al. [2008] a practically motivated trajectory is to be followed by a multi-axis robotic testbed, where the raster scanning trajectory consists of long periods of low-frequency content followed by short periods of high-frequency movements.

- Different aspects of ILC applied to commercial robot systems. See for example Norrlöf [2000], where the issue is discussed both from the design and experimental point of view.
- Discussion of the entire control system, with ILC as a final tool to improve performance. Modelling, feedback and feedforward control for a parallel robot, and finally ILC as a tool to improve accuracy especially for higher frequencies are discussed and experimentally verified in Abdellatif and Heimann [2010].

Industrial products

Besides the first patent by Garden [1971], also the patent by Gunnarsson et al. [2006] is worth mentioning here, where path correction for an industrial robot is described with the use of laser measurements of the tool position in the ILC algorithm. The method is used in for example workcells for laser cutting on car frames. Another example is the patent by Chen et al. [2002], dealing with ILC applied to hard disk drives.

4.10 Summary

Stability and convergence properties of the system using ILC are given in the chapter, with focus on first-order linear algorithms applied to linear systems. The time-domain analysis is more general, including both time-variant as well as iteration-variant systems and algorithms, while the frequency-domain analysis is more intuitive but approximate. The most important aspect in practice is to guarantee monotone convergence. Tuning of the ILC algorithm is a trade-off between performance, where ILC filters are ideally chosen as the inverse of the system dynamics, and robustness, with only learning for frequencies where the model errors are small. The price to pay for cutting off the learning for some frequencies is convergence to a non-zero error level.

Part II

Results

5

Motivation to estimation-based ILC

TRADITIONALLY ILC HAS BEEN APPLIED to systems where the controlled variable also is the measured variable. In industrial robot applications, as for example plasma cutting shown in Figure 5.1, this is typically not the case. Usually only the motor angular positions are measured in a commercial robot system, while the control objective is to follow a desired tool path. In this chapter experimental results are discussed when applying an ILC algorithm to two different industrial robots. First, experiments are performed on a commercial serial robot. Second, a parallel robot prototype will be used as the experimental testbed. In

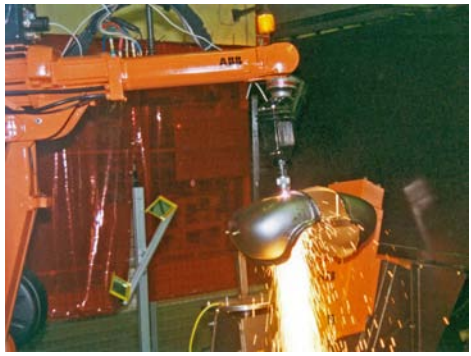


Figure 5.1: *The industrial robot IRB1400 performing plasma cutting. The control objective is to follow a desired tool path, whereas the measured variables are the motor angular positions. The tool position and orientation are not measured due to practical and economical reasons. Photo from ABB Robotics [2007].*

both cases, the ILC algorithm uses the measured motor angular positions directly and the resulting tool performance is evaluated. The key properties of the experimental results are then illustrated by a simulation study where an ILC algorithm is applied to a flexible two-mass model. The main observation in both simulations and experiments is that although the ILC algorithm reduces the error of the measured variable, the performance evaluated in terms of the controlled variable can be worse due to model errors. The aim of this chapter is to illustrate the case when an ILC algorithm is applied to a system where the controlled variable is not the measured variable. The discussion serves as a motivation to the work presented in Chapters 6 to 8, where an estimate of the controlled variable drives the ILC algorithm. The experimental parts of the chapter are based on Wallén et al. [2007a, 2008b, 2010a,b], while more details about the simulation study can be found in Wallén et al. [2009b].

5.1 Problem description

A modern industrial robot contains mechanical flexibilities due to a number of reasons. As a result of the cost-driven development of industrial robots, less rigid mechanical structures are developed. This results in a larger number of mechanical vibration modes and lower resonance frequencies of the robots. The cost reduction also results in robot components with larger individual variation [Moberg, 2010, Brogårdh, 2007, 2009].

A difficulty in industrial applications is to measure the actual robot tool position in an acceptable way, both economically and practically. Therefore, normally only the motor angular positions are measured in commercial industrial robot systems [Brogårdh, 2009, Spong et al., 2006], including the robots used in the experiments in this chapter. The control objective is however to follow a desired tool path. Assuming that correct kinematic and dynamic models of the robot are available, the tool position and orientation could theoretically be derived from the motor angular positions. For every robot individual it would require exact descriptions of complex phenomena, as for example friction, backlash, motor torque ripple and nonlinear stiffness of the gearboxes. In addition, a complete model of the mechanical flexibilities of the robot structure[1] is necessary. This is however not realistic in practice, and the derivation of the tool position from the motor angular positions suffers from model uncertainties.

ILC has traditionally been applied to systems where the controlled variable is measured, some recent examples are Moore et al. [2005], Dabkowski et al. [2010] and Abdellatif and Heimann [2010]. This property is explicitly formulated in the postulate $\mathbf{P}_5'$ of Arimoto[2]. When applying an ILC algorithm to a flexible system based on the position measured on one side of the mechanical flexibility, the postulate $\mathbf{P}_5'$ is satisfied. However, even though the error of the measured variable

[1] See the extended flexible joint dynamic model, summarised in Section 2.3.3 and presented in detail in Moberg [2010], for work in that direction.

[2] See Section 4.5.

decreases with the iterations, the position measured on the other side of the mechanical flexibility converges but towards an incorrect signal when model uncertainties are present. This will be illustrated in both simulations and experiments in the chapter.

5.2 Experiments on a serial robot

The experiments on a serial robot are performed on a large-size commercial robot from ABB with six joints. A robot with similar load capacity (175 kg) can be seen in Figure 5.2. In the experiments the conventional robot controller, implemented by ABB in the IRC5 control system, works in parallel with the ILC algorithm[3].

Figure 5.2: *Example of a large-size industrial robot from ABB with similar size and load capacity (175 kg) as the robot used in the experiments. Photo from ABB Robotics [2007].*

5.2.1 Experimental setup

Robot system

In the experiments the robot tool is supposed to perform a small circle with radius 5 mm. The motion is programmed with a tool velocity of 40 mm/s and the experiments studied in this chapter are performed in an operating point with the robot configuration seen in Figure 5.3. Experiments covering other robot configurations, as well as a broader range of ILC design variables are presented in Wallén et al. [2007a, 2008b]. The resulting tool path is measured by the laser-measurement system LTD500 from Leica Geosystems [2010]. Measurements from the Leica system can be used in the ILC algorithm, as is described in for example Gunnarsson et al. [2006]. It is an expensive measurement system and therefore it is used strictly as an evaluation tool in this chapter.

[3] See Norrlöf [2000] for a detailed description of the experimental setup.

ILC algorithm

The robot is a multivariable system, but for simplicity the robot joints will be treated individually as decoupled systems with a separate ILC algorithm for each joint[4]. Errors originating from the dynamic coupling between the joints will be treated as disturbances. To support the assumption with identical initial conditions for all experiments, the robot starts from the same position with zero velocity. In Figure 5.4a the reference tool path is shown. The robot performs an introductory motion (lead-in), followed by the circle, and ends with a lead-out back to the initial position. The whole trajectory is to be learned by the ILC algorithm, while the performance is evaluated on the circular path, see Figure 5.4b.

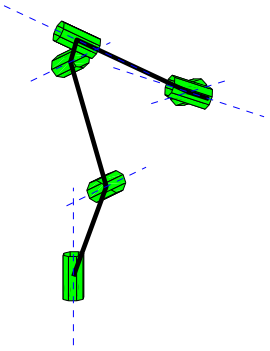
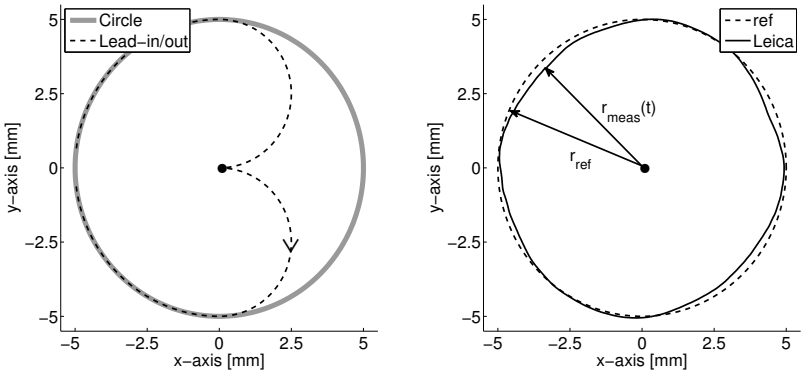


Figure 5.3: *The robot configuration for the operating point.*



(a) *Reference path* ***(b)*** *Performance evaluation*

Figure 5.4: *a) Reference path for the robot tool position with lead-in/lead-out, b) performance evaluated on the circular path, see* (5.6) *to* (5.7).

[4]Compare independent joint control in Section 2.4.

The first-order ILC algorithm (4.12),

$$u_{k+1}(t) = Q(q)\Big(u_k(t) + L(q)e_k(t)\Big) \tag{5.1}$$

is applied to the motors for each of the six robot joints, with the error

$$e_k(t) = r_m(t) - q_{m,k}(t) \tag{5.2}$$

consisting of the motor angular position reference $r_m(t)$ and the measured motor angular position $q_{m,k}(t)$ at iteration k. The ILC input signal $u_k(t)$ is added to the motor angular position reference. All signals are defined on a finite time interval $t = nT_s$, $n \in [0, N-1]$ with N number of samples and sampling interval $T_s = 4\,\text{ms}$. The filter $Q(q)$ is chosen as a zero-phase low-pass filter, and the filter $L(q)$ consists of a time shift of δ samples and a learning gain γ, resulting in $L(q) = \gamma q^{\delta}$. The ILC update equation can explicitly be written as

$$u_{k+1}(t) = Q(q)\Big(u_k(t) + \gamma e_k(t + \delta T_s)\Big) \tag{5.3}$$

with the ILC design variables:

- Type and order of the filter $Q(q)$.
- Cutoff frequency f_c of the filter $Q(q)$.
- Learning gain γ, with $0 < \gamma \leq 1$.
- Time shift δ.

In this chapter the design variable f_c and its influence on the ILC algorithm performance is investigated, while the other ILC design variables remain constant[5]. A second-order Butterworth filter with cutoff frequency f_c is applied using the MATLAB function `filtfilt`, to get a zero-phase behaviour of the resulting filter $Q(q)$. The learning gain $\gamma = 0.9$ is motivated by a trade-off between convergence rate and robustness for each of the robot joints, having a static gain equal to 1. The time shift is chosen as $\delta = 3$, and is experimentally validated to give a stable ILC system. The same ILC design variables are used for all six joints for simplicity reasons and the learning is stopped after five iterations.

5.2.2 Performance measures

For the evaluation of the experimental results, error measures are defined for the performance of the joints and the tool, respectively. The measures are computed for the circular path, not including the lead-in/lead-out parts, see Figure 5.4.

Performance of the joints

The results for each of the six joints are compared to the nominal error when no ILC algorithm is applied ($k = 0$). The nominal error for joint number $i = 1, \ldots, 6$ is given in vector form as in

$$\boldsymbol{e}_0^i = \begin{pmatrix} e_0^i(0) & \ldots & e_0^i((N-1)T_s) \end{pmatrix}^T \tag{5.4}$$

[5] See also Wallén et al. [2007a, 2008b] for experiments with varying robot configuration, time shift δ and a wider range of cutoff frequencies f_c.

for N number of samples along the circle. The reduction of the norm of the motor angular position error (5.2) for joint $i = 1, \ldots, 6$ and iteration k is given by

$$J_k^i = 100 \cdot \frac{\|\boldsymbol{e}_k^i\|}{\max_l \|\boldsymbol{e}_0^l\|} \quad [\%] \tag{5.5}$$

The value is normalised with respect to the largest nominal error (5.4) for all joints. The error measure (5.5) is studied in both 2-norm and ∞-norm.

Performance of the tool

The root mean square (RMS) error of the tool position at iteration k is

$$e_k^{\text{RMS}} = \sqrt{\frac{1}{N} \sum_{t=0}^{(N-1)T_s} \left(r_{\text{ref}} - r_{\text{meas},k}(t)\right)^2} \tag{5.6}$$

where $r_{\text{meas},k}(t)$ denotes the radius of the measured circle at time t, see Figure 5.4b. The radius of the reference circle is $r_{\text{ref}} = 5\,\text{mm}$. The maximum deviation from the reference circle at iteration k is defined as

$$e_k^{\max} = \max_{t=nT_s, n\in[0,N-1]} \left(|r_{\text{ref}} - r_{\text{meas},k}(t)|\right) \tag{5.7}$$

The error measures (5.6) to (5.7) are normalised by the largest nominal error,

$$J_k^{\text{RMS}} = 100 \cdot \frac{e_k^{\text{RMS}}}{e_0^{\text{RMS}}} \quad [\%] \tag{5.8}$$

$$J_k^{\max} = 100 \cdot \frac{e_k^{\max}}{e_0^{\max}} \quad [\%] \tag{5.9}$$

The error measure (5.5) in 2-norm of each joint corresponds to the RMS error of the tool position, J_k^{RMS} in (5.8). The error measure (5.5) in ∞-norm of each joint corresponds to the maximum deviation of the tool position error, $J_k^{\max}$ in (5.9).

5.2.3 Experimental results

Reduced motor angular position and tool position errors

An experiment is performed with the ILC design variable $f_c = 10\,\text{Hz}$ in the filter $Q(q)$. First, the reduction of the motor angular position error (5.2) is evaluated. The error measure J_k^i in (5.5) expressed in ∞-norm is illustrated in Figure 5.5 for each iteration. The error measure (5.5) in 2-norm has similar appearance. It can be seen that the performance is improved for all joints when the ILC algorithm is applied to the robot.

The performance of the tool is evaluated by the error measures (5.8) to (5.9), and the result from the experiment is shown in Figure 5.6. Both J_k^{RMS} and $J_k^{\max}$ of the tool position error have a decreasing trend with respect to the iterations. Figure 5.7 shows the resulting tool position for each iteration, and it can be seen that the measured circle is close to the reference circle after five iterations.

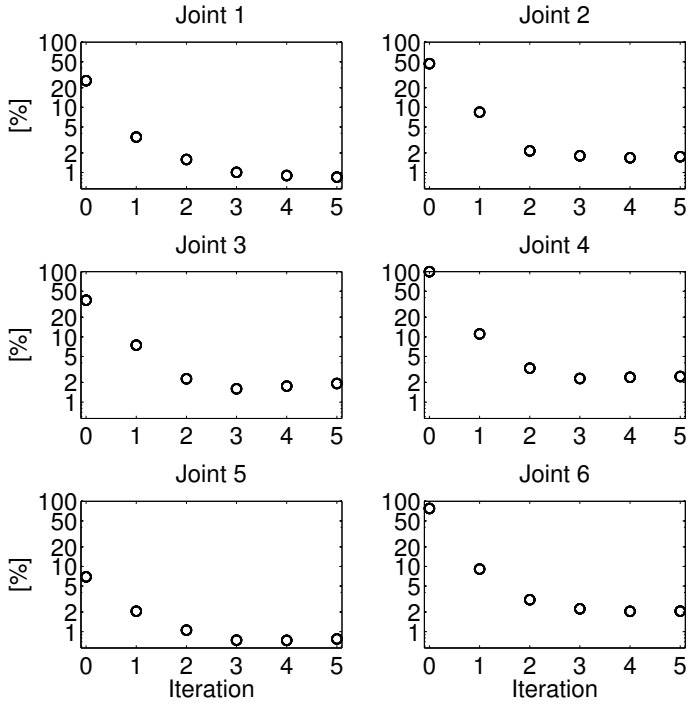


Figure 5.5: *The error measure J_k^i, see* (5.5)*, expressed in ∞-norm for all joints $i = 1, \ldots, 6$ for each iteration. The experiment is performed with the ILC design variable $f_c = 10$ Hz and shows an improved performance for all joints. Note the logarithmic scale.*

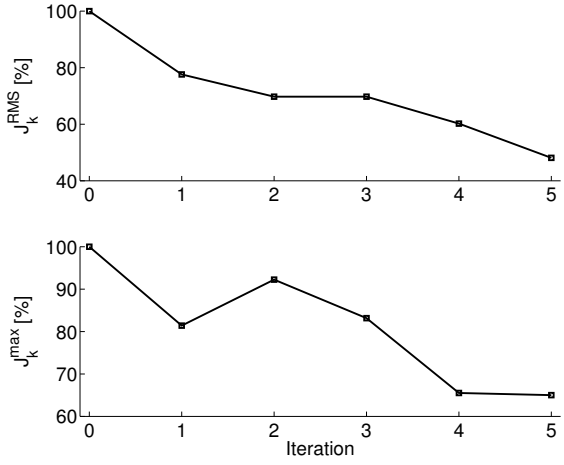


Figure 5.6: *Tool-position error evaluated by J_k^{RMS} from* (5.8)*, and J_k^{max} from* (5.9)*, for each iteration. In the experiment the ILC design variable is $f_c = 10$ Hz.*

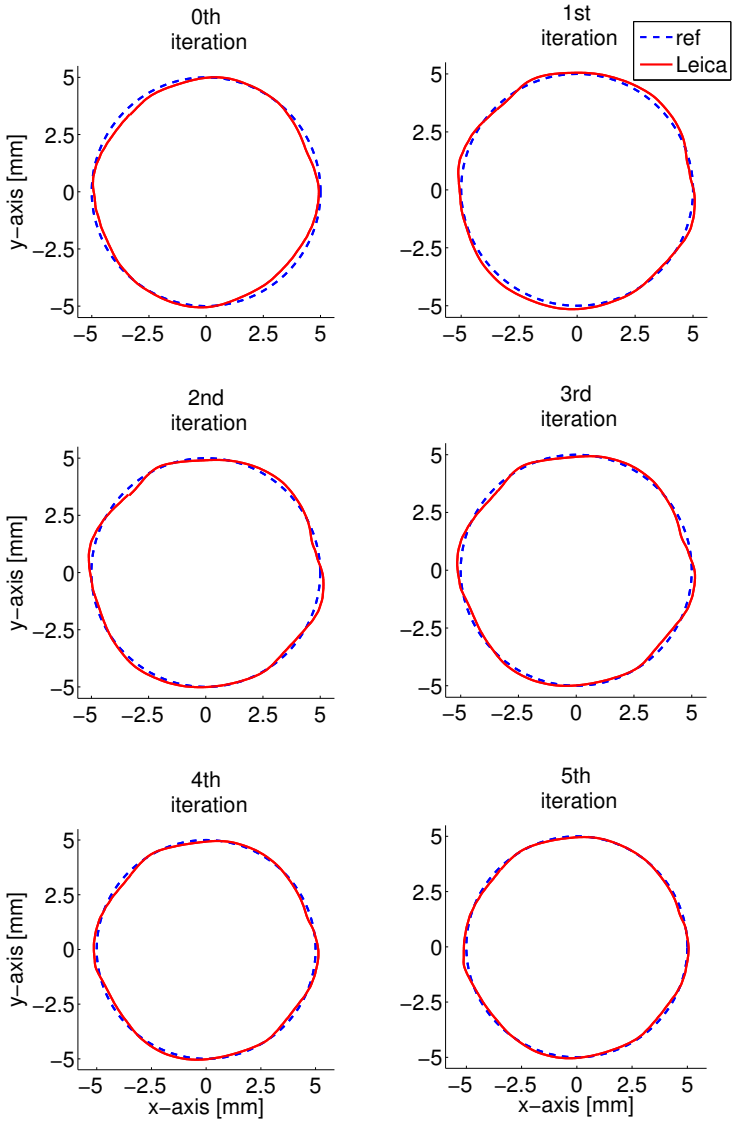


Figure 5.7: *Measured circles of the tool at every iteration, compared to the reference circle. The experiment is performed with ILC design variable* $f_c = 10\,Hz$*. After five iterations the measured circle is close to the reference.*

The experiment shows that a simple ILC design, a first-order ILC algorithm with the same design parameters for all joints, can significantly reduce both the errors for each joint and the resulting tool position error within only a few iterations.

Reduced motor angular position errors, but increased tool position error

The cutoff frequency of $Q(q)$ directly affects the bandwidth of the ILC algorithm. The resulting performance with respect to cutoff frequency is investigated in experiments when $f_c = 10$ and 15 Hz. In Figure 5.8 the error measure J_k^i (5.5) in ∞-norm for each of the joints are shown and it can be seen that a higher cutoff frequency gives a larger reduction of the motor angular position errors. This can be explained by the fact that with a higher cutoff frequency, a larger part of the error is taken into account in the ILC update equation (5.3) and can be corrected.

From the corresponding error measures (5.8) to (5.9) for the tool, shown in Figure 5.9, it can be concluded that the tool-path error is increased for $f_c = 15$ Hz compared to $f_c = 10$ Hz. This is an opposite result compared to the error measure (5.5) of the joints presented in Figure 5.8. An oscillatory behaviour of the tool is noticed after a few iterations, as can be seen in Figure 5.10, where the tool performance at the 0th and 5th iteration are compared. The oscillations are due

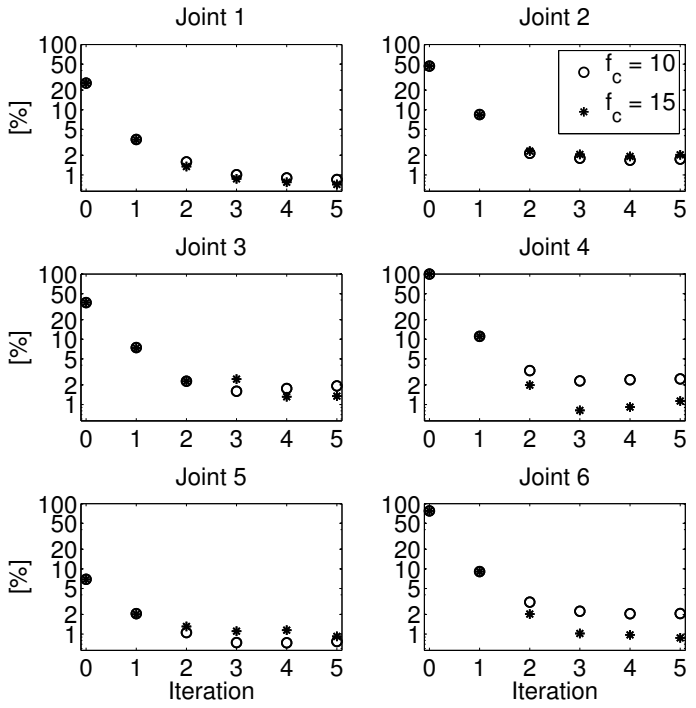


Figure 5.8: *The error measure J_k^i, see* (5.5)*, expressed in ∞-norm for all joints $i = 1, \ldots, 6$, cutoff frequencies $f_c = 10$ and 15 Hz for each iteration. A higher cutoff frequency gives a larger reduction of the errors.*

to the ILC algorithm and the ILC input signal $u_k(t)$, since the only differences between the iterations is the added ILC input signal.

Summary

The experiments show that a simple ILC algorithm applied to the complex robot structure can effectively reduce the motor angular position errors. The behaviour of the tool is more complicated. In the first experiment, with the ILC design variable $f_c = 10\,\text{Hz}$, it is illustrated that also the tool performance is improved. In the second experiment a higher cutoff frequency, $f_c = 15\,\text{Hz}$, is used. In the experiment it is shown that although the ILC algorithm reduces the motor angular position errors, this does not necessarily imply improved tool performance.

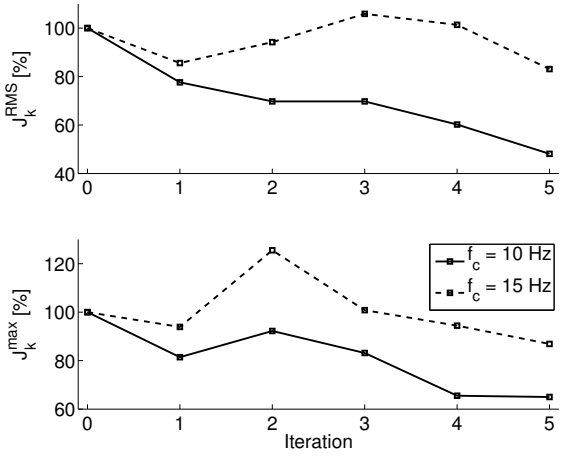


Figure 5.9: *Tool-position error evaluated by J_k^{RMS} from (5.8), and J_k^{max} from (5.9), for each iteration. Higher cutoff frequency f_c gives larger errors.*

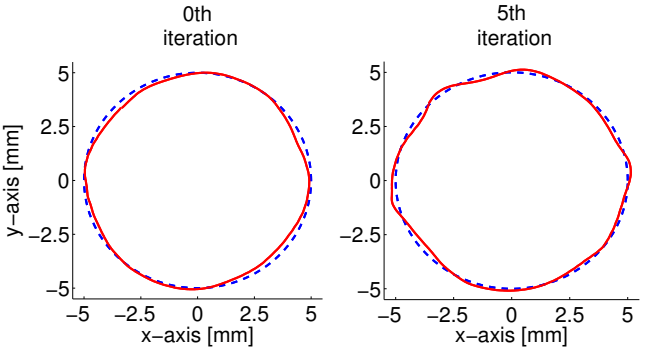


Figure 5.10: *Measured circle of the tool when the experiment is performed with the ILC design variable $f_c = 15\,\text{Hz}$. An oscillatory behaviour is noticed for iteration 5 compared to iteration 0.*

5.3 Experiments on a parallel robot

The experiments on a parallel robot are performed on the Gantry-Tau parallel robot prototype [Johannesson et al., 2004], which is schematically pictured in Figure 5.11. The robot is designed to have a large workspace compared to other parallel robot structures, while still being stiff compared to serial robots.

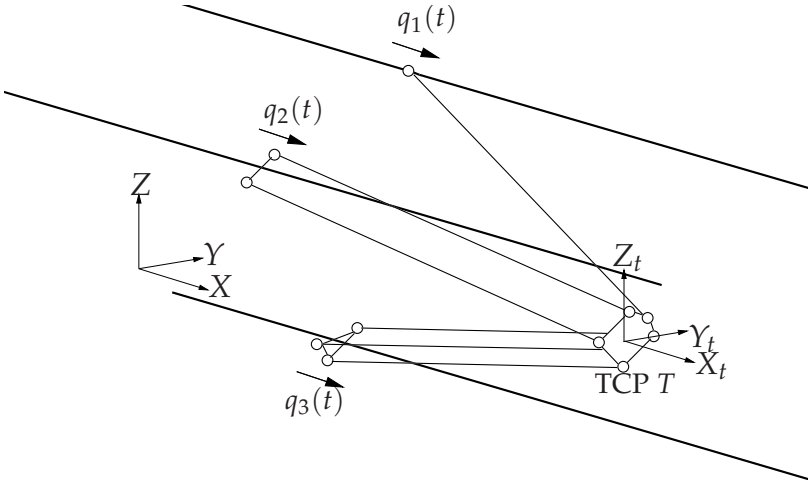


Figure 5.11: *Schematic picture of the Gantry-Tau parallel robot, with the base coordinate frame XYZ, the coordinate frame $X_tY_tZ_t$ of the end-effector plate and the tool position T. Three carts move on guideways. The corresponding cart positions are denoted $q_1(t)$, $q_2(t)$ and $q_3(t)$.*

5.3.1 Experimental setup

Robot system

The Gantry-Tau robot has three kinematic chains, where each chain is driven by a linear actuator consisting of a cart moving on a guideway. The three carts are connected to the end-effector plate via link clusters, resulting in three purely translational DOFs. In the experiments, the robot tool is supposed to perform a rectangular motion with side 10 mm and with a high programmed tool velocity of 100 mm/s. Two length gauges from Heidenhain [2010] are available for measuring the resulting tool position in the X- and Y-direction, respectively.[6]

The individual control loop of the motor for each cart can be approximated by a low-pass filter with five samples delay, with the delay caused by internal data communication in the IRC5 control system. From experiments when a force pulse is generated by the impact of a hammer to the end-effector plate in the X- and Y-direction, respectively, it is concluded that the light-weight robot links and

[6] See Section 8.2 for more information about the robot, control system and measurement devices.

the mechanical framework introduces flexibilities in the robot structure[7]. The first resonance at around 11 Hz in Y-direction is larger in magnitude than the resonance in X-direction and is consequently dominant for the robot behaviour.

ILC algorithm

An ILC algorithm of the type (5.3) is used in the experiments. As the heuristic algorithm design[8] relies on only minor knowledge of the system (static gain and time delay of the system), it is a suitable candidate for the experiments. It results in a filter $L(q)$ consisting of a time shift of $\delta = 5$ samples and a learning gain of $\gamma = 0.9$, based on a time delay of five samples and unity static gain of the closed-loop system for each motor. A second-order low-pass Butterworth filter with cutoff frequency $f_c = 10\,\text{Hz}$ is chosen, implemented by filtering the signal forwards-backwards in time, to obtain a zero-phase filter $Q(q)$. An ILC algorithm is implemented on each of the three carts independently, using the individual motor angular position errors, with the same ILC design variables for all carts.

5.3.2 Performance measures

The experimental evaluation is based on the nominal error when no ILC algorithm is applied (denoted $k = 0$). The nominal error for the three carts is

$$\begin{pmatrix} \boldsymbol{e}^1_{m,0} \\ \boldsymbol{e}^2_{m,0} \\ \boldsymbol{e}^3_{m,0} \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}^1_m \\ \boldsymbol{r}^2_m \\ \boldsymbol{r}^3_m \end{pmatrix} - \begin{pmatrix} \boldsymbol{q}^1_{m,0} \\ \boldsymbol{q}^2_{m,0} \\ \boldsymbol{q}^3_{m,0} \end{pmatrix} \tag{5.10}$$

with motor angular position error $\boldsymbol{e}^i_{m,0}$ and measured motor angular position $\boldsymbol{q}^i_{m,0}$ for cart i at iteration 0 and $\boldsymbol{r}^i_m$ being the corresponding reference. The nominal tool-position error is analogously defined as

$$\begin{pmatrix} \boldsymbol{e}^X_{z,0} \\ \boldsymbol{e}^Y_{z,0} \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}^X_z \\ \boldsymbol{r}^Y_z \end{pmatrix} - \begin{pmatrix} \boldsymbol{z}^X_0 \\ \boldsymbol{z}^Y_0 \end{pmatrix} \tag{5.11}$$

in the X- and Y-direction with the tool-position reference $\boldsymbol{r}_z$ and $\boldsymbol{z}_0$ denoting the measured tool position at iteration 0.

The reduction of the 2-norm of the motor error at iteration k is given in percentage of the nominal motor error (5.10), as in

$$\bar{e}^1_{m,k} = 100 \cdot \frac{\|\boldsymbol{e}^1_{m,k}\|_2}{\|\boldsymbol{e}^1_{m,0}\|_2} \quad [\%] \tag{5.12a}$$

$$\bar{e}^2_{m,k} = 100 \cdot \frac{\|\boldsymbol{e}^2_{m,k}\|_2}{\|\boldsymbol{e}^2_{m,0}\|_2} \quad [\%] \tag{5.12b}$$

$$\bar{e}^3_{m,k} = 100 \cdot \frac{\|\boldsymbol{e}^3_{m,k}\|_2}{\|\boldsymbol{e}^3_{m,0}\|_2} \quad [\%] \tag{5.12c}$$

[7] See a description of the identification experiment and the result in Section 8.4.
[8] See Algorithm 4, Section 4.8 for the details.

The reduction of the 2-norm of the tool-position error is similarly given by

$$\bar{e}^X_{z,k} = 100 \cdot \frac{\|\boldsymbol{e}^X_{z,k}\|_2}{\|\boldsymbol{e}^X_{z,0}\|_2} \quad [\%] \tag{5.13a}$$

$$\bar{e}^Y_{z,k} = 100 \cdot \frac{\|\boldsymbol{e}^Y_{z,k}\|_2}{\|\boldsymbol{e}^Y_{z,0}\|_2} \quad [\%] \tag{5.13b}$$

The quantities (5.12) and (5.13) are based on the part of the trajectory to be learned by the ILC algorithm.

5.3.3 Experimental results

The experimental results when applying the ILC algorithm to each of the three carts independently, using the individual motor angular position errors, are presented in Figure 5.12. The error measure (5.12) is reduced to nearly 2 % for the three carts after approximately five iterations. The corresponding cart position is derived by dividing the measured motor angular position by the gear ratio. The resulting cart position errors after 10 iterations are compared to the nominal cart position errors in Figure 5.13. It can clearly be seen that by applying an ILC algorithm to the individual carts of the Gantry-Tau parallel robot, the cart position errors for the part of the trajectory to be learned ($t = 1.6 - 4.1$ s) are significantly reduced.

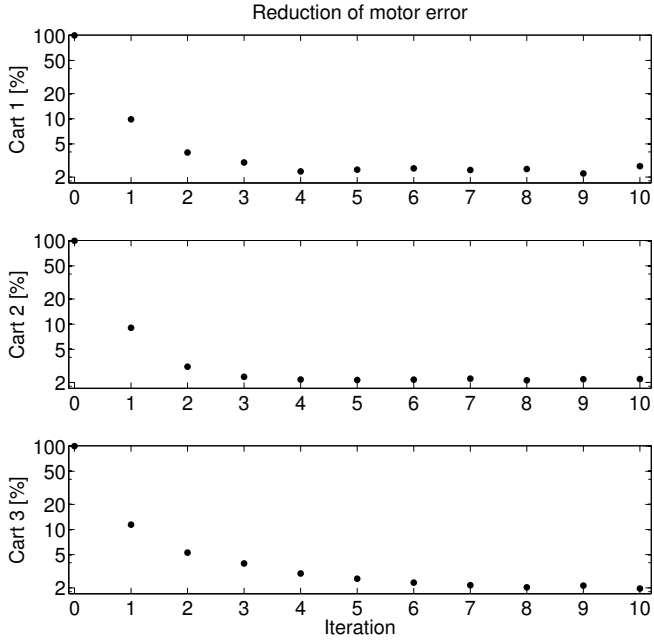


Figure 5.12: *Error measure* (5.12) *of the three carts.*

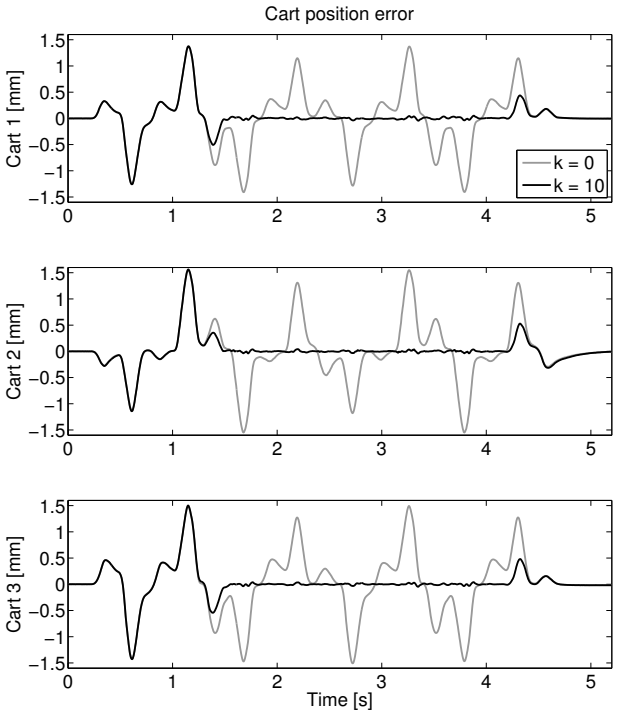


Figure 5.13: *Cart position errors after 10 iterations compared to the nominal errors for the three carts. The errors for the part of the trajectory to be learned ($t = 1.6 - 4.1$ s) are significantly reduced.*

The motor angular position references to be followed by the motors are calculated from the rectangular reference path of the tool transformed by the inverse kinematics. The path calculated from the measured motor angular positions transformed by the forward kinematics, called "kinematics" in Figure 5.14, represents the corresponding tool path if the robot was stiff. Since the ILC algorithm for each cart is based on the motor angular position error, the kinematics path compared to the reference path reflects the remaining control errors seen from the motors, which are very small. For low velocities, much of the error components of the robot tool position can be corrected by using ILC algorithms based on the motor angular position errors, since the influence of the flexibilities in the robot structure is small. When performing ILC experiments at higher velocity, as in this chapter, the effects of the weak structure cannot be controlled by using only measurements of the motor angular positions. This is the case here, which can be seen in Figure 5.14 by the deviation of the measured tool position from the reference tool path.

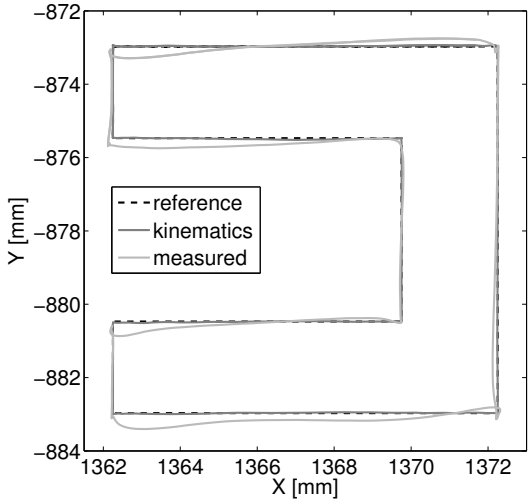


Figure 5.14: *Tool performance after 10 iterations. The reference path (reference) is compared to the resulting path computed from the measured motor angular positions transformed by the forward kinematics (kinematics) and the measured tool position (measured).*

5.4 Simulation study

The results from the ILC experiments in the previous sections all have in common that the errors used in the ILC update equation, the motor angular position errors, are significantly reduced. However, in the experiments on the serial robot, the resulting tool position is improved when using one set of ILC design variables, while another set gives worsened tool behaviour. In the parallel robot experiments, the reduction of the tool-position error is smaller than the reduction of the motor angular position errors.

The experimental results illustrate that one has to be careful when dealing with systems with unmodelled dynamics. The ILC algorithm can increase the oscillations in the system. This can be the case in particular when the controlled variable, here the robot tool position, is not directly measured and included in the ILC algorithm. Therefore, a simulation study is performed by applying an ILC algorithm to the flexible two-mass model illustrated in Figure 5.15, which represents an idealised model of a single robot joint.

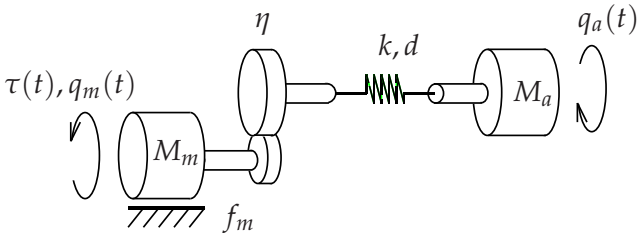


Figure 5.15: *A flexible two-mass model characterised by spring coefficient k, damping coefficient d, viscous friction f_m, gear ratio η, moments of inertia M_m, M_a, motor angular position $q_m(t)$, joint angular position $q_a(t)$ and motor torque $\tau(t)$, generated by a torque constant k_τ times the input voltage.*

5.4.1 Simulation setup

Two-mass system

Two variables are needed to describe the deflection of the flexible joint; the motor angular position $q_m(t)$ seen from the motor side of the joint and the joint angular position $q_a(t)$ seen from the link side of the joint. In this simulation example it is assumed that the motor angular position can be measured, while the controlled variable is the joint angular position. The model equations are given in Section 2.3.3, with the model parameter values in Table 5.1. The parameter values correspond with some minor modifications to the ones obtained for the flexible robot used in Gunnarsson et al. [2007] for studying ILC applied to flexible mechanical systems.

In the simulation, a discrete-time PD-controller including a low-pass filter regulates the motor angular performance. Moderate requirements for the feedback

can be chosen, since the desired servo performance is achieved by the ILC algorithm. The controller $F(q)$ is obtained by manual tuning, resulting in

$$F(q) = K_1 + \frac{K_2 q - K_3}{q - K_4}$$

with the controller parameters given in Table 5.2. A block diagram[9] of the system and controller is given in Figure 5.16. The motor angular position $q_m(t)$ is related to the joint angular position $q_a(t)$ by the true system $T^0_{yz}(s)$, see (2.12b). From the figure, it can also be seen that the ILC input signal $u_k(t)$ is added to the motor angular position reference $r_m(t)$. The closed-loop system is driven by a joint angular position reference $r_a(t)$, here chosen as a filtered step. The corresponding motor angular position reference $r_m(t)$ is then computed from

$$r_m(t) = F_r(q) r_a(t) \tag{5.14}$$

The pre-filter $F_r(q)$ is given as the inverse of a sampled version of a model $T_{yz}(s)$ of the true system $T^0_{yz}(s)$, multiplied by a factor $1/q$ to make $F_r(q)$ proper. The closed-loop system and ILC algorithm are simulated using SIMULINK with a sampling interval of $T_s = 0.01$ s.

Table 5.1: *Model parameter values.*

$\eta = 0.2$	$M_m = 0.0021$	$M_a = 0.0991$	$k = 8$
$d = 0.0924$	$f_m = 0.0713$	$k_\tau = 0.122$	

Table 5.2: *Controller parameter values.*

$K_1 = 5$	$K_2 = 2$	$K_3 = 2$	$K_4 = 0.905$

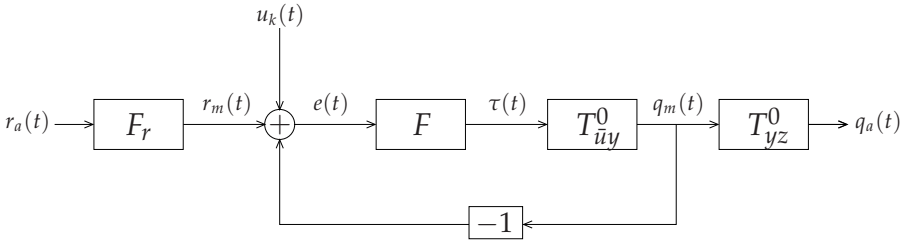


Figure 5.16: *The system illustrated by the true systems $T^0_{\bar{u}y}$ and T^0_{yz}, relating motor torque $\tau(t)$ to motor angular position $q_m(t)$, and motor angular position $q_m(t)$ to joint angular position $q_a(t)$, respectively. F represents the controller. The reference $r_a(t)$ is filtered by the pre-filter F_r, giving $r_m(t)$, and the ILC input $u_k(t)$ is added to $r_m(t)$.*

[9] In Figure 5.16 the subscripts of the transfer operators follow the notation in Chapter 6, with input $\bar{u}(t)$ to the system, measured variable $y(t)$ and controlled variable $z(t)$.

ILC algorithm

A first-order ILC algorithm of the type (5.3) is applied to the two-mass system, with learning gain γ and time shift of δ samples, see Table 5.3 for the numerical values. The filter $Q(q)$ is implemented by forward-backward filtering of the signal through a causal second-order low-pass Butterworth filter with cutoff frequency f_c Hz to obtain zero-phase characteristics.

Table 5.3: *ILC design parameter values.*

$\gamma = 0.95$	$\delta = 10$	$f_c = 9$

5.4.2 Simulation results

The resulting improvement of the joint angular position, the controlled variable, is of interest. In the simulation example, the motor angular position error

$$e_k(t) = r_m(t) - q_{m,k}(t) \tag{5.15}$$

is used in the ILC algorithm (5.3). Assume that the relation between the measured variable $q_m(t)$ and controlled variable $q_a(t)$ is completely known, see Figure 5.16, and that a stable inverse exists. A correct motor angular position reference $r_m(t)$ can then be derived from the joint angular position reference according to

$$r_m(t) = \left(T_{yz}(q)\right)^{-1} r_a(t) \tag{5.16}$$

The resulting motor angular position $q_m(t)$ is close to the reference $r_m(t)$ when the ILC algorithm has converged, as is illustrated in Figure 5.17. The joint angular position $q_a(t)$, the controlled variable, is thereby also driven towards the correct reference signal $r_a(t)$ for the case with nominal system parameters, seen in Figure 5.18. The improvement compared to the result when no ILC algorithm is applied is clearly seen.

An uncertain model $T_{yz}(q)$ would give an incorrect value of $q_m(t)$ and thereby also incorrect $q_a(t)$, since a reference $r_m(t)$ derived from (5.16) would be followed. This does not necessarily imply that the resulting joint angular position error is reduced. It is illustrated by modifying the joint stiffness parameter by +40 %, motivated by the case when the stiffness of the gearbox is uncertain. Applying the same ILC algorithm to the modified system results in a motor angular position $q_m(t)$ following the desired trajectory in Figure 5.17. The resulting joint angular position, see Figure 5.18, is however deteriorated when model errors are introduced, since the joint angular position converges towards an incorrect reference signal due to the model error. The motor angular position reference is thereby essential for the resulting joint angular position after convergence.

Similar results could be seen in the previous experiments. When applying the ILC algorithms using motor angular positions to the robot, the motor performance is significantly improved. Learning of the frequency components of the

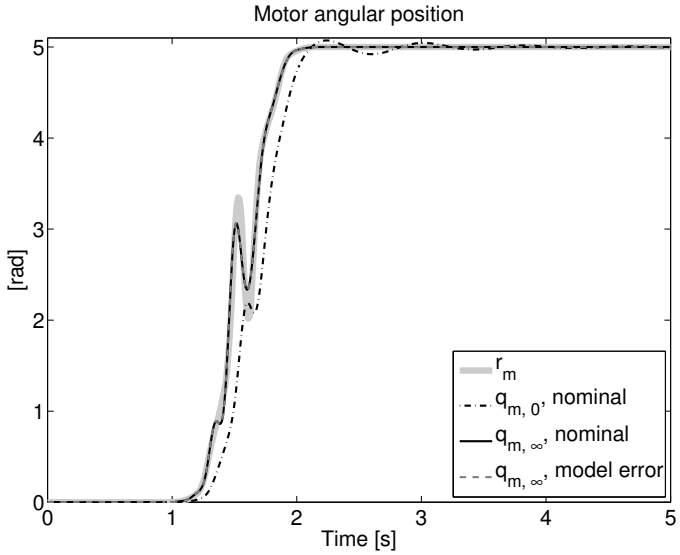


Figure 5.17: *Resulting motor angular position when the ILC algorithm has converged (denoted ∞). The performance is improved both in the nominal case and the case with model error, compared to when no ILC algorithm is applied (denoted 0).*

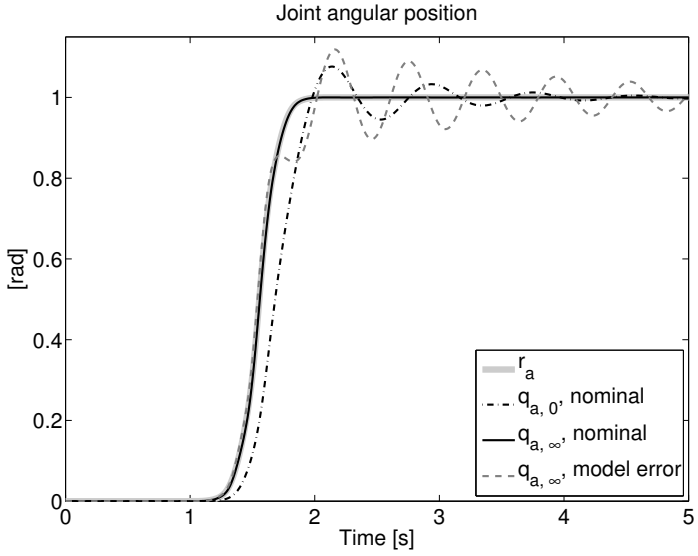


Figure 5.18: *Resulting joint angular position when the ILC algorithm has converged (denoted ∞). The performance is improved in the nominal case. However, when introducing model errors, the performance is deteriorated compared to when no ILC algorithm is applied (denoted 0).*

error up to around 10 Hz for the serial robot also implies improved tool performance. For higher frequencies of the robot system, the motor angular position reference does not correspond to the desired tool reference due to uncertainties in the model of the robot dynamics. For example, if the dominating resonance frequency of the robot is not modelled sufficiently well, it means that although the part of the motor angular position error up to and above this frequency is reduced, an oscillatory behaviour could appear in the tool position since an incorrect reference signal is followed. For the experiments with the Gantry-Tau parallel robot, the motor angular position reference is calculated from the tool position reference transformed by the inverse kinematics. In this case reduced motor angular position error does not mean a similar level of reduction of the tool position error. The flexible dynamic structure of the robot is not taken into account at all in the generation of the motor reference.

The results from both experiments and simulations clearly motivate the need for using an estimate of the controlled variable in the ILC algorithm, that is,

$$e_k(t) = r_a(t) - \hat{q}_{a,k}(t) \tag{5.17}$$

in the case with the two-mass model. One way of forming the estimate $\hat{q}_{a,k}(t)$ of the joint angular position is by using the nominal model $T_{yz}(q)$, as in

$$\hat{q}_{a,k}(t) = T_{yz}(q) q_{m,k}(t) \tag{5.18}$$

By inserting (5.18) into (5.17), the error (5.17) used in the ILC algorithm can be rewritten as the motor angular position error filtered through the nominal model $T_{yz}(q)$, as in

$$e_k(t) = T_{yz}(q)\Big(r_m(t) - q_{m,k}(t)\Big)$$

which has similarity with (5.16). The resulting system performance $q_a(t)$ clearly depends on the accuracy of the model. An alternative way to obtain estimates of the controlled variable is to use additional sensors, like accelerometers, in combination with signal processing and estimation techniques, see for example Karlsson and Norrlöf [2005]. These estimates can then be used in an ILC algorithm to be able to improve the performance of the controlled variable. Estimation-based ILC will be the topic for Chapters 6 to 8.

5.5 Conclusions

Traditionally ILC has been applied to systems where the controlled variable also is the measured variable. In industrial robot applications this is typically not the case; in a commercial robot system the motor angular positions are measured, while the control objective is to follow a desired tool path. Measuring the tool position in an acceptable way both economically and practically is difficult.

This chapter discusses experimental results when an ILC algorithm using measured motor angular positions is applied to both a serial robot and a parallel robot. It is shown that an improved motor angular position does not necessar-

ily imply an improved tool position. The underlying problem can be described by difficulties in improving performance of the controlled variable by ILC algorithms based on only measured variables when the dynamic model suffers from uncertainties. This problem is studied in simulations of a flexible two-mass system, representing an idealised model of a single robot joint. When only the motor angular position error is used in the ILC update equation, the joint angular position converges towards an incorrect signal when model errors are introduced, as expected. This is explained by the motor angular position reference, computed from the inverse of the nominal system relating motor angular position to joint angular position.

One way of handling the difficulties described above is to use an estimate of the controlled variable in the ILC algorithm. The estimate could, for example, be obtained by using additional sensors in combination with signal processing and estimation algorithms. In Chapter 6 the theoretical basis for analysis of estimation-based ILC is given, together with simulations and experiments in Chapters 7 to 8.

6

A framework for analysis of estimation-based ILC

A FRAMEWORK FOR analysis of ILC algorithms is proposed for the situation when an ILC algorithm uses an estimate of the controlled variable. Under the assumption that the ILC input converges to a bounded signal, a general expression for the asymptotic error of the controlled variable is given. The expression for the asymptotic error is illustrated by a numerical example where an ILC algorithm is applied to a flexible two-mass model of a robot joint. The chapter is based on Wallén et al. [2009c, 2011].

6.1 Introduction

The aim of this chapter is to present a framework for analysis of the situation when an ILC algorithm is combined with a procedure for generating an estimate of the controlled variable. Estimation techniques and ILC have been combined in only a few publications. One example is Gunnarsson et al. [2007], where the ILC algorithm uses an estimate of the joint angular position, computed using measurements of the motor angular position and the joint angular acceleration of a flexible one-link robot arm. In Ratcliffe et al. [2006] a norm-optimal ILC algorithm is applied to a gantry robot, and the states are estimated by means of a full-state observer. In Schöllig and D'Andrea [2009] a state-space model is obtained by linearisation along the desired trajectory. The model error is then estimated using a Kalman filter in the iteration domain. The control signal at next iteration is given by minimising the deviation of the states from the desired trajectory. Another example is Tayebi and Xu [2003], where the estimated states for a class of time-varying nonlinear systems are used in an ILC algorithm and the asymptotic behaviour of the system is discussed. The focus in these papers is on specific estimation and/or ILC algorithm techniques, while the following

aspects are of main interest here:

- A framework for analysis of the properties of the ILC algorithm when an estimate of the controlled variable is used in the ILC algorithm.
- An expression for the asymptotic error of the controlled variable when an ILC algorithm using an estimate of the controlled variable has converged.

6.2 System description

Consider the system shown in Figure 6.1. The linear discrete-time system T has two inputs; the reference $r(t)$ and the ILC input signal $u_k(t)$, with k denoting the iteration number. The outputs are the measured variable $y_k(t)$ and the controlled variable $z_k(t)$. Throughout the chapter the variable $z_k(t)$ should follow the reference $r(t)$. The system T can have internal feedback, which means that it contains the system to be controlled as well as the controller.

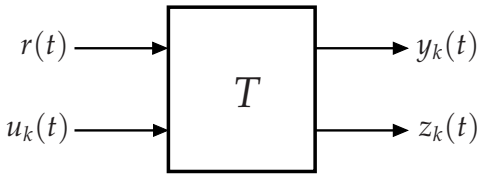


Figure 6.1: *Description of the system T with reference $r(t)$, ILC input $u_k(t)$, measured variable $y_k(t)$ and controlled variable $z_k(t)$ at iteration k.*

Denoting the true system by the superscript "0", the description of the system T at iteration k is

$$y_k(t) = T^0_{ry}(q)r(t) + T^0_{uy}(q)u_k(t) \tag{6.1a}$$

$$z_k(t) = T^0_{rz}(q)r(t) + T^0_{uz}(q)u_k(t) \tag{6.1b}$$

The transfer operators $T^0_{ry}(q)$, $T^0_{uy}(q)$, $T^0_{rz}(q)$ and $T^0_{uz}(q)$ are assumed to be stable and causal. System and measurement disturbances are not included here for simplicity reasons. In the following two examples it is illustrated how the system description (6.1) incorporates both open-loop and closed-loop systems.

Example 6.1: Open-loop system

First, an example of the description (6.1) is given by the stable open-loop system shown in Figure 6.2. The true system, denoted $G^0(q)$, is given by the discrete-time state-space description

$$x_k(t+1) = A^0 x_k(t) + B^0 u_k(t) \tag{6.2a}$$

$$y_k(t) = C^0 x_k(t) \tag{6.2b}$$

$$z_k(t) = M^0 x_k(t) \tag{6.2c}$$

with the ILC input signal $u_k(t)$ at iteration k. A sampling interval of $T_s = 1$ is assumed in the remainder of the chapter if nothing else is stated. The outputs are

the measured variable $y_k(t)$ and controlled variable $z_k(t)$. From (6.2) the transfer operators $T^0_{uy}(q)$ and $T^0_{uz}(q)$ are given by

$$T^0_{uy}(q) = C^0(qI - A^0)^{-1}B^0 \tag{6.3a}$$

$$T^0_{uz}(q) = M^0(qI - A^0)^{-1}B^0 \tag{6.3b}$$

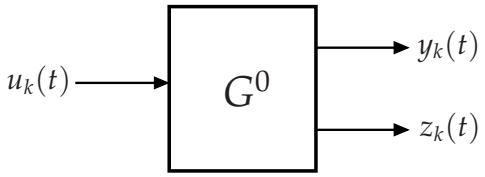


Figure 6.2: *Stable open-loop system; an example of the system description* (6.1) *with input $u_k(t)$ and outputs $y_k(t)$ and $z_k(t)$.*

Example 6.2: Closed-loop system

The system description (6.1) with inputs $r(t)$, $u_k(t)$ and outputs $y_k(t)$, $z_k(t)$ can also be exemplified by the closed-loop system shown in Figure 6.3. It means that the system T has internal feedback and contains both the system to be controlled and the controller in operation. In a real application the controller structure can be more complex than the one shown in Figure 6.3, where the structure considered here is used for illustration purposes.

The system to be controlled is given by the description (6.2), with input $\bar{u}_k(t)$, which is the control signal from the controller $F(q)$ at iteration k. The transfer operators are similarly to (6.3) denoted $T^0_{\bar{u}y}(q)$ and $T^0_{\bar{u}z}(q)$. Throughout the chapter it is assumed that the controller $F(q)$ is given, and that the closed-loop system is stable. Noting the choice of adding the ILC input $u_k(t)$ to the reference $r(t)$ in this example, it gives the closed-loop system from inputs $r(t)$ and $u_k(t)$ to out-

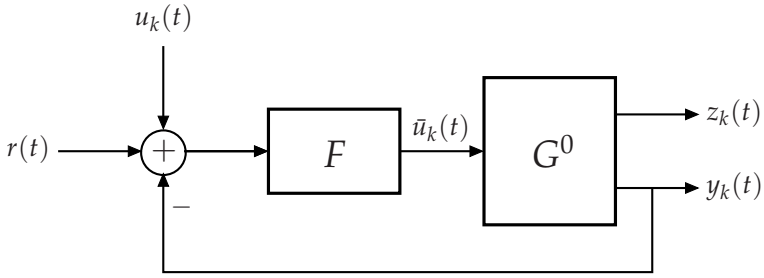


Figure 6.3: *Closed-loop system; an example of the system description* (6.1) *with inputs $r(t)$, $u_k(t)$ and outputs $y_k(t)$, $z_k(t)$. The input to $G^0(q)$ is $\bar{u}_k(t)$ from the controller $F(q)$.*

puts $y_k(t)$ and $z_k(t)$ as

$$\begin{aligned} y_k(t) &= \frac{F(q)T^0_{\tilde{u}y}(q)}{1+F(q)T^0_{\tilde{u}y}(q)}r(t) + \frac{F(q)T^0_{\tilde{u}y}(q)}{1+F(q)T^0_{\tilde{u}y}(q)}u_k(t) \\ &= T^0_{ry}(q)r(t) + T^0_{uy}(q)u_k(t) \end{aligned} \tag{6.4a}$$

$$\begin{aligned} z_k(t) &= \frac{F(q)T^0_{\tilde{u}z}(q)}{1+F(q)T^0_{\tilde{u}y}(q)}r(t) + \frac{F(q)T^0_{\tilde{u}z}(q)}{1+F(q)T^0_{\tilde{u}y}(q)}u_k(t) \\ &= T^0_{rz}(q)r(t) + T^0_{uz}(q)u_k(t) \end{aligned} \tag{6.4b}$$

where the transfer operators $T^0_{ry}(q)$, $T^0_{uy}(q)$, $T^0_{rz}(q)$ and $T^0_{uz}(q)$ can be identified by comparison with (6.1). They are stable according to the assumption about the closed-loop system.

Finally it can be noted that the system description (6.1) is a natural extension of the description (4.1), presented in Norrlöf and Gunnarsson [2002a]. With system and measurement disturbances omitted, the system description (4.1) is

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

This system is then controlled by an ILC algorithm and the properties of the resulting ILC system are analysed with focus on the measured variable $y_k(t)$ in Chapter 4, as well as in Norrlöf and Gunnarsson [2002a].

For the analysis in Section 6.5, a matrix description of the system and the ILC algorithm is used. The system description (6.1) is then rewritten in matrix form as

$$\boldsymbol{y}_k = \boldsymbol{T}^0_{ry}\boldsymbol{r} + \boldsymbol{T}^0_{uy}\boldsymbol{u}_k \tag{6.5a}$$

$$\boldsymbol{z}_k = \boldsymbol{T}^0_{rz}\boldsymbol{r} + \boldsymbol{T}^0_{uz}\boldsymbol{u}_k \tag{6.5b}$$

with the system matrices derived from the pulse-response coefficients of the corresponding transfer operators. This is further described in Section 4.4.

6.3 Estimation of the controlled variable

In the system description (6.1) the measured variable $y_k(t)$ and controlled variable $z_k(t)$ are involved. Consider a system where $y_k(t)$ and $z_k(t)$ in some sense reflects the same quantity, and assume that if $y_k(t)$ follows the reference $r(t)$, it does not give the desired value of $z_k(t)$. Obviously, if an ILC input $u_k(t)$ is applied to the system where the ILC algorithm is using the error $r(t) - y_k(t)$, it will drive the controlled variable $z_k(t)$ towards an incorrect value since the dynamics between $y_k(t)$ and $z_k(t)$ is neglected. It is therefore natural to use an estimate of $z_k(t)$, denoted $\hat{z}_k(t)$, in the ILC algorithm. The following representation of the estimate $\hat{z}_k(t)$ is proposed, where the estimate is generated using the refer-

ence $r(t)$, ILC input $u_k(t)$ and measured variable $y_k(t)$ as

$$\hat{z}_k(t) = F_r(q)r(t) + F_u(q)u_k(t) + F_y(q)y_k(t) \tag{6.6}$$

and the filters $F_r(q)$, $F_u(q)$ and $F_y(q)$ are assumed to be stable. See Figure 6.4 for an illustration of the system T together with the estimator (6.6). Using the matrix description, the relation (6.6) can be rewritten as

$$\hat{\boldsymbol{z}}_k = \boldsymbol{F}_r\boldsymbol{r} + \boldsymbol{F}_u\boldsymbol{u}_k + \boldsymbol{F}_y\boldsymbol{y}_k \tag{6.7}$$

with the matrices derived from the pulse-response coefficients of the filters $F_r(q)$, $F_u(q)$ and $F_y(q)$, similarly as in (4.3).

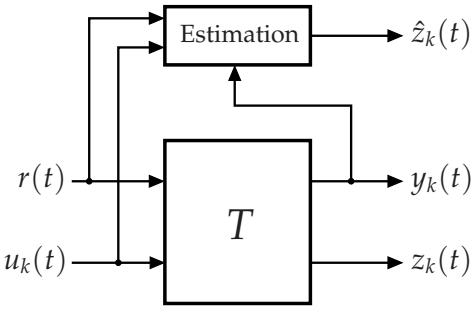


Figure 6.4: *Description of the system T with reference $r(t)$, ILC input $u_k(t)$, measured variable $y_k(t)$ and controlled variable $z_k(t)$ at iteration k. The estimator (6.6) results in an estimate $\hat{z}_k(t)$ of the controlled variable. The variable $z_k(t)$ should track the reference $r(t)$.*

The relation (6.6) will now be illustrated in three examples, showing how the estimate $\hat{z}_k(t)$ can be created. The cases are also discussed in more detail in Section 6.6.

Example 6.3: Case 1

A naive way of estimating $z_k(t)$ would be to neglect the dynamics between the measured variable $y_k(t)$ and controlled variable $z_k(t)$ and to let

$$\hat{z}_k(t) = y_k(t)$$

With the representation (6.6), this means choosing

$$F_r(q) = 0, \quad F_u(q) = 0, \quad F_y(q) = 1$$

Example 6.4: Case 2A

There are situations where the relationship between the measured variable $y_k(t)$ and controlled variable $z_k(t)$ can be explicitly described by the stable discrete-time relation[1]

$$z_k(t) = T_{yz}^0(q)y_k(t) \tag{6.8}$$

[1] See Section 6.6.5 for an example where (6.8) holds.

By using a nominal model $T_{yz}(q)$ of the relationship (6.8) between $y_k(t)$ and $z_k(t)$, it suggests an estimate in the form

$$\hat{z}_k(t) = T_{yz}(q)y_k(t) \tag{6.9}$$

which can be incorporated in the description (6.6) with

$$F_r(q) = 0, \quad F_u(q) = 0, \quad F_y(q) = T_{yz}(q)$$

In the third example $z_k(t)$ is estimated from the system input $\bar{u}_k(t)$ and measurement $y_k(t)$ using an observer. An important aspect of using observers is the ability of fusing information from additional sensors together with the original measurements available in the system. This is discussed in some more detail in Chapter 3, with focus on Kalman filtering and complementary filtering. The ability of fusing information together to form estimates of relevant signals is a prerequisite for the results presented in Chapters 7 and 8. In Chapter 7 the joint angular positions of a flexible nonlinear two-link robot model are estimated using an EKF based on measurements of the motor angular position and the tool acceleration. In Chapter 8 the tool position of a parallel robot is estimated experimentally by using measured motor angular positions and tool acceleration in a complementary filter and a Kalman filter, respectively. These estimates are thereafter used in ILC algorithms to improve the robot performance.

Example 6.5: Case 2B

Based on a nominal model of the state-space description (6.2) with input $\bar{u}_k(t)$, an observer can be formed as

$$\hat{x}_k(t+1) = A\hat{x}_k(t) + B\bar{u}_k(t) + K\Big(y_k(t) - C\hat{x}_k(t)\Big) \tag{6.10a}$$

$$\hat{z}_k(t) = M\hat{x}_k(t) \tag{6.10b}$$

where $y_k(t)$ can be a vector consisting of several measurements. Using transfer operators the estimate of the controlled variable can be expressed by

$$\begin{aligned}\hat{z}_k(t) &= M\Big(qI - (A - KC)\Big)^{-1}B\bar{u}_k(t) + M\Big(qI - (A - KC)\Big)^{-1}Ky_k(t) \\ &= F_{\bar{u}}(q)\bar{u}_k(t) + F_{\bar{y}}(q)y_k(t)\end{aligned} \tag{6.11}$$

From the closed-loop system in Figure 6.3, the filters in (6.6) can be identified as

$$\begin{aligned}\hat{z}_k(t) &= F_{\bar{u}}(q)F(q)r(t) + F_{\bar{u}}(q)F(q)u_k(t) + \Big(F_{\bar{y}}(q) - F_{\bar{u}}(q)F(q)\Big)y_k(t) \\ &= F_r(q)r(t) + F_u(q)u_k(t) + F_y(q)y_k(t)\end{aligned} \tag{6.12}$$

To summarise, from Examples 6.3 to 6.5 it can be seen that the representation (6.6) covers different ways of generating the estimate $\hat{z}_k(t)$; by neglecting the dynamics, by using a nominal model of the relation between $y_k(t)$ and $z_k(t)$, and by using observers. It is therefore natural to use the relation (6.6) in the ILC algorithm, as is seen in the forthcoming section.

6.4 ILC algorithm

In this chapter the structure (4.12) for an ILC algorithm in transfer-operator form is considered, where the update equation is given by

$$u_{k+1}(t) = Q(q)\Big(u_k(t) + L(q)\epsilon_k(t)\Big) \tag{6.13}$$

The error used in the ILC algorithm,

$$\epsilon_k(t) = r(t) - \hat{z}_k(t)$$

is the difference between the reference $r(t)$ and the estimate $\hat{z}_k(t)$ from (6.6) of the controlled variable $z_k(t)$. The controlled variable $z_k(t)$ should follow the reference $r(t)$. Then, the following error

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

is used to evaluate the performance of the system. Using the matrices $\boldsymbol{L}$ and $\boldsymbol{Q}$, it results in the matrix form of the ILC update equation and error used in the ILC algorithm as follows

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}(\boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{\epsilon}_k) \tag{6.14a}$$

$$\boldsymbol{\epsilon}_k = \boldsymbol{r} - \hat{\boldsymbol{z}}_k \tag{6.14b}$$

while the system performance (6.4) is evaluated by

$$\boldsymbol{e}_k = \boldsymbol{r} - \boldsymbol{z}_k \tag{6.15}$$

6.5 Analysis

To support the analysis and make it easier to follow the derivations in this section, the main equations (6.5), (6.7), and (6.14) to (6.15) from previous sections are summarised here. The system given by

$$\boldsymbol{y}_k = \boldsymbol{T}^0_{ry}\boldsymbol{r} + \boldsymbol{T}^0_{uy}\boldsymbol{u}_k \tag{6.16a}$$

$$\boldsymbol{z}_k = \boldsymbol{T}^0_{rz}\boldsymbol{r} + \boldsymbol{T}^0_{uz}\boldsymbol{u}_k \tag{6.16b}$$

is controlled by using the ILC input signal $\boldsymbol{u}_k$, updated by

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}(\boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{\epsilon}_k) \tag{6.17a}$$

$$\boldsymbol{\epsilon}_k = \boldsymbol{r} - \hat{\boldsymbol{z}}_k \tag{6.17b}$$

using the estimate

$$\hat{\boldsymbol{z}}_k = \boldsymbol{F}_r\boldsymbol{r} + \boldsymbol{F}_u\boldsymbol{u}_k + \boldsymbol{F}_y\boldsymbol{y}_k \tag{6.18}$$

The system performance is evaluated by

$$\boldsymbol{e}_k = \boldsymbol{r} - \boldsymbol{z}_k \tag{6.19}$$

From the relations (6.16) to (6.19) the ILC system equation is now derived, which will be used in the stability analysis.

Lemma 6.1 (ILC system equation). *Consider the system in* (6.16) *using the ILC update equation* (6.17) *and the estimate* $\hat{\boldsymbol{z}}_k$ *from* (6.18)*. Then the ILC system equation is given by*

$$\boldsymbol{u}_{k+1} = \boldsymbol{H}\boldsymbol{u}_k + \boldsymbol{H}_r\boldsymbol{r} \tag{6.20}$$

where

$$\boldsymbol{H} = \boldsymbol{Q}\Big(I - \boldsymbol{L}(\boldsymbol{F}_u + \boldsymbol{F}_y\boldsymbol{T}_{uy}^0)\Big) \tag{6.21a}$$

$$\boldsymbol{H}_r = \boldsymbol{Q}\boldsymbol{L}\Big(I - (\boldsymbol{F}_r + \boldsymbol{F}_y\boldsymbol{T}_{ry}^0)\Big) \tag{6.21b}$$

Proof: Using (6.16), the error $\boldsymbol{\epsilon}_k$ in (6.17b) can be written

$$\begin{aligned}\boldsymbol{\epsilon}_k &= \boldsymbol{r} - \boldsymbol{F}_r\boldsymbol{r} - \boldsymbol{F}_u\boldsymbol{u}_k - \boldsymbol{F}_y(\boldsymbol{T}_{ry}^0\boldsymbol{r} + \boldsymbol{T}_{uy}^0\boldsymbol{u}_k)\\ &= \Big(I - (\boldsymbol{F}_r + \boldsymbol{F}_y\boldsymbol{T}_{ry}^0)\Big)\boldsymbol{r} - (\boldsymbol{F}_u + \boldsymbol{F}_y\boldsymbol{T}_{uy}^0)\boldsymbol{u}_k\end{aligned}$$

Inserting the error $\boldsymbol{\epsilon}_k$ into the ILC update equation (6.17a) then gives

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}\Big(I - \boldsymbol{L}(\boldsymbol{F}_u + \boldsymbol{F}_y\boldsymbol{T}_{uy}^0)\Big)\boldsymbol{u}_k + \boldsymbol{Q}\boldsymbol{L}\Big(I - (\boldsymbol{F}_r + \boldsymbol{F}_y\boldsymbol{T}_{ry}^0)\Big)\boldsymbol{r}$$

and the result follows. □

Stability of the ILC system in Lemma 6.1 can be expressed based on the results in Chapter 4.

Theorem 6.1 (Stability). *Consider the ILC system equation* (6.20) *in Lemma 6.1,*

$$\boldsymbol{u}_{k+1} = \boldsymbol{H}\boldsymbol{u}_k + \boldsymbol{H}_r\boldsymbol{r}$$

This system is stable if and only if $\rho(\boldsymbol{H}) < 1$*, where* $\rho(\cdot)$ *is the spectral radius of the matrix.*

Proof: Follows from Theorem 4.5 with the ILC system (6.20). □

The system performance $\boldsymbol{e}_k$ when the number of iterations goes to infinity is given by the following lemma.

Lemma 6.2 (Asymptotic behaviour). *Under the assumption that the input* $\boldsymbol{u}_k$ *in the ILC system equation* (6.20) *from Lemma 6.1 converges to a bounded signal as* $k \to \infty$*, the limit is given by*

$$\boldsymbol{u}_\infty = \Big(I - \boldsymbol{Q}\Big(I - \boldsymbol{L}(\boldsymbol{F}_u + \boldsymbol{F}_y\boldsymbol{T}_{uy}^0)\Big)\Big)^{-1}\boldsymbol{Q}\boldsymbol{L}\Big(I - (\boldsymbol{F}_r + \boldsymbol{F}_y\boldsymbol{T}_{ry}^0)\Big)\boldsymbol{r} \tag{6.22}$$

The corresponding asymptotic error $\boldsymbol{r} - \boldsymbol{z}_\infty$ *becomes*

$$\boldsymbol{e}_\infty = \Big(I - \boldsymbol{T}_{rz}^0 - \boldsymbol{T}_{uz}^0\Big(I - \boldsymbol{Q}(I - \boldsymbol{L}(\boldsymbol{F}_u + \boldsymbol{F}_y\boldsymbol{T}_{uy}^0))\Big)^{-1}\boldsymbol{Q}\boldsymbol{L}\Big(I - (\boldsymbol{F}_r + \boldsymbol{F}_y\boldsymbol{T}_{ry}^0)\Big)\Big)\boldsymbol{r} \tag{6.23}$$

Proof: Let $k \to \infty$ and solve the ILC system equation (6.20) from Lemma 6.1,

$$\boldsymbol{u}_\infty = \boldsymbol{Q}\Big(I - \boldsymbol{L}(\boldsymbol{F}_u + \boldsymbol{F}_y \boldsymbol{T}_{uy}^0)\Big)\boldsymbol{u}_\infty + \boldsymbol{Q}\boldsymbol{L}\Big(I - (\boldsymbol{F}_r + \boldsymbol{F}_y \boldsymbol{T}_{ry}^0)\Big)\boldsymbol{r}$$

for $\boldsymbol{u}_\infty$. Insert $\boldsymbol{u}_\infty$ in the relation for the error of the controlled variable, using (6.16b),

$$\boldsymbol{e}_\infty = \boldsymbol{r} - \boldsymbol{z}_\infty = \boldsymbol{r} - (\boldsymbol{T}_{rz}^0 \boldsymbol{r} + \boldsymbol{T}_{uz}^0 \boldsymbol{u}_\infty)$$

to derive the asymptotic error. □

From the expression (6.23) it can be seen that the asymptotic error $\boldsymbol{e}_\infty$ depends on:

- The feedback controller via the possibly closed-loop system matrices $\boldsymbol{T}_{ry}^0$, $\boldsymbol{T}_{uy}^0$, $\boldsymbol{T}_{rz}^0$ and $\boldsymbol{T}_{uz}^0$, as seen in Example 6.2.
- The method to estimate $\hat{\boldsymbol{z}}$ via the representation (6.7) and the matrices $\boldsymbol{F}_r$, $\boldsymbol{F}_u$ and $\boldsymbol{F}_y$, see Examples 6.3 to 6.5.
- The magnitude of the model errors via the matrices $\boldsymbol{F}_r$, $\boldsymbol{F}_u$ and $\boldsymbol{F}_y$, see Examples 6.3 to 6.5.
- The choice of ILC algorithm, via the matrices $\boldsymbol{Q}$ and $\boldsymbol{L}$.

The influence of the individual elements on the asymptotic error is not obvious from the expression (6.23). The next section discusses various aspects of how the choice of estimation filters and the magnitude of model errors affect the resulting value of $\boldsymbol{e}_\infty$, both by studying the expression (6.23) in detail for four different cases and by a numerical example.

From a design perspective the following result has an important impact.

Theorem 6.2 (Monotone convergence). *Consider the ILC system* (6.20). *If the maximum singular value satisfies* $\bar{\sigma}(\boldsymbol{H}) \leq \lambda < 1$, *then the system is stable, and*

$$\|\boldsymbol{u}_\infty - \boldsymbol{u}_k\|_2 \leq \lambda^k \|\boldsymbol{u}_\infty - \boldsymbol{u}_0\|_2$$

with $\boldsymbol{u}_\infty$ *from* (6.22) *in Lemma 6.2.*

Proof: Follows from Theorem 4.6 with the ILC system equation (6.20) and with the asymptotic ILC input $\boldsymbol{u}_\infty$ from Lemma 6.2. □

Monotone convergence of $\boldsymbol{u}_k$ is a good design criterion in applications. From Theorem 6.2 it can be seen that the smaller the maximum singular value of the matrix $\boldsymbol{H}$, the faster the convergence. It is also necessary to check $\boldsymbol{e}_0$ and $\boldsymbol{e}_\infty$ such that the error is reduced as desired. A design criterion therefore should include minimising the maximum singular value $\bar{\sigma}(\boldsymbol{H})$ as well as the asymptotic error $\boldsymbol{e}_\infty$.

6.6 Illustration of the results

The asymptotic error $\boldsymbol{e}_\infty$ of the controlled variable from Lemma 6.2 is discussed in this section. First, in Sections 6.6.1 to 6.6.4 it is assumed that $\boldsymbol{L}$ can be chosen such that $\boldsymbol{u}_k$ converges to a bounded signal when $\boldsymbol{Q} = I$ in the ILC algorithm (6.17). Although this idealised choice of $\boldsymbol{Q}$ is seldom possible in practice, it is fruitful in the analysis of $\boldsymbol{e}_\infty$, since by this choice only the influence of estimation matrices $\boldsymbol{F}_r$, $\boldsymbol{F}_u$ and $\boldsymbol{F}_y$ is studied and the result is independent of the choice of ILC matrices $\boldsymbol{Q}$ and $\boldsymbol{L}$. In addition to this idealised discussion, a numerical example with a flexible two-mass system is given in Section 6.6.5, with a general matrix $\boldsymbol{Q}$ (different from I). The transient and asymptotic behaviour of the two-mass system is discussed when an ILC algorithm is applied to the system, using different estimates $\hat{\boldsymbol{z}}_k$.

To support the intuition of the results in Sections 6.6.1 to 6.6.4, the relations are illustrated for a system where it is possible to write an explicit stable relationship (6.8) between the measured variable $y_k(t)$ and controlled variable $z_k(t)$,

$$z_k(t) = T_{yz}^0(q) y_k(t)$$

See Section 6.6.5 for further details, where such a system is exemplified. From the pulse-response coefficients of the system $T_{yz}^0(q)$, the matrix $\boldsymbol{T}_{yz}^0$ can be defined similarly as in (4.3), resulting in

$$\boldsymbol{z}_k = \boldsymbol{T}_{yz}^0 \boldsymbol{y}_k \tag{6.24}$$

The ILC algorithm (6.17) is in this section applied to the system (6.16) for the following cases:

1. The ILC input is updated using $\boldsymbol{\epsilon}_k = \boldsymbol{r} - \boldsymbol{y}_k$, that is, the dynamics between $\boldsymbol{y}_k$ and $\boldsymbol{z}_k$ is neglected.

2A. The ILC input is updated using $\boldsymbol{\epsilon}_k = \boldsymbol{r} - \boldsymbol{T}_{yz}\boldsymbol{y}_k$, that is, $\hat{\boldsymbol{z}}_k$ is estimated from a nominal model of the relation (6.24).

2B. The ILC input is updated using $\boldsymbol{\epsilon}_k = \boldsymbol{r} - \hat{\boldsymbol{z}}_k$, where $\hat{\boldsymbol{z}}_k = \boldsymbol{F}_r\boldsymbol{r} + \boldsymbol{F}_u\boldsymbol{u}_k + \boldsymbol{F}_y\boldsymbol{y}_k$. The filters $\boldsymbol{F}_r$, $\boldsymbol{F}_u$ and $\boldsymbol{F}_y$ are obtained from the observer (6.10).

3. As a comparison, in order to show what ideally can be achieved, it is assumed that the controlled variable $\boldsymbol{z}_k$ can be measured and that the ILC input is updated using $\boldsymbol{\epsilon}_k = \boldsymbol{r} - \boldsymbol{z}_k$.

The choices of filters $F_r(q)$, $F_u(q)$ and $F_y(q)$ in (6.6) to implement the cases 1, 2A and 2B were previously discussed in Examples 6.3 to 6.5 in Section 6.3. In Table 6.1 the resulting ILC system equation (6.20) and the asymptotic error $e_\infty(t)$ of the controlled variable $z_k(t)$, see Lemma 6.2, are summarised for the cases above. These relations are next discussed in more detail.

6.6.1 Case 1 — ILC using measured variable

For case 1 the dynamics between the measured variable $\boldsymbol{y}_k$ and the controlled variable $\boldsymbol{z}_k$ is neglected, giving the estimate $\hat{\boldsymbol{z}}_k = \boldsymbol{y}_k$, see Example 6.3. The actual

Table 6.1: *Summary of the description of a system using ILC and performance when the ILC algorithm uses various errors, cases 1 to 3, for the ideal choice $Q(q) = 1$.*

Case	ϵ_k	$\boldsymbol{u}_{k+1} = \boldsymbol{H}\boldsymbol{u}_k + \boldsymbol{H}_r\boldsymbol{r}$	$e_\infty(t) = r(t) - z_\infty(t), \quad Q(q) = 1$
1	$\boldsymbol{r} - \boldsymbol{y}_k$	$\boldsymbol{H} = \boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_{uy}^0)$ $\boldsymbol{H}_r = \boldsymbol{Q}\boldsymbol{L}(I - \boldsymbol{T}_{ry}^0)$	$\Big(1 - T_{yz}^0(q)\Big)r(t)$
2A	$\boldsymbol{r} - \boldsymbol{T}_{yz}\boldsymbol{y}_k$	$\boldsymbol{H} = \boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_{yz}\boldsymbol{T}_{uy}^0)$ $\boldsymbol{H}_r = \boldsymbol{Q}\boldsymbol{L}(I - \boldsymbol{T}_{yz}\boldsymbol{T}_{ry}^0)$	$\Big(1 - T_{yz}^0(q)T_{yz}^{-1}(q)\Big)r(t)$
2B	$\boldsymbol{r} - (\boldsymbol{F}_r\boldsymbol{r} + \boldsymbol{F}_u\boldsymbol{u}_k + \boldsymbol{F}_y\boldsymbol{y}_k)$	$\boldsymbol{H} = \boldsymbol{Q}\Big(I - \boldsymbol{L}(\boldsymbol{F}_u + \boldsymbol{F}_y\boldsymbol{T}_{uy}^0)\Big)$ $\boldsymbol{H}_r = \boldsymbol{Q}\boldsymbol{L}\Big(I - (\boldsymbol{F}_r + \boldsymbol{F}_y\boldsymbol{T}_{ry}^0)\Big)$	$\left(1 - T_{yz}^0(q)\dfrac{T_{ry}^0(q)F_u(q) + \Big(1 - F_r(q)\Big)T_{uy}^0(q)}{F_u(q) + F_y(q)T_{uy}^0(q)}\right)r(t)$
3	$\boldsymbol{r} - \boldsymbol{z}_k$	$\boldsymbol{H} = \boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_{yz}^0\boldsymbol{T}_{uy}^0)$ $\boldsymbol{H}_r = \boldsymbol{Q}\boldsymbol{L}(I - \boldsymbol{T}_{yz}^0\boldsymbol{T}_{ry}^0)$	0

relation between $\boldsymbol{y}_k$ and $\boldsymbol{z}_k$ is given by (6.16). The assumption in Section 6.6 that $\boldsymbol{L}$ can be chosen such that $\boldsymbol{Q} = I$ implies $\boldsymbol{\epsilon}_\infty = 0$. The asymptotic error $\boldsymbol{e}_\infty$ from Lemma 6.2 is then given by

$$\boldsymbol{e}_\infty = \Big(I - \boldsymbol{T}_{rz}^0 - \boldsymbol{T}_{uz}^0(\boldsymbol{L}\boldsymbol{T}_{uy}^0)^{-1}\boldsymbol{L}(I - \boldsymbol{T}_{ry}^0)\Big)\boldsymbol{r} \tag{6.25}$$

Using the assumption that the system can be described by an explicit relation between the measured and controlled variable, see (6.24), it results in the relations

$$\boldsymbol{T}_{rz}^0 = \boldsymbol{T}_{yz}^0\boldsymbol{T}_{ry}^0 \tag{6.26a}$$

$$\boldsymbol{T}_{uz}^0 = \boldsymbol{T}_{yz}^0\boldsymbol{T}_{uy}^0 \tag{6.26b}$$

For the SISO system described by (6.1), the relation (6.25) can under the assumption (6.26) be rewritten in transfer-operator form as

$$e_\infty(t) = \Big(1 - T_{yz}^0(q)\Big)r(t)$$

In general, $T_{yz}^0(q) \neq 1$ results in an asymptotic error $e_\infty(t) \neq 0$ of the controlled variable, as expected.

For the situation with a general matrix $\boldsymbol{Q}$, it can be seen that the expression (6.23) for the asymptotic error $\boldsymbol{e}_\infty$ is equal to (4.32) when $\boldsymbol{T}_{yz} = \boldsymbol{T}_{yz}^0 = I$.

6.6.2 Case 2A — ILC using estimate from model of direct relation

Under the assumption that the relation (6.24) between $\boldsymbol{y}_k$ and $\boldsymbol{z}_k$ holds, a straightforward way to estimate $\boldsymbol{z}_k$ is by using the nominal dynamics, giving $\hat{\boldsymbol{z}}_k = \boldsymbol{T}_{yz}\boldsymbol{y}_k$. The asymptotic error $\boldsymbol{e}_\infty$ from Lemma 6.2, for $\boldsymbol{Q} = I$ and using the approximation (6.26), results in

$$\boldsymbol{e}_\infty = \Big(I - \boldsymbol{T}_{yz}^0\Big(\boldsymbol{T}_{ry}^0 + \boldsymbol{T}_{uy}^0(\boldsymbol{L}\boldsymbol{T}_{yz}\boldsymbol{T}_{uy}^0)^{-1}\boldsymbol{L}(I - \boldsymbol{T}_{yz}\boldsymbol{T}_{ry}^0)\Big)\Big)\boldsymbol{r} \tag{6.27}$$

For the SISO system given by (6.1), the relation (6.27) for the asymptotic error can be rewritten as

$$e_\infty(t) = \Big(1 - T_{yz}^0(q)T_{yz}^{-1}(q)\Big)r(t)$$

using transfer operators. From this expression it can be seen that when the nominal model $T_{yz}(q)$ is equal to the true system $T_{yz}^0(q)$, the error $e_\infty(t) = 0$. An obvious question is how close $T_{yz}(q)$ needs to be to $T_{yz}^0(q)$ to get better performance using the case 2A compared to case 1. This will depend on the properties of the particular application.

Study the expression (6.23) of the asymptotic error for case 2A with a general matrix $\boldsymbol{Q}$, under the assumption of $\boldsymbol{T}_{yz} = \boldsymbol{T}_{yz}^0$, that is, $\hat{\boldsymbol{z}}_k = \boldsymbol{z}_k$,

$$\boldsymbol{e}_\infty = \Big(I - \boldsymbol{T}_{rz}^0 - \boldsymbol{T}_{uz}^0\Big(I - \boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_{yz}^0\boldsymbol{T}_{uy}^0)\Big)^{-1}\boldsymbol{Q}\boldsymbol{L}\Big(I - \boldsymbol{T}_{yz}^0\boldsymbol{T}_{ry}^0\Big)\Big)\boldsymbol{r} \tag{6.28}$$

Using (6.26), the expression (6.28) can be rewritten to (4.32), with the error $\boldsymbol{e}_k = \boldsymbol{r} - \boldsymbol{z}_k$ used in the ILC algorithm (4.13).

6.6.3 Case 2B— ILC using estimate from linear observer

The estimate $\hat{\boldsymbol{z}}_k$ is now given by (6.18),

$$\hat{\boldsymbol{z}}_k = \boldsymbol{F}_r \boldsymbol{r} + \boldsymbol{F}_u \boldsymbol{u}_k + \boldsymbol{F}_y \boldsymbol{y}_k$$

Using $\boldsymbol{Q} = I$, the asymptotic error $\boldsymbol{e}_\infty$ from Lemma 6.2 is given by

$$\boldsymbol{e}_\infty = \Big(I - \boldsymbol{T}_{rz}^0 - \boldsymbol{T}_{uz}^0 \Big(\boldsymbol{L}(\boldsymbol{F}_u + \boldsymbol{F}_y \boldsymbol{T}_{uy}^0) \Big)^{-1} \boldsymbol{L} \Big(I - (\boldsymbol{F}_r + \boldsymbol{F}_y \boldsymbol{T}_{ry}^0) \Big) \Big) \boldsymbol{r}$$

Under the assumption that relation (6.26) holds, the asymptotic error can for the SISO system (6.1) be rewritten in transfer-operator form as

$$e_\infty(t) = \left(1 - T_{yz}^0(q) \frac{T_{ry}^0(q) F_u(q) + (1 - F_r(q)) T_{uy}^0(q)}{F_u(q) + F_y(q) T_{uy}^0(q)} \right) r(t) \tag{6.29}$$

The filters $F_r(q)$, $F_u(q)$ and $F_y(q)$ are given from the observer (6.10) in Example 6.5, and the closed-loop system is from Example 6.2 given by

$$y_k(t) = \underbrace{\frac{F(q) T_{\bar{u}y}^0(q)}{1 + F(q) T_{\bar{u}y}^0(q)}}_{T_{ry}^0(q)} r(t) + \underbrace{\frac{F(q) T_{\bar{u}y}^0(q)}{1 + F(q) T_{\bar{u}y}^0(q)}}_{T_{uy}^0(q)} u_k(t)$$

The asymptotic error $e_\infty(t)$ in (6.29) can then be rewritten as

$$e_\infty(t) = \left(1 - T_{yz}^0(q) \frac{T_{\bar{u}y}^0(q)}{F_{\bar{u}}(q) + F_{\bar{y}}(q) T_{\bar{u}y}^0(q)} \right) r(t) \tag{6.30}$$

The transfer operators in (6.30) are given by

$$T_{\bar{u}y}^0(q) = C^0 (qI - A^0)^{-1} B^0$$
$$T_{\bar{u}z}^0(q) = M^0 (qI - A^0)^{-1} B^0$$

from (6.3) for the system with input $\bar{u}_k(t)$, and

$$F_{\bar{u}}(q) = M \Big(qI - (A - KC) \Big)^{-1} B$$
$$F_{\bar{y}}(q) = M \Big(qI - (A - KC) \Big)^{-1} K$$

from Example 6.5. It is seen that the expression for $e_\infty(t)$ in (6.30) is independent of the controller $F(q)$, and only depends on the magnitude of the model errors.

For an observer (6.10) based on the true system, the denominator of (6.30) is

$$\begin{aligned} F_{\bar{u}}(q) + F_{\bar{y}}(q) T_{\bar{u}y}^0(q) &= M^0 (qI - A^0)^{-1} B^0 \\ &= T_{\bar{u}z}^0(q) = T_{yz}^0(q) T_{\bar{u}y}^0(q) \end{aligned} \tag{6.31}$$

where the last equality can be realised from (6.8). From this expression it is seen that the asymptotic error in (6.30) is $e_\infty(t) = 0$, which is expected when $\hat{z}_k(t)$ results from an observer based on the true system.

Similarly to the cases above, study now a situation with a general transfer operator $Q(q)$ and where the nominal model is equal to the true system. The asymptotic error $e_\infty(t)$ is, with the filters $F_r(q)$, $F_u(q)$ and $F_y(q)$ from the observer (6.10), given by

$$e_\infty(t) = 1 - T_{rz}^0(q) - T_{uz}^0(q) \frac{Q(q)L(q)\Big(1 - \Big(F_{\bar{u}}(q)F(q) + (F_{\bar{y}}(q) - F_{\bar{u}}(q)F(q))T_{r\bar{y}}^0(q)\Big)\Big)}{1 - Q(q)\Big(1 - L(q)\Big(F_{\bar{u}}(q)F(q) + (F_{\bar{y}}(q) - F_{\bar{u}}(q)F(q))T_{u\bar{y}}^0(q)\Big)\Big)} \tag{6.32}$$

Study now the term $F_{\bar{u}}(q)F(q) + \Big(F_{\bar{y}}(q) - F_{\bar{u}}(q)F(q)\Big)T_{r\bar{y}}^0(q)$ in (6.32). It can be rewritten as

$$F_{\bar{u}}(q)F(q) + \Big(F_{\bar{y}}(q) - F_{\bar{u}}(q)F(q)\Big)T_{r\bar{y}}^0(q) = \frac{F(q)\Big(F_{\bar{u}}(q) + F_{\bar{y}}(q)T_{\bar{u}\bar{y}}^0(q)\Big)}{1 + F(q)T_{\bar{u}\bar{y}}^0(q)}$$

from the closed-loop system in Example 6.2,

$$T_{r\bar{y}}^0(q) = T_{u\bar{y}}^0(q) = \frac{F(q)T_{\bar{u}\bar{y}}^0(q)}{1 + F(q)T_{\bar{u}\bar{y}}^0(q)} \tag{6.33}$$

The same result is given for $F_{\bar{u}}(q)F(q) + \Big(F_{\bar{y}}(q) - F_{\bar{u}}(q)F(q)\Big)T_{u\bar{y}}^0(q)$ in (6.32). Then, from (6.31), the relation (6.33) can be written as

$$\frac{F(q)\Big(F_{\bar{u}}(q) + F_{\bar{y}}(q)T_{\bar{u}\bar{y}}(q)\Big)}{1 + F(q)T_{\bar{u}\bar{y}}(q)} = \frac{F(q)T_{\bar{y}z}^0(q)T_{\bar{u}\bar{y}}^0(q)}{1 + F(q)T_{\bar{u}\bar{y}}(q)} = T_{\bar{y}z}^0(q)T_{r\bar{y}}^0(q) = T_{rz}^0(q) = T_{uz}^0(q)$$

which is inserted in the expression for the asymptotic error (6.32). It can be seen that (6.32) is equal to (4.32), when $\hat{z}_k(t)$ from an observer based on the true system is used in the ILC algorithm (4.13).

6.6.4 Case 3 — ILC using controlled variable

The last case is used to show what ideally can be achieved when the ILC algorithm (6.17) uses the error $\boldsymbol{\epsilon}_k = \boldsymbol{r} - \boldsymbol{z}_k$. For a general matrix $\boldsymbol{Q}$, the asymptotic error $\boldsymbol{e}_\infty$ of the controlled variable is from Lemma 6.2 given by

$$\boldsymbol{e}_\infty = \Big(I - \boldsymbol{T}_{rz}^0 - \boldsymbol{T}_{uz}^0\Big(I - \boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_{uz}^0)\Big)^{-1}\boldsymbol{Q}\boldsymbol{L}(I - \boldsymbol{T}_{rz}^0)\Big)\boldsymbol{r}$$

As expected, this relation is equal to the asymptotic error (4.32), with the error $\boldsymbol{e}_k = \boldsymbol{r} - \boldsymbol{z}_k$ used in the ILC algorithm (4.13). This expression can for a SISO system described in transfer-operator form (6.1) be rewritten as

$$e_\infty(t) = \frac{\Big(1 - Q(q)\Big)\Big(1 - T_{rz}^0(q)\Big)}{1 - Q(q)\Big(1 - L(q)T_{uz}^0(q)\Big)}r(t)$$

Clearly $e_\infty(t)$ is zero when the ILC algorithm converges for the choice $Q(q) = 1$, achieving perfect tracking of the controlled variable $z_k(t)$.

6.6.5 Numerical example

In this section the cases 1 to 3 are illustrated and compared when the ILC algorithm (6.17) is applied to a two-mass system consisting of two masses connected by a spring-damper pair[2], as illustrated in Figure 2.4. The deflection of the system is described by the angular position of the first mass, referred to as motor angular position $q_m(t)$, and the angular position of the second mass, called joint angular position $q_a(t)$. The input torque applied to the first mass is denoted $\bar{u}(t)$ to agree with the notation introduced in Example 6.2. The model equations are given in detail in (2.11), with the model parameter values in Table 6.2.

A state-space description of the two-mass system can be derived using the following state vector

$$x(t) = \begin{pmatrix} q_a(t) & \dot{q}_a(t) & q_m(t) & \dot{q}_m(t) \end{pmatrix}^T$$

The measured variable is the motor angular position $q_m(t)$, while the controlled variable is the joint angular position $q_{a,k}(t)$. From the system dynamics in (2.11) it is straightforward to compute the transfer operators

$$q_m(t) = T^0_{\bar{u}y}(p)\bar{u}(t) \tag{6.34a}$$

$$q_a(t) = T^0_{yz}(p)q_m(t) \tag{6.34b}$$

where p is the derivative operator. In Figure 6.5 a block diagram with a system given by the structure (6.34) is illustrated. This can be compared to the closed-loop system in Figure 6.3. It is not possible to find an exact discrete-time representation of the relation between $q_m(t)$ and $q_a(t)$ in (6.34b) since $q_m(t)$ is an internal continuous variable in the two-mass system described by (2.11). When the sampling interval is short it is however possible to find a good approximation, using for example the Tustin derivative approximation.

The system is stabilised by a controller $F(q)$, and an ILC input signal $u_k(t)$ is added to the reference $r(t)$ at iteration k. Moderate requirements for the feedback can be chosen, since the desired servo performance is planned to be achieved by the ILC algorithm. The two-mass system is controlled by using a discrete-time PD-controller including a low-pass filter obtained by manual tuning, resulting in

$$F(q) = K_1 + \frac{K_2 q - K_3}{q - K_4}$$

The controller parameters are given in Table 6.3. The closed-loop system and ILC algorithm is simulated using SIMULINK with a sampling interval of $T_s = 0.01$ s.

A model error is introduced in the system by reducing the spring coefficient by 50 % from the nominal value. This means that the true system has a less rigid spring than expected. The joint angular position $q_a(t)$ is used as an evaluation variable that should follow the desired reference $r_a(t)$, which is chosen as a smooth step for all cases. This is illustrated in Figure 6.7.

[2] See Section 2.3.3 for the details.

Table 6.2: *Model parameter values.*

$\eta = 0.2$	$M_m = 0.0021$	$M_a = 0.0991$	$k = 8$
$d = 0.0924$	$f_m = 0.0713$	$k_\tau = 0.122$	

Table 6.3: *Controller parameter values.*

$K_1 = 5$	$K_2 = 2$	$K_3 = 2$	$K_4 = 0.905$

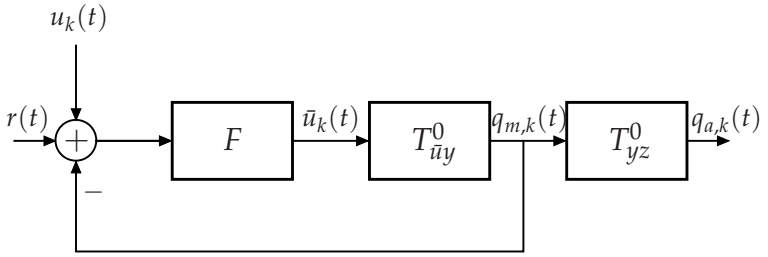


Figure 6.5: *The two-mass system illustrated by $T^0_{\bar{u}y}(q)$ and $T^0_{yz}(q)$; transfer function from motor torque $\bar{u}_k(t)$ to motor angular position $q_{m,k}(t)$ and from motor angular position $q_{m,k}(t)$ to joint angular position $q_{a,k}(t)$, respectively. The system is controlled by the feedback controller $F(q)$, and the ILC input $u_k(t)$ added to the reference $r(t)$ at iteration k.*

Tuning of observers

An observer of the form (6.10) is used, based on the nominal model of the state-space description (6.2). Two cases $2B_1$ and $2B_2$ will be studied. In case $2B_1$ the motor angular position $q_m(t)$ is measured, while in case $2B_2$ also the joint angular acceleration $\ddot{q}_a(t)$ is measured,

$$y(t) = \begin{cases} q_m(t), & \text{case } 2B_1 \\ \begin{pmatrix} q_m(t) & \ddot{q}_a(t) \end{pmatrix}^T, & \text{case } 2B_2 \end{cases}$$

The controlled variable, the true joint angular position $q_a(t)$, is only used for evaluation purposes in the simulation. The objective is to find an observer gain K giving as small estimation error $q_a(t) - \hat{q}_a(t)$ as possible, that is, a robust estimation problem. The observer gain K is here given from a stationary discrete-time Kalman filter with the covariance matrices, R_v for the process noise and R_w for the measurement noise, as design variables. The following optimisation problem illustrates the aim to find the best tuning when the model is subject to model

errors

$$\begin{aligned} \min_{a,b,c} \quad & \|\boldsymbol{q}_a - \hat{\boldsymbol{q}}_a\|_2 \\ \text{subject to} \quad & R_v = \operatorname{diag}\begin{pmatrix} 0 & 10^a & 0 & 10^b \end{pmatrix} \\ & R_w = \begin{cases} 1, & \text{case } 2\text{B}_1 \\ \operatorname{diag}\begin{pmatrix} 1 & 10^c \end{pmatrix}, & \text{case } 2\text{B}_2 \end{cases} \end{aligned} \tag{6.35}$$

where the optimisation problem is solved approximately by gridding over the optimisation parameters and by simulation of the true system. Large elements in the observer gain K are possible due to ideal sensors with no offset or noise. The covariance matrices are here determined for the reference trajectory shown in Figure 6.7 and for a specific model error, and the result is not possible to generalise to arbitrary trajectories and model errors. Hence, the robustness of the observer is not evaluated. Although the observer is tuned for a special operating point with ideal measurements, it can be used for illustrative purposes to discuss and exemplify the cases 1 to 3.

Tuning of ILC algorithms

For each individual case 1 to 3 an ILC algorithm (6.13) is designed. One possible choice, which is used here, is $L(q) = \gamma q^{\delta}$. A low-pass second-order Butterworth filter $\bar{Q}(q)$ is chosen, with cutoff frequency f_c above the bandwidth and resonance frequency of the closed-loop system. This enables correction also of errors associated with the resonance frequency. The filter $Q(q)$ is then given by filtering the signal forwards and backwards through the filter $\bar{Q}(q)$ to give zero-phase characteristics. The corresponding matrices $\boldsymbol{Q}$ and $\boldsymbol{L}$ are found by forming the Toeplitz matrices similarly as in (4.3) from the pulse response coefficients[3] of the transfer operators $Q(q)$ and $L(q)$.

The ILC design parameters f_c (resolution 1 Hz), δ (integer) and γ (resolution 0.1) are heuristically tuned for the cases 1 to 3, based on the reduction of the 2-norm of the error $\boldsymbol{\epsilon}_k$ used in the ILC algorithm. The ILC algorithms are tuned such that $\|\boldsymbol{\epsilon}_\infty\|_2$ is approximately 3 % of $\|\boldsymbol{\epsilon}_0\|_2$, that is, when no ILC algorithm is applied (see Table 6.4). The tuning is a trade-off between transient performance and similar level of error reduction for $\boldsymbol{\epsilon}_k$ for the cases. The cutoff frequency f_c is chosen as large as possible to be able to suppress also higher frequencies of the error. For all cases it results in $\bar{\sigma}(\boldsymbol{H}) < 1$ for the true system, giving monotone convergence according to Theorem 6.2. The tuning for the cases results in:

1. ILC design parameters $f_c = 9\,\text{Hz}$, $\delta = 10$, $\gamma = 0.4$, results in $\bar{\sigma}(\boldsymbol{H}) \approx 0.969$.

2A. ILC design parameters $f_c = 9\,\text{Hz}$, $\delta = 27$, $\gamma = 0.3$, results in $\bar{\sigma}(\boldsymbol{H}) \approx 0.990$.

2B$_1$. Joint angular position $\hat{\boldsymbol{q}}_a$ estimated by a Kalman filter from measurements of motor angular position $\boldsymbol{q}_m$. ILC design parameters $f_c = 9\,\text{Hz}$, $\delta = 17$, $\gamma = 1.4$ results in $\bar{\sigma}(\boldsymbol{H}) \approx 0.999$.

[3] The matrix $\boldsymbol{L}$ is formed according to the implementation alternative B and the matrix $\boldsymbol{Q}$ according to alternative I as described in Section 9.5.2.

2B$_2$. Joint angular position $\hat{\boldsymbol{q}}_a$ estimated by a Kalman filter from measurements of motor angular position $\boldsymbol{q}_m$ and joint angular acceleration $\ddot{\boldsymbol{q}}_a$. ILC design parameters $f_c = 7\,\text{Hz}$, $\delta = 27$, $\gamma = 0.5$, results in $\bar{\sigma}(\boldsymbol{H}) \approx 0.982$.

3. ILC design parameters $f_c = 7\,\text{Hz}$, $\delta = 28$, $\gamma = 0.5$, results in $\bar{\sigma}(\boldsymbol{H}) \approx 0.982$.

Evaluation measure

The performance of the two-mass system in the simulations for the different cases is evaluated with respect to the error $\boldsymbol{e}_k = \boldsymbol{r} - \boldsymbol{q}_{a,k}$ of the controlled variable. The reduction of the 2-norm of the error for the joint angular position is calculated as

$$e_{\text{red},k} = 100 \cdot \frac{\|\boldsymbol{e}_k\|_2}{\|\boldsymbol{e}_0\|_2} \quad [\%] \tag{6.36}$$

The reduction of the 2-norm of the error $\boldsymbol{\epsilon}_k = \boldsymbol{r} - \hat{\boldsymbol{q}}_{a,k}$ used in the ILC algorithm (6.14) is given by

$$\epsilon_{\text{red},k} = 100 \cdot \frac{\|\boldsymbol{\epsilon}_k\|_2}{\|\boldsymbol{\epsilon}_0\|_2} \quad [\%] \tag{6.37}$$

The quantities (6.36) and (6.37) measure the relative reduction of the 2-norm of the error, expressed in percentage of the 2-norm of the error when no ILC algorithm is applied ($k = 0$). In this numerical example it is possible to compute these quantities since the true system is known, see (2.11), with the model parameter values in Table 6.2.

Discussion

The evaluation measures (6.36) and (6.37) for the resulting asymptotic errors $\boldsymbol{e}_\infty$ and $\boldsymbol{\epsilon}_\infty$, computed using the true system (2.11) for the cases 1 to 3, are summarised in Table 6.4. It can clearly be seen that the resulting error reduction of the joint angular position $\boldsymbol{q}_{a,k}$ depends on the quality of the estimate. The same conclusion can be drawn from Figure 6.6, where the error reduction (6.36) of the joint angular position $\boldsymbol{q}_{a,k}$ for the cases is illustrated for the first 250 iterations. It can also be noted from Figure 6.6 that we can have non-monotone convergence of the joint angular position error $\boldsymbol{e}_k = \boldsymbol{r} - \boldsymbol{q}_{a,k}$, since the ILC algorithm is tuned with respect to the estimated joint angular position error $\boldsymbol{\epsilon}_k = \boldsymbol{r} - \hat{\boldsymbol{q}}_{a,k}$.

From Figure 6.6 and Table 6.4 it can be seen that case 1 gives a poor reduction of the joint angular position error, with $e_{\text{red},\infty} \approx 110\,\%$. However, the ILC algorithm manages to significantly reduce the error $\boldsymbol{\epsilon}_k$ used in the ILC algorithm, resulting in $\epsilon_{\text{red},\infty} \approx 3.1\,\%$ from Table 6.4. This result is expected for the two-mass model, where there is a significant dynamic relationship between the variables $\boldsymbol{q}_m$ and $\boldsymbol{q}_a$ which is completely neglected in case 1. The same conclusion can also be drawn from the illustration of the resulting joint angular position $\boldsymbol{q}_a$ after 1000 iterations seen in Figure 6.7. Only the time delay is compensated, while the overshoot is even larger compared to the performance without any ILC algorithm applied to the system.

Table 6.4: *Summary of the asymptotic error reduction for the two-mass system when the ILC algorithm uses various estimates of the controlled variable $\boldsymbol{q}_a$; cases 1 to 3.*

Case	$\hat{\boldsymbol{q}}_{a,k}$	$\epsilon_{\mathrm{red},\infty}$	$e_{\mathrm{red},\infty}$
1	$\eta \boldsymbol{q}_{m,k}$	3.09	109.95
2A	$\boldsymbol{T}_{yz} \boldsymbol{q}_{m,k}$	2.97	54.83
$2B_1$	$\boldsymbol{F}_r \boldsymbol{r} + \boldsymbol{F}_u \boldsymbol{u}_k + \boldsymbol{F}_y \boldsymbol{q}_{m,k}$	2.98	44.14
$2B_2$	$\boldsymbol{F}_r \boldsymbol{r} + \boldsymbol{F}_u \boldsymbol{u}_k + \boldsymbol{F}_y \begin{pmatrix} \boldsymbol{q}_{m,k} \\ \ddot{\boldsymbol{q}}_{a,k} \end{pmatrix}$	3.05	3.12
3	$\boldsymbol{q}_a$	3.07	3.07

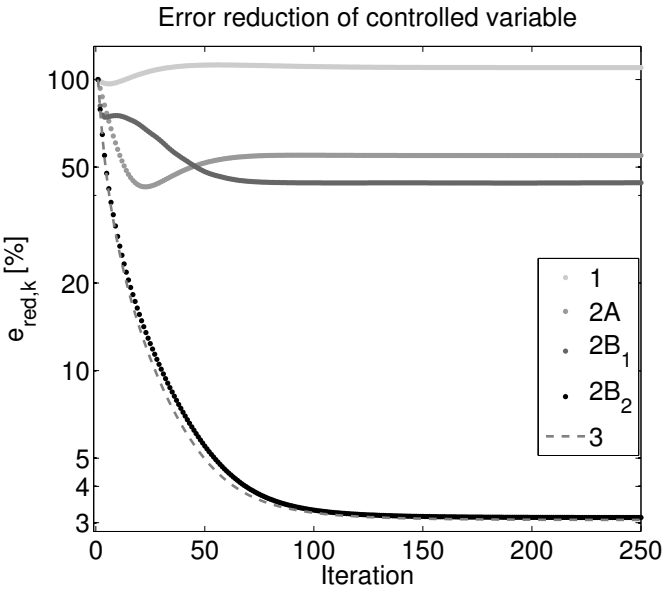


Figure 6.6: *Reduction of $\boldsymbol{e}_k$ given by* (6.36) *for the first 250 iterations for the cases 1 to 3. The error reduction highly depends on the quality of the estimate $\hat{\boldsymbol{q}}_{a,k}$.*

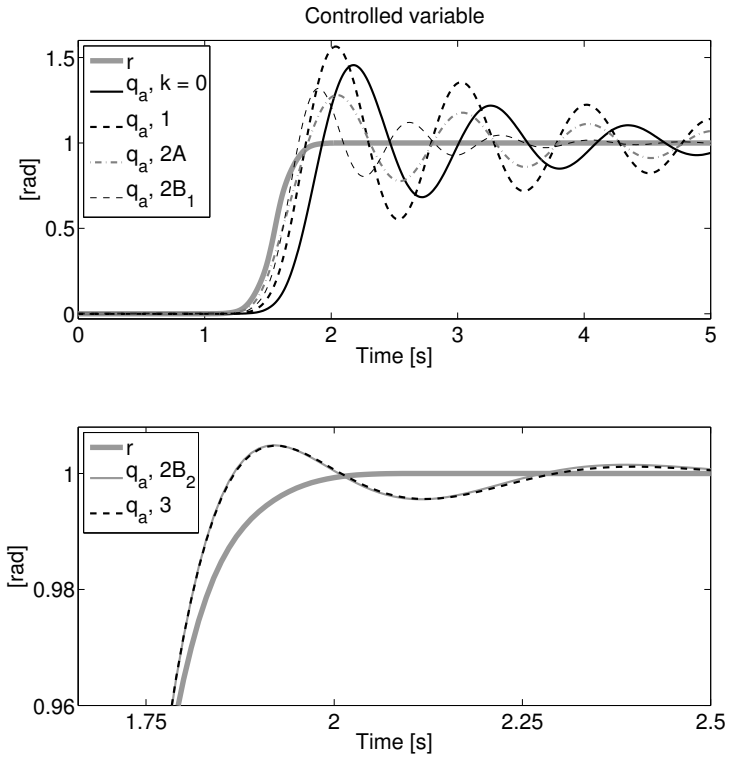


Figure 6.7: *Actual joint angular position $q_a(t)$ after 1000 iterations for the cases 1 to 2B$_1$ compared to the joint angular position when no ILC algorithm is applied (denoted $k = 0$) in the upper figure. The lower figure shows $q_a(t)$ after 1000 iterations for the cases 2B$_2$ and 3. Note the different axes in the figures. The performance highly depends on the quality of the estimate $\hat{q}_a(t)$.*

Remark 6.1. It can be seen that case 2A is a special case of the observer in case 2B with a particular choice of the design variables R_1 and R_2 in the Kalman filter, see (6.35). With

$$R_v = \operatorname{diag}\begin{pmatrix} 0 & 1 & 0 & \alpha \end{pmatrix}, \quad \alpha \to \infty$$
$$R_w = 1$$

only the process noise on the motor angular acceleration $\ddot{q}_m(t)$ is entered. Relatively small process noise on the joint angular acceleration $\ddot{q}_a(t)$ implies that the estimate $\hat{q}_a(t)$ is derived by simulating the motor angular position $q_m(t)$ through the nominal dynamics $T_{yz}(q)$.

Numerical calculations show that the Kalman filter approaches

$$F_{\bar{u}}(q) \to 0, \qquad F_{\bar{y}}(q) \to T_{yz}(q)$$

when increasing the value of α. Hence, the estimate of the joint angular position is obtained by simulating the motor angular position through the nominal dynamics, which gives case 2A.

The general observer design has more degrees of freedom and this fact can explain why case $2B_1$, which only uses the motor angular position in the observer, still gives a slightly better error reduction of the actual joint angular position compared to case 2A, with $e_{\text{red},\infty} \approx 44\,\%$ compared to $e_{\text{red},\infty} \approx 55\,\%$ for case 2A, see Table 6.4. Case 2A still gives a significant improvement compared to case 1 ($e_{\text{red},\infty} \approx 110\,\%$) and the simple design compared to case 2B can still make case 2A a preferred choice in a real application.

The major advantage of the general observer, especially considering state-space formulation and Kalman filter design, is that it is straightforward to include additional measurements in the estimation. Additional measurements will make the tuning more complicated, but it can be done using optimisation techniques where the estimation error is minimised based on measurements from the true system, for example as described in (6.35). Using both measured motor angular position $\boldsymbol{q}_m$ and joint angular acceleration $\ddot{\boldsymbol{q}}_a$ in the observer improves the result of the joint significantly, because of the improved quality of the estimate. The result can be seen for case $2B_2$ in Table 6.4 with $e_{\text{red},\infty} \approx 3.1\,\%$, from Figure 6.6, and from the true joint angular position in Figure 6.7, which is very close to the reference.

6.7 Conclusions

In this chapter a framework is proposed for analysis of an ILC algorithm using an estimate of the controlled variable obtained from an observer-based estimation procedure. The need for such a framework is motivated by the fact that in some applications the controlled variable is not the measured variable, and the model suffers from uncertainties. The description is a natural extension of the system description in Norrlöf and Gunnarsson [2002a] and is focusing on the performance of the controlled variable. A general expression for the asymptotic error of the controlled variable when the ILC algorithm has converged is derived using the framework. The asymptotic error is illustrated for three types of cases and the influence of model errors is discussed. Furthermore, a numerical example of an ILC algorithm applied to a flexible two-mass model of a robot joint is studied for the case when the system is subject to model errors. The simulation shows better performance of the controlled variable when the estimate used in the ILC algorithm is based on a Kalman filter design instead of a nominal model. The benefit of fusing information from additional sensors with the original measurements available is illustrated. Naturally, a more accurate estimate improves the ILC system performance. It is also shown how the expression for the asymptotic error can be used in analysis and design.

7

Estimation-based ILC applied to a nonlinear robot model

In this chapter an ILC algorithm is applied to a nonlinear two-link robot model with flexible joints, where the joint angular positions are estimated by an extended Kalman filter (EKF) in two ways:

1. Using measurements of motor angular positions.
2. Using measurements of both motor angular positions and tool acceleration.

These estimates are then used in an ILC algorithm. The simulation results show that the performance of the joint angular positions is clearly improved compared to when only motor angular positions are used in the ILC algorithm. This is the case even if model errors are introduced. The work presented in this chapter serves as a case study for combining EKF and ILC for estimation-based ILC. The main reference is Wallén et al. [2009b].

7.1 Two-link robot model

The flexible nonlinear two-link robot model shown in Figure 7.1 is used in the simulation study. The model is fully described in Moberg et al. [2008], and corresponds to the second and third link of a large six-DOF serial industrial robot. The elasticity in the robot is modelled as flexible joints, while the links are assumed rigid. The deflection in each joint is described by the motor angular position $q_m(t)$ on the motor side of the joint and the joint angular position $q_a(t)$ on the link side of the joint. The robot tool position P, see Figure 7.1, is derived from the joint angular positions by the forward kinematics[1].

[1] The main equations for the robot model are summarised in Section 2.3.3.

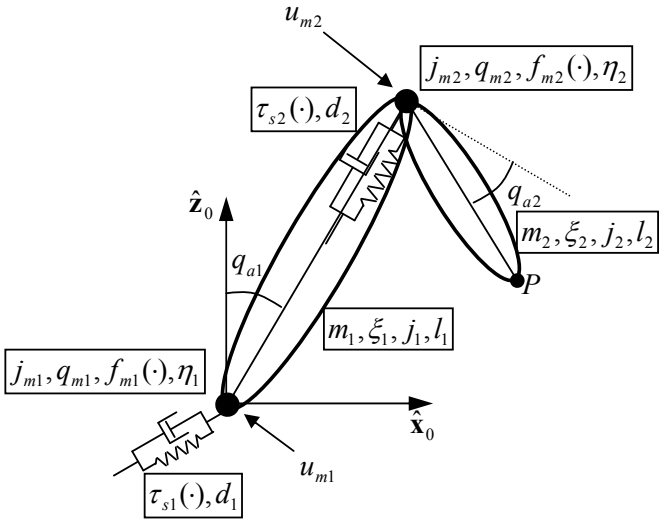


Figure 7.1: *The two-link robot model used in the simulation. The rigid links are described by mass m, link length l, center of mass ξ and inertia j with respect to the center of mass. The elastic joints are described by gear ratio η, nonlinear spring torque τ_s and linear damping d. The nonlinear friction torque f acts on the motors. The deflection in each joint is described by the motor angular position q_m and joint angular position q_a.*

Each link is modelled as a rigid body with mass m, link length l, center of mass ξ and inertia j with respect to the center of mass. Each elastic joint is described by a nonlinear spring torque τ_s, linear damping d and gear ratio η. The nonlinear friction torque f is approximated as acting only on the motors. The motor angular position vector $q_m(t)$ is the measured variable of the robot system[2], and it is subject to one-sample time delay. Additionally, the motor torque vector $u_m(t)$ is subject to saturation. Now introduce the vector $q(t)$ of angular positions,

$$q(t) = \begin{pmatrix} q_{a1}(t) & q_{a2}(t) & q_{m1}(t)/\eta_1 & q_{m2}(t)/\eta_2 \end{pmatrix}^T$$

The robot dynamics is described by

$$M(q)\ddot{q} + C(q, \dot{q}) + G(q) + D\dot{q} + K(q) + F(\dot{q}) = u(t) \tag{7.1}$$

where the time dependence of the vector of angular positions, $q(t)$, is omitted for brevity, see Section 2.3.3 for the details. The robot dynamics is described by the inertia matrix $M(q)$, the vector $C(q, \dot{q})$ of speed-dependent torques (Coriolis and centripetal terms), the vector $G(q)$ of gravity torques, the joint flexibilities described by linear viscous damping D and nonlinear stiffness $K(q)$. The vector $F(\dot{q})$ denotes nonlinear friction torques and $u(t)$ is the vector of input torques. If the

[2]This reflects the properties of a commercial robot system, where normally only the motor angular positions are measured [Brogårdh, 2009, Spong et al., 2006].

terms $D\dot{q}$ and $K(q)$ are omitted, the robot dynamics (7.1) results in a standard rigid-body model. By introducing the state vector $x(t)$ as

$$x(t) = \begin{pmatrix} q(t)^T & \dot{q}(t)^T \end{pmatrix}^T$$

and denoting

$$\begin{aligned} x_1(t) &= \begin{pmatrix} q_{a1}(t) & q_{a2}(t) \end{pmatrix}^T \\ x_2(t) &= \begin{pmatrix} q_{m1}(t)/\eta_1 & q_{m2}(t)/\eta_2 \end{pmatrix}^T \\ x_3(t) &= \dot{x}_1(t) \\ x_4(t) &= \dot{x}_2(t) \end{aligned}$$

the robot dynamics (7.1) can be described by the nonlinear state-space model

$$\dot{x}(t) = \begin{pmatrix} x_3(t) \\ x_4(t) \\ M_a^{-1}\big(x_1(t)\big)\Big(- C\big(x_1(t), x_3(t)\big) - G\big(x_1(t)\big) - A\big(x(t)\big)\Big) \\ M_m^{-1}\Big(A\big(x(t)\big) + F\big(x_4(t)\big) + u_m(t)\Big) \end{pmatrix} \tag{7.3}$$

with $M_a\big(x_1(t)\big)$ and M_m denoting the inertia matrices for the links and motors, respectively, and

$$A\big(x(t)\big) = D\big(x_3(t) - x_4(t)\big) + K\big(x_1(t), x_2(t)\big) \tag{7.4}$$

The robot system used in the simulation is schematically pictured in Figure 7.2. The joint angular position reference $r_a(t)$ is computed from the desired tool trajectory by means of the kinematic robot model. The corresponding motor angular position reference $r_m(t)$ is derived from the dynamic robot model. The robot model is controlled using independent joint control, represented by the controller block in Figure 7.2, and by a model-based feedforward term $u_{\text{ff}}(t)$. The figure also illustrates how the ILC input signal $u_k(t)$ at iteration k is added to the motor angular position reference $r_m(t)$, and thereby works as a complement to the conventional controller. The robot system is implemented in MATLAB and SIMULINK with a sampling interval $T_s = 0.25\,\text{ms}$.

In the simulation an accelerometer is attached to the robot tool. The tool acceleration equations can be derived by differentiating the velocity kinematics, resulting in

$$a_b(t) = J\big(q_a(t)\big)\ddot{q}_a(t) + \dot{J}\big(q_a(t)\big)\dot{q}_a(t)$$

where $q_a(t)$ denotes the joint angular position vector and $J(\cdot)$ is the Jacobian, see (2.6). The tool acceleration is expressed in the base coordinate frame of the robot, denoted by subscript b. The measured acceleration $a^M(t)$ includes the effect of gravity g_b and is expressed in the coordinate system of the sensor. This can be described by the relation

$$a_s^M(t) = R_s^b\big(q_a(t)\big)\big(a_b(t) - g_b\big) + \delta_s(t) + w_s(t)$$

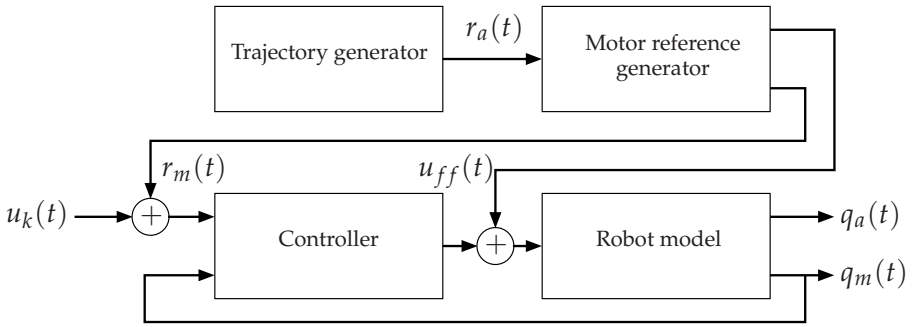


Figure 7.2: *The robot system used in the simulation. The motor angular position reference $r_m(t)$ is computed using the kinematic and dynamic models. The robot model is controlled by independent joint control, represented by the controller block, and by a model-based feedforward term $u_{ff}(t)$. The ILC input signal $u_k(t)$ at iteration k is added to $r_m(t)$.*

where the accelerometer measurement is subject to drift $\delta_s(t)$ and measurement noise $w_s(t)$, both expressed in the sensor coordinate frame. The transformation from the base frame to the coordinate frame of the sensor (denoted s) is described by the transformation matrix $R_s^b(\cdot)$. The measured acceleration can then be modelled as

$$a_s(t) = R_s^b\Big(x_1(t)\Big)\Big(J\Big(x_1(t)\Big)\dot{x}_3(t) + \dot{J}\Big(x_1(t)\Big)x_3(t) - g_b\Big) + \delta_s(t) + w_s(t) \tag{7.5}$$

In the simulations the drift and disturbances are set to zero for simplicity. To avoid problems with drift in the estimates as described in Henriksson et al. [2009], the assumption is made that the robot is subject to no gravitation, that is, $G(\cdot)$ and g_b is zero. The tool motion studied in the simulation is half a circle in the $\hat{x}_0\hat{z}_0$-plane with radius 5 mm. The resulting joint angular position vector $q_a(t)$ when following the desired trajectory can be seen in Figure 7.3. Since the aim of this chapter is to perform a case study when combining ILC and EKF, the motion is kept near the working point in order to keep the nonlinear properties of the system small.

Linearisation of the robot model

For the EKF, the state-space description (7.3) is discretised using the Euler forward difference approximation, and is thereafter linearised around the previous estimate, as described in Section 3.2.2.

For analysis of the convergence properties of the two-link robot model controlled by an ILC algorithm, the robot dynamics is linearised around the working point. This gives a relation from $u_k(t)$ as input and the motor and joint angular positions as output.

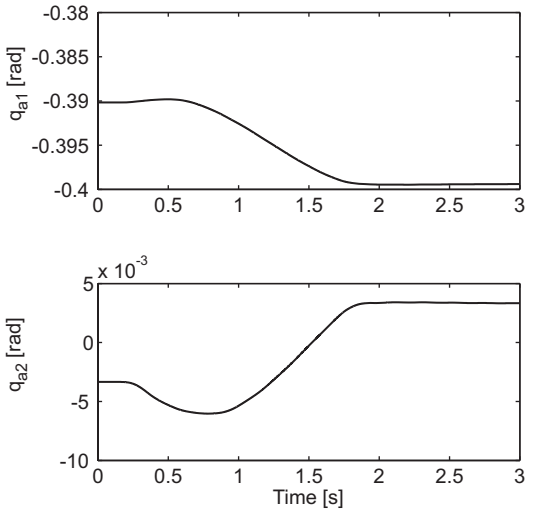


Figure 7.3: *The desired tool motion is half a circle, with the resulting joint angular positions $q_{a1}(t)$ and $q_{a2}(t)$ shown in the figure.*

7.2 Estimation algorithms

In this chapter an EKF algorithm is used to estimate the joint angular position vector $q_a(t)$. The reader is directed to Section 3.2.2 and the general references Anderson and Moore [1979] or Kailath et al. [2000] for a description of the estimation algorithm.

The work is based on the results presented in Henriksson et al. [2009]. In this paper different versions of the EKF as well as the deterministic observer by De Luca et al. [2007] are used for estimating the robot tool position, and the methods are verified experimentally on an industrial robot. For the details regarding the estimation, tuning process and robustness of the methods, the reader is directed to Henriksson et al. [2009]. In this chapter the following two ways of producing the estimate $\hat{q}_a(t)$ are used, called EKF Motor and EKF Complete.

EKF Motor The joint angular position $q_a(t)$ is estimated from measurements of motor angular position $q_m(t)$. The measurement equation (3.1b) is then given by

$$h\Big(x(t), u(t)\Big) = x_2(t)$$

The resulting estimate is denoted $\hat{q}_{a,\mathrm{M}}(t)$. The EKF Motor approach is used as a baseline for what can be achieved in the estimation without having additional sensors.

EKF Complete The joint angular position $q_a(t)$ is estimated from measurements of the motor angular position $q_m(t)$ and the tool acceleration $a(t)$. The measure-

ment equation (3.1b) is now given by

$$h\Big(x(t), u(t)\Big) = \begin{pmatrix} x_2(t) \\ a_s(t) \end{pmatrix}$$

with the measurement equation of the acceleration given by (7.5). The resulting estimate is denoted $\hat{q}_{a,\mathrm{C}}(t)$. For EKF Complete the description of the two-link robot model is fully utilised, including friction and nonlinear stiffness.

The tuning of the noise covariances in the estimation approaches are performed automatically, as is described in Henriksson et al. [2009]. In the tuning, the objective is to minimise the error $q_a(t) - \hat{q}_a(t)$ on a set of data from simulations where the robot performs various movements near the actual working point.

In Figure 7.4 the estimation error for the estimates $\hat{q}_{a,\mathrm{M}}(t)$ and $\hat{q}_{a,\mathrm{C}}(t)$ are compared when the robot tool performs the circular motion described in Section 7.1. The case with nominal model parameters is compared to when model errors are introduced by modifying the joint stiffness parameters by −30 %. It can be seen that the estimate $\hat{q}_{a,\mathrm{C}}(t)$ from EKF Complete is better than the estimate $\hat{q}_{a,\mathrm{M}}(t)$ from EKF Motor, especially for the case with model errors, as is also noted in Henriksson et al. [2009]. This is natural, since EKF Motor relies on only motor angular position measurements and the dynamic robot model to estimate the tool position. EKF Complete is therefore more robust with respect to model errors, since it also includes measurements of the tool behaviour.

It should be noted that more computational resources are required for EKF Complete than for EKF Motor. Although the EKF in combination with ILC is applied offline, this will become an important issue for the experimental implementation for problems with larger dimensions.

7.3 ILC algorithms

For the analysis, a linear system in the disturbance-free case is studied, described by the model (6.1),

$$y_k(t) = T_{ry}(q)r(t) + T_{uy}(q)u_k(t) \tag{7.6a}$$

$$z_k(t) = T_{rz}(q)r(t) + T_{uz}(q)u_k(t) \tag{7.6b}$$

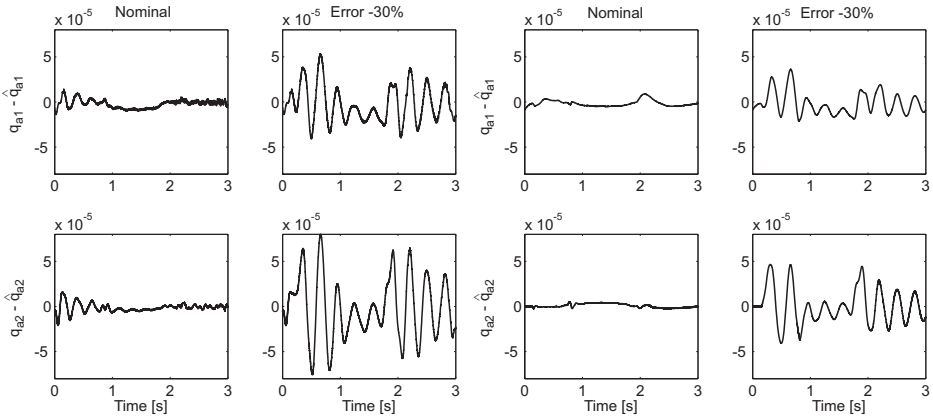
with the measured variable $y_k(t)$ and controlled variable $z_k(t)$. The system is controlled by an ILC algorithm of the form (4.12),

$$u_{k+1}(t) = Q(q)\Big(u_k(t) + L(q)\epsilon_k(t)\Big) \tag{7.7}$$

with the error $\epsilon_k(t)$ given by

$$\epsilon_k(t) = r(t) - \hat{z}_k(t) \tag{7.8}$$

and where $\hat{z}_k(t)$ is an estimate of the controlled variable $z_k(t)$. For the two-link robot model studied in this chapter, the controlled variable is the joint angular position $q_a(t)$.



(a) *EKF Motor* ***(b)*** *EKF Complete*

Figure 7.4: *The estimation error $q_a(t) - \hat{q}_a(t)$ shown for the two joint angular positions for the case with nominal parameters and modified joint stiffness parameters by −30 %. a) EKF based on motor angular position measurements (EKF Motor). b) EKF based on measurements of motor angular position and tool acceleration (EKF Complete).*

The cases to be studied can be divided into:

1. $q_m(t)$ is the measured variable. In the ILC update equation the following error $\epsilon_k(t) = r_m(t) - q_{m,k}(t)$ is used.

2A. $q_a(t)$ is estimated based on measurements of $q_m(t)$, resulting in the estimate $\hat{q}_{a,\mathrm{M}}(t)$ from EKF Motor. The error $\epsilon_k(t) = \Pi\Big(r_a(t) - \hat{q}_{a,\mathrm{M},k}(t)\Big)$, where Π is a matrix of gear ratios, is used in the ILC update equation.

2B. $q_a(t)$ is estimated based on measurements of $q_m(t)$ and tool acceleration $a(t)$, resulting in the estimate $\hat{q}_{a,\mathrm{C}}(t)$ by using EKF Complete. The ILC update equation uses the error $\epsilon_k(t) = \Pi\Big(r_a(t) - \hat{q}_{a,\mathrm{C},k}(t)\Big)$.

In this chapter an ILC algorithm with a diagonal structure is assumed, that is, the joints will be treated individually as decoupled systems with a separate ILC algorithm (7.7) for each joint. The following choices of filters $Q(q)$ and $L(q)$ are used in the simulation study:

$Q(q)$: Low-pass second-order Butterworth filter with cutoff frequency f_c above the resonance frequency of the controlled system. The filter is applied by filtering the signal forwards and backwards in time to give zero-phase characteristics of the filter $Q(q)$.

$L(q)$: The filter is given by $L(q) = \gamma q^{\delta}$, with learning gain γ and time shift of δ samples.

The same ILC filters are used for both motors for simplicity.

7.4 Simulation results

The results achieved when using estimates of the joint angular positions in the ILC update equation (estimation-based ILC) are compared to the results when using the measured motor angular positions directly in the ILC algorithm. The flexible nonlinear robot model described in Section 7.1 is used in the simulation. The reference $r_m(t)$ is computed from the nominal system, see Figure 7.2, and can be seen as a filtered version of the joint angular position reference $r_a(t)$. In the simulation the joint angular position tracking error $r_a(t) - q_a(t)$ is used as evaluation variable for all cases.

To evaluate performance and robustness when using estimated joint angular position in the ILC update equation (7.7), the stiffness parameters of the joints are changed by −30 % in the robot model. The reduction of the error vector denoted $e_k(t)$ is evaluated by

$$\bar{e}_k = 100 \cdot \frac{\|\boldsymbol{e}_k\|}{\|\boldsymbol{e}_0\|} \quad [\%] \tag{7.9}$$

where $\boldsymbol{e}_k$ is normalised by the error when no ILC algorithm is applied (iteration 0), and is studied in both 2-norm and ∞-norm.

7.4.1 Case 1 — ILC using measured motor angular position

For case 1 the motor angular position error $\epsilon_k(t) = r_m(t) - q_m(t)$ is used in the ILC update equation (7.7). The ILC parameters used in the simulation are cutoff frequency $f_c = 8\,\text{Hz}$, gain $\gamma = 0.5$ and time shift $\delta = 70$.

For the robot system, the relation from the vector of ILC input signals to the measured motor angular positions (measured variable $y_k(t)$) is described by

$$\begin{pmatrix} y_{k,1}(t) \\ y_{k,2}(t) \end{pmatrix} = \underbrace{\begin{pmatrix} T_{uy,11}(q) & T_{uy,12}(q) \\ T_{uy,21}(q) & T_{uy,22}(q) \end{pmatrix}}_{T_{uy}(q)} \begin{pmatrix} u_{k,1}(t) \\ u_{k,2}(t) \end{pmatrix} + \underbrace{\begin{pmatrix} T_{ry,11}(q) & T_{ry,12}(q) \\ T_{ry,21}(q) & T_{ry,22}(q) \end{pmatrix}}_{T_{ry}(q)} \begin{pmatrix} r_1(t) \\ r_2(t) \end{pmatrix}$$

When applying the ILC algorithm for each motor $i = 1, 2$,

$$u_{k+1,i}(t) = Q_i(q)\Big(u_{k,i}(t) + L_i(q)\epsilon_{k,i}(t)\Big)$$

it results in the ILC system

$$\begin{pmatrix} u_{k+1,1}(t) \\ u_{k+1,2}(t) \end{pmatrix} = \underbrace{\begin{pmatrix} Q_1(q)\Big(1 - L_1(q)T_{uy,11}(q)\Big) & -Q_1(q)L_1(q)T_{uy,12}(q) \\ -Q_2(q)L_2(q)T_{uy,21}(q) & Q_2(q)\Big(1 - L_2(q)T_{uy,22}(q)\Big) \end{pmatrix}}_{H(q)} \begin{pmatrix} u_{k,1}(t) \\ u_{k,2}(t) \end{pmatrix}$$

$$+ \underbrace{\begin{pmatrix} Q_1(q)L_1(q)\Big(1 - T_{ry,11}(q)\Big) & -Q_1(q)L_1(q)T_{ry,12}(q) \\ -Q_2(q)L_2(q)T_{ry,21}(q) & Q_2(q)L_2(q)\Big(1 - T_{ry,22}(q)\Big) \end{pmatrix}}_{H_r(q)} \begin{pmatrix} r_1(t) \\ r_2(t) \end{pmatrix}$$

This can be written more compactly as

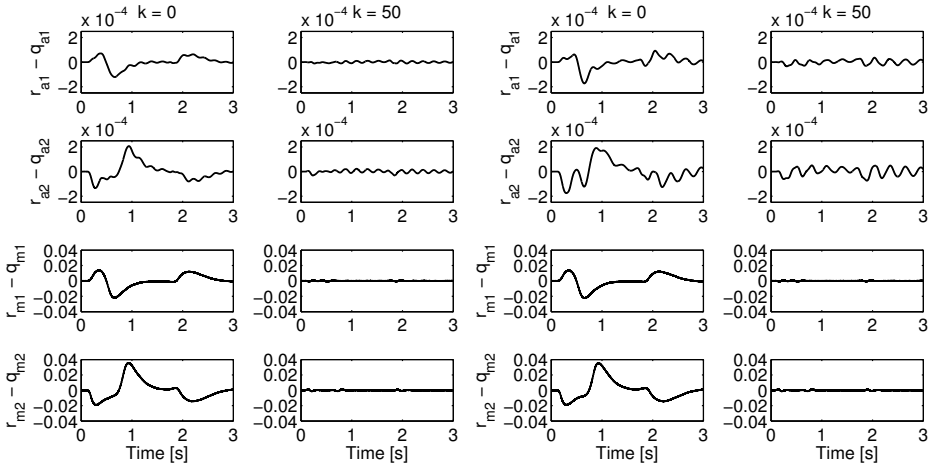
$$u_{k+1}(t) = Q(q)\Big(I - L(q)T_{uy}(q)\Big)u_k(t) + Q(q)L(q)\Big(I - T_{ry}(q)\Big)r(t) \tag{7.10}$$

where

$$Q(q) = \begin{pmatrix} Q_1(q) & 0 \\ 0 & Q_2(q) \end{pmatrix}, \qquad L(q) = \begin{pmatrix} L_1(q) & 0 \\ 0 & L_2(q) \end{pmatrix}$$

The same ILC filters are used for both motors for simplicity, that is, $Q_1(q) = Q_2(q)$ and $L_1(q) = L_2(q)$. The ILC system (7.10) is stable with monotone convergence of the ILC input signal $\boldsymbol{u}_k$ if the maximum singular value satisfies $\bar{\sigma}(\boldsymbol{H}) < 1$, according to Theorem 6.2. For case 1 and the parameters used in the simulation, it results in $\bar{\sigma}(\boldsymbol{H}) \approx 0.9 < 1$ for both the model with nominal parameters and when model errors are introduced.

In Figure 7.5 the motor and joint angular position errors are shown when no ILC algorithm is applied to the system (denoted iteration $k = 0$), together with the errors after iteration $k = 50$. The case for a model with nominal parameters is compared to when a model error of $-30\,\%$ for the stiffness parameters is introduced. In Figure 7.6 the error measure (7.9) in 2-norm for the resulting motor angular position errors and joint angular position errors are shown. As can be seen, the error reduction for the joints for iteration $k = 50$ is smaller for the case with model errors introduced compared to the case with nominal stiffness parameters.



(a) *Nominal model parameters* ***(b)*** *Model error*

Figure 7.5: *Case 1: Resulting motor and joint angular position errors for ILC iteration $k = 50$, compared to the errors when no ILC algorithm is applied to the system (denoted $k = 0$). a) nominal system parameters, b) model errors introduced. The error reduction is smaller for the case with model errors, as expected. The different scaling for motor angular position and joint angular position is due to the gear ratio $\eta = 200$.*

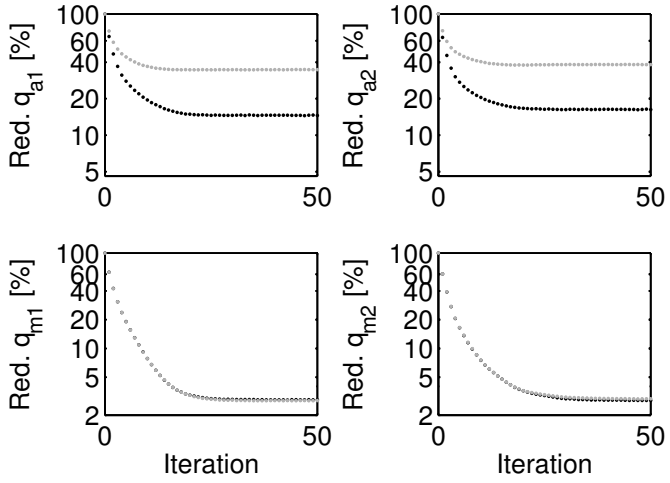


Figure 7.6: *Case 1: Error measure* (7.9) *in 2-norm as a function of iteration number for the motor and joint angular position errors. System with nominal parameters (black) and the system with model errors (grey).*

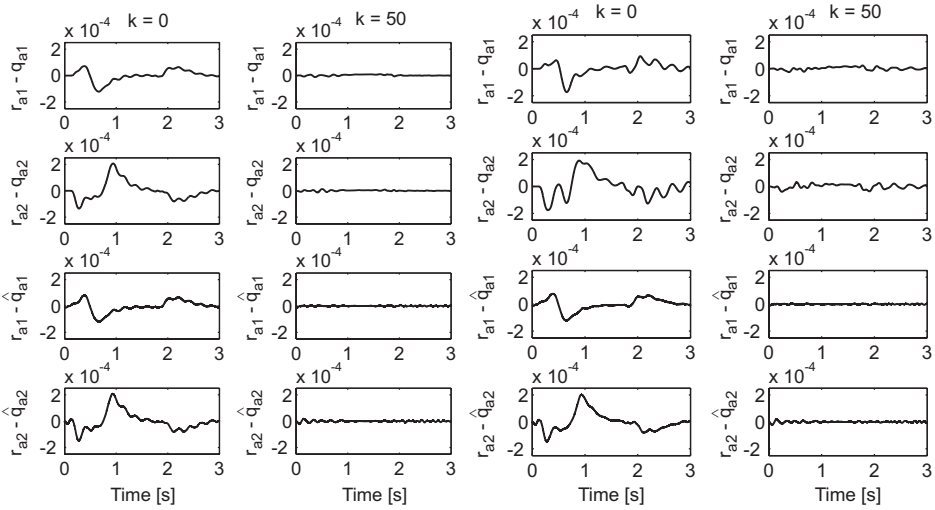
It can be explained by the fact that the motor angular position reference $r_m(t)$ is a filtered version of the joint angular position reference $r_a(t)$. This works as long as the model from actual motor angular position $q_m(t)$ to actual joint angular position $q_a(t)$ is correct, so that controlling $q_m(t)$ towards $r_m(t)$ will give $q_a(t)$ close to $r_a(t)$. In the case of model errors, $q_m(t)$ is still controlled towards $r_m(t)$, but $q_m(t)$ now does not result in correct value of $q_a(t)$. This behaviour was also confirmed in the results from simulations and experiments discussed in Chapter 5.

7.4.2 Case 2 — ILC using estimated joint angular position

The result in the previous section is now compared to the case when an estimate $\hat{q}_a(t)$ is used in the ILC update equation (7.7). The ILC parameters are tuned in simulation to give a convergent ILC system for case 2A and case 2B, for both the system with nominal parameters and for the model-error case.

Case 2A

For case 2A the ILC update equation (7.7) uses the estimate $\hat{q}_{a,\mathrm{M}}(t)$ from EKF Motor. Tuning of the ILC parameters results in $f_c = 8$ Hz for the cutoff frequency of the ILC filter $Q(q)$, gain $\gamma = 0.5$ and time shift $\delta = 220$. The simulation results can be seen in Figures 7.7 and 7.8. Compared to the results for case 1, shown in Figures 7.5 and 7.6, it can be seen that the gap is reduced between the error reduction for the case with nominal model parameters and when model errors are introduced. This is also seen in Table 7.1, where the 2-norm and ∞-norm of error measure (7.9) for the joint angular position errors for iteration $k = 50$



(a) *Nominal model parameters* ***(b)*** *Model error*

Figure 7.7: *Case 2A: Resulting motor and joint angular position errors for ILC iteration $k = 50$, compared to the errors when no ILC algorithm is applied to the system (denoted $k = 0$). a) nominal system parameters, b) model errors.*

are compared. For example, from Table 7.1 it can be seen that the maximum gap between the error reduction in 2-norm for the nominal case and in case of model errors is 22 percentage points (motor 2). For case 2A the corresponding value is 13 percentage points (motor 2). In case of model errors, this means that the actual joint angular position is improved for case 2B compared to case 2A. However, since the cases represent different approaches and tuning of the ILC algorithms, it is only possible to do a qualitative comparison.

Case 2B

For case 2B the ILC update equation (7.7) uses the estimate $\hat{q}_{a,C}(t)$ from EKF Complete. Since the results only show a minor difference compared to the corresponding results for case 2A shown in Figure 7.7, this figure is omitted here. As can be seen in Figure 7.9 and Table 7.1, the error reduction is slightly better for EKF Complete than for EKF Motor in Figure 7.6. The maximum gap between the error reduction for the system with nominal parameters compared to when introducing model errors is now reduced to approximately 10 percentage points for case 2B (compare 13 percentage points for case 2A) when the 2-norm of the error measure (7.9) is studied. This result can be explained by the rather small reduction of estimation error with EKF Complete compared to EKF Motor, seen in Figure 7.4. This agrees with the result shown in Henriksson et al. [2009], where EKF Motor and EKF Complete show similar performance for some evaluation measures. However, EKF Complete will show its advantages for the case of

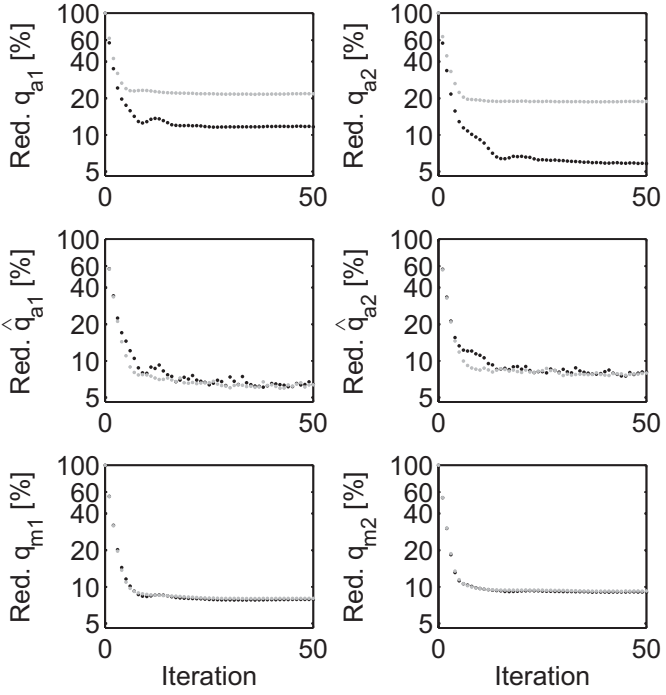


Figure 7.8: *Case 2A: Error measure* (7.9) *in 2-norm as a function of iteration number for the motor and joint angular position errors, for the case with nominal system parameters (black) and the system with model errors (grey).*

Table 7.1: *2-norm and* ∞*-norm of the error measure* (7.9) *for the joint angular positions for iteration* $k = 50$*, for the cases with a system with nominal parameters and when the joint stiffness parameters are modified by* -30 *%.*

Conditions	Nom. e_{a1}	Nom. e_{a2}	Mod. err. e_{a1}	Mod. err. e_{a2}
Case 1, $\|\cdot\|_2$	14.6 %	16.3 %	34.7 %	38.2 %
$\|\cdot\|_\infty$	11.7 %	15.6 %	21.1 %	39.0 %
Case 2A, $\|\cdot\|_2$	11.7 %	5.8 %	21.8 %	18.8 %
$\|\cdot\|_\infty$	9.6 %	6.5 %	14.5 %	20.5 %
Case 2B, $\|\cdot\|_2$	7.6 %	5.6 %	17.5 %	14.6 %
$\|\cdot\|_\infty$	6.2 %	6.0 %	11.9 %	21.1 %

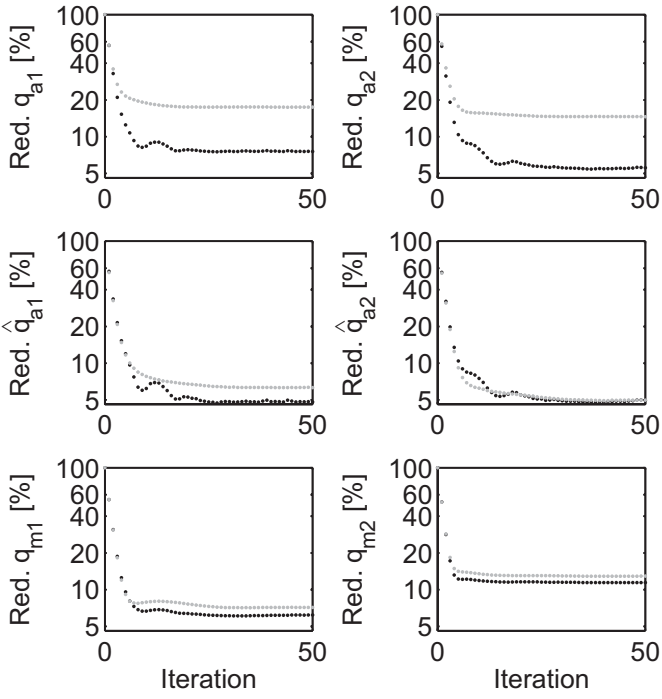


Figure 7.9: *Case 2B: Error measure* (7.9) *as a function of iteration number for the motor and joint angular position errors for the case with nominal system parameters (black) and the system with model errors (grey).*

a tool-force disturbance, as seen in Henriksson et al. [2009], which is a scenario not investigated here.

To summarise, the overall conclusion is that it is possible to achieve better performance of the tool by using an estimate of the joint angular position $q_a(t)$ in the ILC algorithm. Tuning of the EKF and/or the ILC algorithm with respect to the model error can improve the results further.

7.5 Conclusions

In this chapter an extended Kalman filter (EKF) and an ILC algorithm are combined for a flexible nonlinear two-link robot model, and the simulation study serves as a case study of estimation-based ILC. Estimates of the joint angular positions are derived in two ways; 1) using measurements from motor angular positions, and 2) using measurements from both motor angular positions and tool acceleration. The case where the ILC update equation is based on only motor angular positions is compared to the case where estimated joint angular positions

are used in the ILC algorithm. Robustness and performance are studied for the different cases. The results show that it is possible to improve tool performance when an estimate is used in the ILC algorithm, compared to when only originally available measurements of motor angular positions are used. In Chapter 8, an experimental comparison of ILC using tool-position estimates is performed on a parallel robot.

8

Estimation-based ILC applied to a parallel robot

THREE DIFFERENT APPROACHES of ILC applied to a parallel robot are studied in this chapter. First, the ILC algorithm is using measured motor angular positions, which normally are the only measurements available in a commercial robot system. Second, tool-position estimates are used in the ILC algorithm. Finally, for evaluation purposes, the ILC algorithm uses the actual tool position measured by external sensors. The approaches are evaluated experimentally on a Gantry-Tau parallel robot prototype, and compared by means of the resulting tool performance. The principal experimental result of ILC using estimates is discussed and serves as an experimental evaluation of the results presented in Chapters 6 and 7. It is concluded from the experiments that the tool performance can be improved by using tool-position estimates in the ILC algorithm, compared to when directly using measurements of motor angular positions in the algorithm.

The case when applying an ILC algorithm to a system following trajectories with lead-in/lead-out is considered. Dynamic modelling of the Gantry-Tau robot is also discussed. The derived models are used for estimation of the robot tool position and for tuning of the ILC filters, thus enabling learning of frequency components of the error also above the dominating resonance frequencies of the system. The work presented in this chapter is based on Wallén et al. [2010a,b].

8.1 Introduction

Parallel robots have potential of high performance due to their higher stiffness, accuracy and achievable accelerations compared to serial robots. Although the interest in parallel robots was renewed by the Stewart-Gogh platform, only the Delta structure has found a noteworthy application in industry [Merlet, 2006].

The first challenge regarding parallel robots is to develop a mechanical structure useful for high-performance applications [Brogårdh, 2002]. Most parallel robots have the disadvantage of a small workspace. However, the Gantry-Tau configuration [Johannesson et al., 2004], which is shown in Figures 8.1 and 8.2, has a large workspace compared to other parallel robots, while still being stiff compared to serial robots. Another advantage of the Gantry-Tau robot is the possibility of a modular construction and reconfiguration for flexible manufacturing of small lot sizes [Dressler et al., 2007a] or of machining large components with high accuracy, for example in aerospace industry [Crothers et al., 2010].

The second challenge is the performance needed for applications where the serial robot technology is unsatisfactory [Brogårdh, 2002]. Dressler et al. [2010] showed the importance of kinematic calibration and accurate dynamic modelling to achieve high precision manufacturing with the Gantry-Tau robot. One way of improving robot performance is to rely on high-precision components and assembly together with highly accurate and often expensive measurement devices, as well as very detailed models. Another method of performance improvement could be to use ILC. It is difficult to achieve good results with only motor angular positions in the ILC algorithm, since the tool position converges towards an incorrect value when the robot model suffers from uncertainties, see Chapter 5. By using additional sensors in combination with signal processing and estimation algorithms, estimates of the robot tool position can be obtained, as is discussed in Karlsson and Norrlöf [2005]. These estimates can then be used in an ILC algorithm, and hence be able to compensate for errors originating from the robot structure.

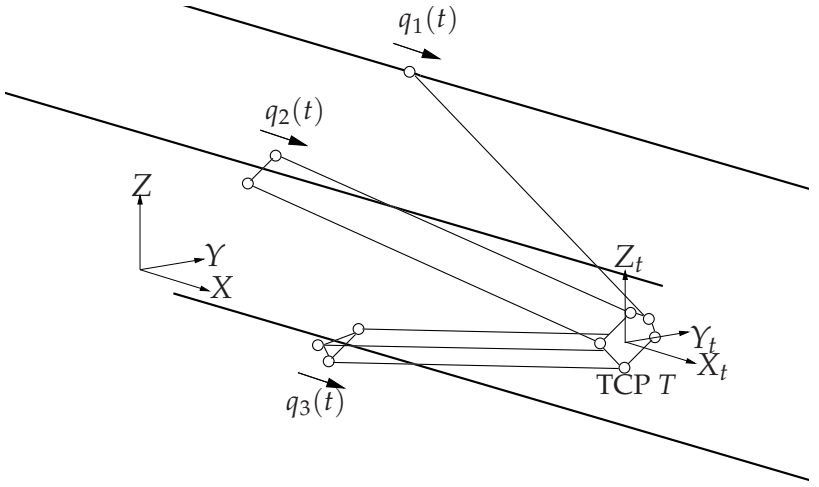


Figure 8.1: *Schematic picture of the Gantry-Tau parallel robot, with the base coordinate frame XYZ, the coordinate frame $X_tY_tZ_t$ of the end-effector plate and the tool position T. Three carts move on guideways. The corresponding cart positions are denoted $q_1(t)$, $q_2(t)$ and $q_3(t)$.*

In this chapter, experimental results for ILC algorithms using measurements of motor angular positions are compared to results for ILC algorithms using tool-position estimates. For evaluation purposes, the tool performance is measured by length gauges, which are not available in practical applications. Experiments are also performed to illustrate the improvement for the (ideal) case when measurements of the controlled variable are available for usage in the ILC algorithm. The main contributions of this chapter are:

- Experimental evaluation of estimation-based ILC by applying an ILC algorithm using tool-position estimates on the Gantry-Tau parallel robot.
- Learning of frequency components of the error up to and above the dominating resonance frequencies of the robot system by model-based tuning of the ILC filter $Q(q)$.
- Experimental study of the case with ILC algorithms applied to a system following trajectories with lead-in/lead-out.

There are only few earlier contributions where ILC algorithms are applied to parallel robots. In for example Abdellatif et al. [2006] and Abdellatif and Heimann [2010], linear ILC algorithms are applied to a direct-driven hexapod with arm slider length of 1 m. The ILC algorithms use the measured joint positions, and evaluation of the tool performance is performed by transforming the joint positions by the forward kinematics to a corresponding tool position. In Cheung and Hung [2009], an ILC algorithm using measured joint positions is applied to a planar parallel direct-driven robot prototype with a small workspace (4×2 cm) intended for semiconductor packaging operations. The ILC algorithm is intended to reduce the low-frequency error components, and therefore higher frequency components of the error, especially around the resonance frequency of the system, are not learned. A high-precision laser measurement system provides measurements of the resulting tool position. In Burdet et al. [2001], ILC algorithms using the feedback motor torque at the previous iteration are applied to a three-DOF micro-Delta parallel robot and two sizes of two-DOF parallel robots with arm lengths of 12 cm and 80 cm, respectively. To overcome problems with a robot structure containing mechanical flexibilities, the learning algorithm has a low bandwidth to prevent instability. A three-DOF direct-driven Stewart platform with arm lengths of approximately 0.5 m is used in Chuang and Chang [2001]. The performance is improved by using an ILC algorithm based on measured joint angular positions.

To summarise, the experiments in the experimental studies mentioned above are performed on small parallel robots with different design and in many cases stiffer mechanical structure compared to the Gantry-Tau robot. Problems with flexible dynamics are solved by learning of the frequency-components of the error only below the resonance frequencies of the system.

8.2 Robot and sensors

In this section the Gantry-Tau robot prototype is presented in more detail, as well as the external sensors used in the experiments.

8.2.1 Gantry-Tau robot

The Gantry-Tau robot, shown in Figures 8.1 and 8.2, is a Gantry variant of the Tau parallel robot [Crothers et al., 2010]. Each chain is driven by a linear actuator consisting of a cart moving on a guideway. The three carts are connected to the end-effector plate via link clusters, grouped in a 3-2-1 configuration. The links, with lengths of 1.8 – 2 m, are connected to the carts and end effector by spherical joints. The resulting cross-section area in the YZ-plane of the workspace is 1.1×0.95 m. The Tau configuration is such that the links belonging to one cluster form parallelograms. This results in a constant end-effector orientation, and the Gantry-Tau robot has three purely translational DOF. There exist several possibilities of extending the Gantry-Tau configuration to a robot with higher DOF. One, which is implemented on the prototype used in this work, is to add a two-axis serial wrist on the end-effector plate of the parallel robot, see Figure 8.2b, to give altogether five DOF. The wrist joints are not used in the experiments.

For the robot prototype used here and the calibrated nominal kinematic model, the mean static end-effector positioning error over the whole workspace is experimentally identified to be 140 μm.

8.2.2 Control system

The Gantry-Tau robot is controlled by an industrial ABB IRC5 system, where the trajectory generator is based on the kinematic model in Dressler et al. [2007b]. Implementation of customised control solutions and integration of external sensors are enabled by an extension to the IRC5 system [Blomdell et al., 2005], where signals sent from the IRC5 main system can be read and modified externally. Data from external sensors are obtained directly in the shared memory, and are synchronised with the robot system at 250 Hz sampling rate.

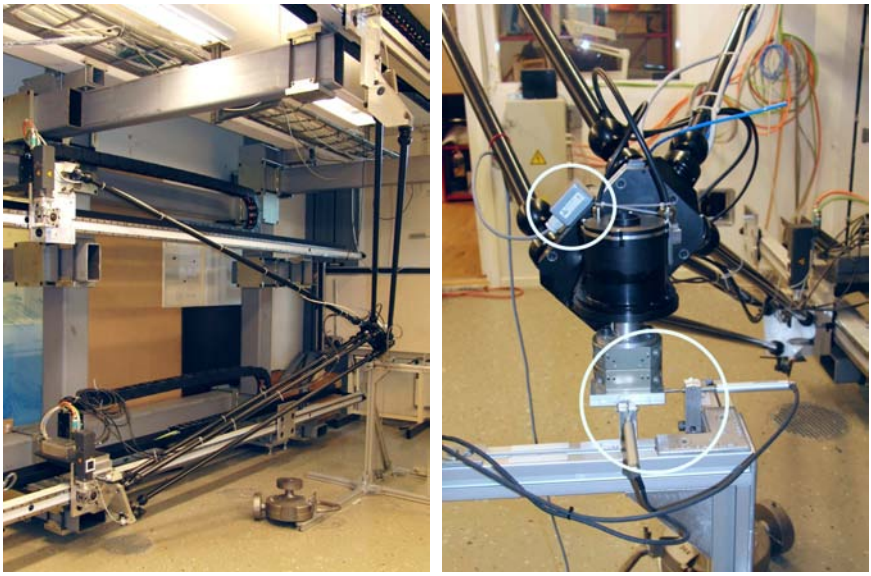
Each motor is controlled independently with a standard cascade P/PI-controller with outer position and inner velocity loop[1]. For the ILC experiments, the motor position references and motor velocity references are modified.

8.2.3 External sensors

In the experiments, the acceleration of the robot tool is measured by an additional accelerometer placed on the end-effector plate. The actual tool position is evaluated using length gauges.

3-DOF accelerometer The accelerometer MMA7361L from Freescale [2010], is mounted on the three-DOF end effector, see Figure 8.2b. The Gantry-Tau robot

[1] Independent joint control, see Section 2.4.2.



(a) *Robot* ***(b)*** *External sensors*

Figure 8.2: *Experimental setup of the Gantry-Tau robot; a) robot structure, b) mounting of length gauges and accelerometer.*

has a constant end-effector orientation, which means that the accelerometer orientation is not changed during the experiments. The measured standard deviation of the measurement noise for all channels is 0.06–0.07 $\mathrm{m/s^2}$.

Length gauges Two length gauges ST 3078 from Heidenhain [2010] mounted on a stand are used for measuring the tool position, see Figure 8.2b. As only two gauges are available, only the motion in the XY-direction (horizontal motion) is measured. The gauges have a range of 30 mm and accuracy of ± 1 μm.

8.3 System properties

In this section, the nominal performance of the robot is discussed. The dynamic behaviour as well as the dynamic repeatability are investigated when following a rectangular path.

8.3.1 Trajectory

In the experiments, a rectangular path with side 10 mm is used, see Figure 8.3b, with $v(t) = 100$ mm/s being the highest achievable tool velocity due to acceleration limits. Sharp edges, due to the velocity references set to zero in the corners, and a large tool velocity in between pronounce the dynamic characteristics of the robot. The motion is performed in two consecutive rounds. This enables to di-

vide the trajectory into an initial motion, called lead-in, followed by the actual trajectory to be learned, called learning part, and a lead-out. Lead-in/lead-out is useful in practical applications, for example laser cutting where a constant tool velocity along the path is required. The full potential of lead-in is not utilised in this work, since the path corners are programmed with zero velocity. The usage of lead-in/lead-out can still be motivated by the experimental evaluation of the new combination of ILC and lead-in/lead-out, see Section 8.5.

8.3.2 Nominal performance

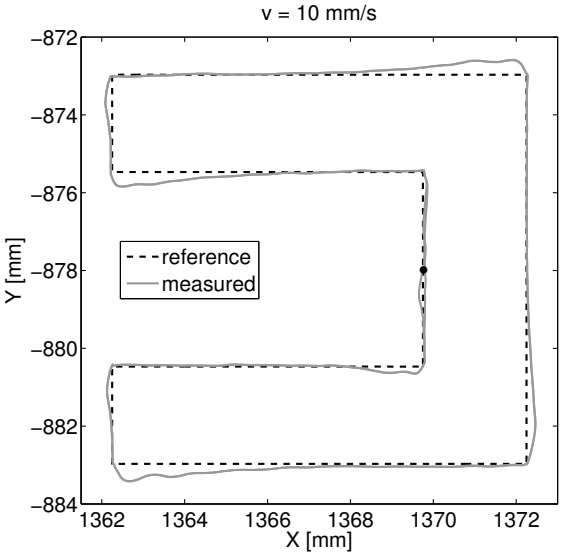
To illustrate the dynamic properties of the Gantry-Tau robot, the rectangular path is first programmed with a low tool velocity of $v(t) = 10\,\text{mm/s}$. The resulting tool performance is then compared with an experiment following the same path with a high programmed tool velocity of $v(t) = 100\,\text{mm/s}$. The results are depicted in Figure 8.3, where the actual tool behaviour measured in the XY-direction is compared to the reference path. The influence of dynamic effects in the robot structure can be considered as small for the low-velocity motion. One possible explanation of the remaining errors seen in Figure 8.3a can therefore be friction in the motors and drivelines. The oscillatory behaviour, especially seen in the lower left corner, could possibly be explained by static friction when measuring the tool position by the length gauges sliding on the tool plate, see Figure 8.2b. In Figure 8.3b the corresponding result is shown for the high-velocity experiment. The larger deviation from the reference can be explained both by larger motor control errors compared to the low-velocity motion and by dynamic effects in the robot structure. As is discussed in more detail in Section 8.4, the robot is stiff in the X-direction, while it has a distinct resonance frequency at around 11.5 Hz in the Y-direction in this operating point. One possible explanation of the overshoot after each corner could be the flexible structure.

8.3.3 Repeatability

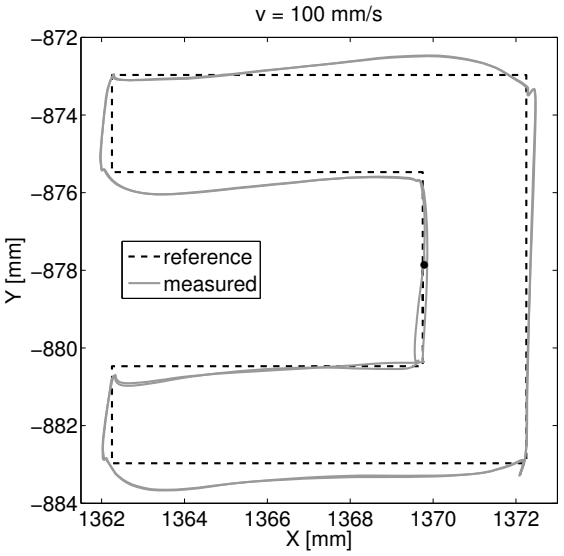
Repeatability of a system is a key property for success when using ILC, since only the repetitive part of the error can be corrected by the ILC algorithm. In order to investigate the repeatability of the Gantry-Tau robot, five identical experiments have been performed. The repeatability is evaluated by the difference

$$e_{\text{rep}}(t) = \bar{q}_i^1(t) - \bar{q}_i^j(t), \quad i = 1,\ldots,3, \quad j = 2,\ldots,5 \tag{8.1}$$

with $\bar{q}_i^j(t)$ denoting the calculated cart position for cart i and experiment j, given by $q_{m,i}^j(t)/\eta$, the motor angular position divided by the gear ratio. Figure 8.4 illustrates the difference (8.1) for cart 3 and is representative for the results for all carts. The maximum value of $e_{\text{rep}}(t)$ of around 40 μm is small compared to the maximum initial error of around 1.5 mm (see Figure 5.13). The non-repeatability can probably be explained by static friction, which requires a varying amount of time to integrate the torques to overcome the friction for different experiments. Changed experimental conditions, like temperature, can cause larger differences and experiments have shown that it is beneficial with lead-in/lead-out.



(a) *Low velocity*



(b) *High velocity*

Figure 8.3: *Nominal performance of the robot for low tool velocity ($v(t) = 10$ mm/s) and high tool velocity ($v(t) = 100$ mm/s). The motion starts at the point and is directed downwards. The tool reference path (reference) is drawn together with the actual tool behaviour measured by length gauges (measured). The deviation from the reference is larger in b) compared to the deviation shown in a). It can be explained both by larger motor control errors and by dynamic effects in the robot structure.*

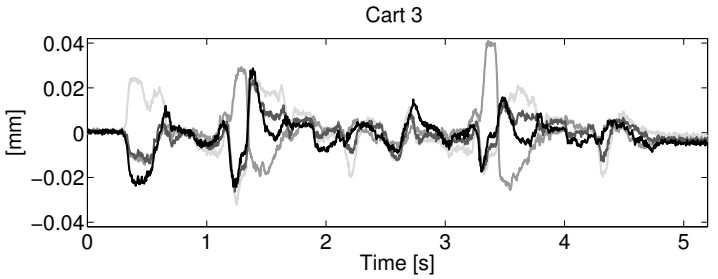


Figure 8.4: *The difference* (8.1) *for cart 3, for five identical experiments illustrates the repeatability of the system.*

8.4 Robot models

Previous work on dynamic modelling of the Gantry-Tau robot includes rigid-body dynamic modelling in Dressler et al. [2007b] and black-box identification of flexible dynamics in Cescon et al. [2009]. Identification and dynamic modelling of a slightly different Gantry-Tau prototype of the same size as the robot used in this chapter is presented in for example Tyapin et al. [2002] and Hovland et al. [2007]. While Tyapin et al. [2002] calculated the lowest mechanical resonance frequency of the Gantry-Tau robot to be larger than 50 Hz, it was measured to lie around 14 Hz in Cescon et al. [2009]. The experimental results presented in this thesis show a notable flexible behaviour, which implies that a rigid-body model is not sufficient. Most of the flexibilities are assumed to lie in the carbon fibre links and the framework, as the newly developed spherical joints are very stiff and a comparison of measured motor angular positions and cart positions show considerably less flexibility than the tool motion. The serial wrist has been added since the work presented in Cescon et al. [2009], which has changed the dynamic properties of the robot. Therefore, a new modelling needs to be done.

As the trajectory is confined to a small part of the workspace, see Figure 8.2, local linear models are identified. In Figure 8.5, a schematic picture is given of the robot system and the signals used in the identification. The motor angular position reference $r_m(t)$ is derived from the tool-position reference $r(t)$ by using the inverse kinematics. A calculated tool position $z_c(t)$ is derived from measured motor angular positions $q_m(t)$ transformed by the forward kinematics. The measured tool position is denoted $z(t)$. The relation between the identified models is illustrated in Figure 8.6. The first section presents the identified SISO models for each of the carts, with motor angular position reference $r_m(t)$ as input and measured motor angular position $q_m(t)$ as output. The models are used for tuning the ILC algorithm using motor angular position measurements. Thereafter modelling and identification of the complete robot structure is summarised. The pulse-response experiments in Cescon et al. [2009] are repeated to update information about the resonance frequencies of the robot system. Furthermore, models used for tuning and analysis of the ILC algorithm using tool position and for

estimating the tool motion are described. It should however be emphasised that the linear models derived are only valid for a small operating range due to the highly nonlinear robot system. The models are therefore to be used as a guidance for the design of the ILC filters, while the final choice is made by experimental tuning. The main modelling focus for the models used in the ILC filter design is to capture the main resonance frequencies of the system. Future work includes an extensive modelling of the nonlinear properties of the system, including the entire flexible dynamics, friction and backlash.

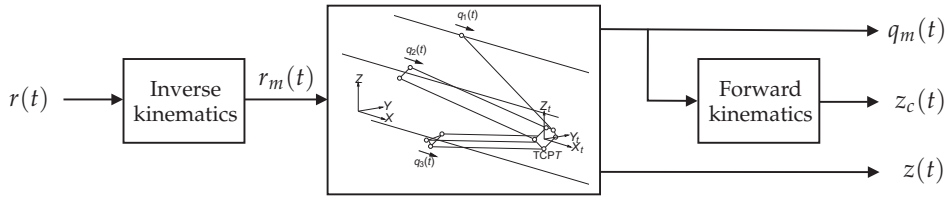


Figure 8.5: *Robot system with signals used in the identification. The motor angular position reference $r_m(t)$ is derived from the tool-position reference $r(t)$ using the inverse kinematics. A calculated tool position $z_c(t)$ is derived from measured motor angular position $q_m(t)$ transformed by the forward kinematics. The measured tool position is denoted $z(t)$.*

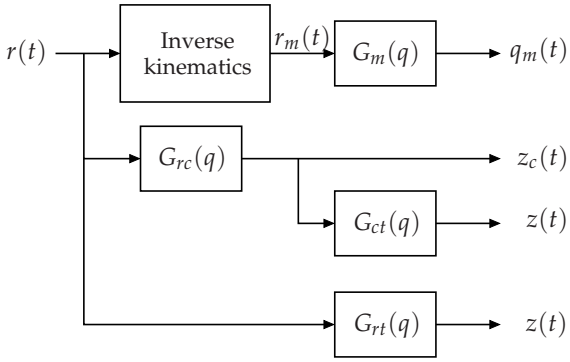


Figure 8.6: *Illustration of the relation between the identified models.*

8.4.1 Motor models

Identification experiments are performed, where linear SISO models with motor angular position reference $r_m(t)$ as input and measured motor angular position $q_m(t)$ as output are derived for the three carts. In order to disregard the nonlinear coupling effects between the carts, step-response experiments are performed with all carts moving simultaneously, giving a resulting motion in the X-direction, see Figure 8.1. This procedure facilitates obtaining linear SISO models, but it also limits the validity of the identified models for other movements involv-

ing coupling effects. The amplitude of the steps are chosen as 1 mm, 1.5 mm and 2 mm, respectively, motivated by the sensor limitations.

A model structure with few parameters is sufficient for analysis and design of the ILC algorithm using motor angular positions, since the chosen design approach only requires the static gain and delay of the system, see Section 8.8. Therefore a first-order ARX model structure [Ljung, 1999] is identified from data, using the System Identification Toolbox in MATLAB [Ljung, 2010]. The resulting models for the three carts are given by

$$G_{m,1}(q) = q^{-5}\frac{0.03527}{1-0.9647q^{-1}} \tag{8.2a}$$

$$G_{m,2}(q) = q^{-5}\frac{0.03425}{1-0.9656q^{-1}} \tag{8.2b}$$

$$G_{m,3}(q) = q^{-5}\frac{0.02748}{1-0.9728q^{-1}} \tag{8.2c}$$

with the sampling interval $T_s = 4\,\text{ms}$. Results from validation of the models can be seen in Figure 8.7, using validation data from the 1.5 mm step-response experiment. However, even if different data are used for identification and validation, still the same type of movement is used. Validation with data from the rectangular high-velocity motion shows that the identified models (8.2) have too low bandwidth concerning this type of movement, where also stronger coupling between the motors appear. Still, the system has the same time delay and static gain.

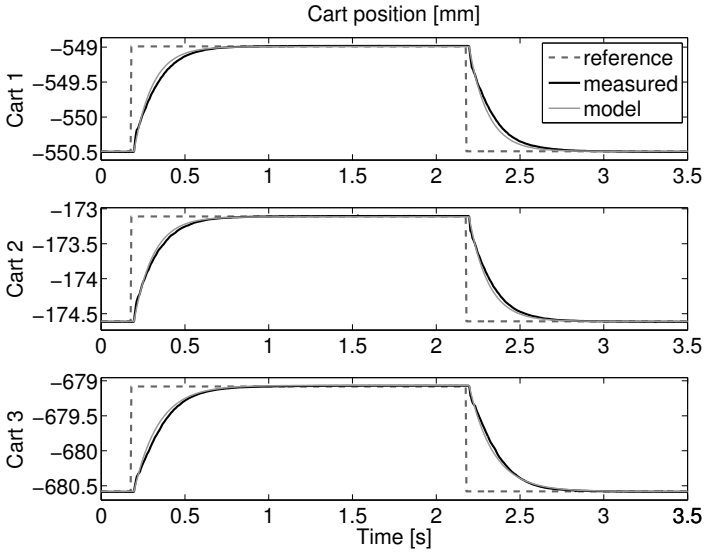


Figure 8.7: *Model validation of the cart positions. Output (model) from the motor models* (8.2) *for the three carts, validated using data (measured) from the step-response experiments.*

The five samples delay seen in (8.2) is caused by internal data communication in the IRC5 system. Linear models performing well for all kinds of movements are very difficult to find, but the identified models still fulfill their purpose for tuning the ILC algorithm using measured motor angular positions.

8.4.2 Models of the complete robot structure

Resonance frequency

Pulse-response experiments are performed to determine the resonance frequencies of the robot including the serial wrist. A force pulse is generated by the impact of a hammer on the end-effector plate in the X- and Y-direction, respectively. The resulting tool position is measured by the linear gauges in the X- and Y-direction, from which the tool vibration can be determined by inspection in the time domain. When the force is applied in the X-direction, a resonance of 7.4 Hz is observed in the X-direction. The tool vibration is between 11.4 and 11.9 Hz in the Y-direction, which after a few oscillations turns into a 7.4 Hz vibration. When the force is applied in the Y-direction, the resonance frequency in the X-direction is around 10.5 Hz and in the Y-direction around 11.5 Hz, see Table 8.1. The resonance in the Y-direction is larger in magnitude than the resonance in the X-direction and consequently dominant for the robot behaviour.

Table 8.1: *Results from force pulse-response experiments. The table shows the resulting robot tool vibration in the X- and Y-direction, with the resonance in the Y-direction being dominant for the robot behaviour.*

		Force-pulse direction	
		X	Y
Tool vibration [Hz]	X	7.4	10.5
	Y	11.4 – 11.9 7.4	11.5

Linear black-box models

Different linear black-box models have been identified for usage in the ILC algorithm and for Kalman filter design. As only two-dimensional measurements are available, the identified MIMO models include only the XY-components of the tool position. In the identification experiments, a pseudo random binary sequence (PRBS) signal with an amplitude of 1 mm is added to each of the motor angular position references. As it is difficult to obtain a model performing well enough for estimation, identification is also made on data from experiments for the rectangular trajectory with high-velocity motion. A schematic picture of the robot system with the signals used in the identification is given in Figure 8.5. The relation between the identified models is illustrated in Figure 8.6.

First, a model from tool-position reference $r(t)$ to measured tool position $z(t)$ is identified from PRBS data. The model can be described by the relation

$$z(t) = G_{rt}(q)\, r(t) \tag{8.3}$$

where the subscripts r and t denote "reference" and "tool", respectively. The identification procedure aims at identifying a model capturing the resonance frequencies of the robot system, since the model is used for design of the ILC algorithm using measured tool position[2].

Thereafter a model is identified with input $r(t)$ and output $z_c(t)$,

$$z_c(t) = G_{rc}(q)\, r(t) \tag{8.4}$$

where $z_c(t)$ is the calculated tool position from measured motor angular positions transformed by the forward kinematics. The subscript c is introduced for "calculated". This model is used for stability analysis of the ILC algorithms using estimates of the robot tool position[3].

A state-space model with calculated tool position $z_c(t)$ as input and measured tool position $z(t)$ as output is identified, to be used in the estimation of the robot tool position[4]. The model can be written

$$z(t) = G_{ct}(q)\, z_c(t) \tag{8.5}$$

The model is identified from data when following the rectangular trajectory with high tool velocity, since it is difficult to obtain a model performing well enough for estimation from PRBS data.

8.5 ILC algorithms

For analysis, the descriptions of the system and ILC algorithm introduced in Chapter 6 are used, and are shortly summarised here for convenience.

A stable system is studied, given by the following discrete-time description

$$y_k(t) = T_{ry}(q)r(t) + T_{uy}(q)u_k(t) \tag{8.6a}$$

$$z_k(t) = T_{rz}(q)r(t) + T_{uz}(q)u_k(t) \tag{8.6b}$$

with measured variable $y_k(t)$, controlled variable $z_k(t)$ and reference $r(t)$ to be followed by the controlled variable. An ILC algorithm of the type

$$u_{k+1}(t) = Q(q)\Big(u_k(t) + L(q)\epsilon_k(t)\Big) \tag{8.7}$$

is applied to the system (8.6), using the error

$$\epsilon_k(t) = r(t) - \hat{z}_k(t) \tag{8.8}$$

[2] See Section 8.8.3.
[3] Described in Section 8.8.2.
[4] See Section 8.6 for the details.

The estimate $\hat{z}_k(t)$ of the controlled variable is described by (6.6),

$$\hat{z}_k(t) = F_r(q)r_z(t) + F_u(q)u_k(t) + F_y(q)y_k(t) \tag{8.9}$$

with the stable filters $F_r(q)$, $F_u(q)$ and $F_y(q)$. Now, expressing (8.6) to (8.9) in matrix form, and if

$$\bar{\sigma}\Big(\boldsymbol{Q}\big(I - \boldsymbol{L}(\boldsymbol{F}_u + \boldsymbol{F}_y\boldsymbol{T}_{uy})\big)\Big) < 1 \tag{8.10}$$

then the ILC system is stable and $\boldsymbol{u}_k$ converges monotonically to the asymptotic ILC input $\boldsymbol{u}_\infty$, according to Theorem 6.2.

Implementation

The filter $L(q)$ in (8.7) is chosen as $L(q) = \gamma q^\delta$ in the experiments, and is implemented by extending the error signal using the last value. The filter $Q(q)$ is chosen as a non-causal filter with zero-phase characteristics. This is implemented by filtering the signal forwards and backwards in time through a causal filter $\bar{Q}(q)$ with no extension of the signal[5]. The filter $\bar{Q}(q)$ is designed for each specific experiment.

The resulting ILC input signal $u_k(t)$ is applied to a system where the reference trajectory to be learned (learning part of the trajectory) is preceded and followed by lead-in and lead-out parts. Therefore, the ends of the ILC input signal are weighted in the time domain by a vector

$$\boldsymbol{w} = \begin{pmatrix} w(1) & \dots & w(n) \end{pmatrix}^T \tag{8.11}$$

with n elements to give a smooth transition between the learning part and the lead-in/lead-out parts of the trajectory. A vector with $n = 100$ elements is used in the experiments, generated by the MATLAB command `tukeywin(2*n, 0.95)`, to give a flat curve at the ends and a smooth increase of the weighting coefficients in between, see Figure 8.8.

Time-domain weighting of the ILC input signal has previously been discussed in for example van Oosten et al. [2004], where however only zero or identity weights are used to choose actuation and observation intervals for the ILC algorithm in a point-to-point application. Experiments on a flexible beam in point-to-point motions are presented in van de Wijdeven and Bosgra [2007], where also non-diagonal weighting matrices with arbitrary elements are used. The idea of smoothing the input signal of an RC algorithm is evaluated by simulations in Chen and Longman [1999], where the learning input signals are applied in a smooth manner and at an appropriate time. The resulting learning rate and final error level are improved by gradually converting from one control action to the next over 12 time steps with a time weighting linear in time or determined from cubic splines. To the best knowledge of the author, no experimental results have been published previously in an ILC application where the trajectory to be followed has lead-in/lead-out.

[5] This means that implementation alternative B-I is used, see Chapter 9.

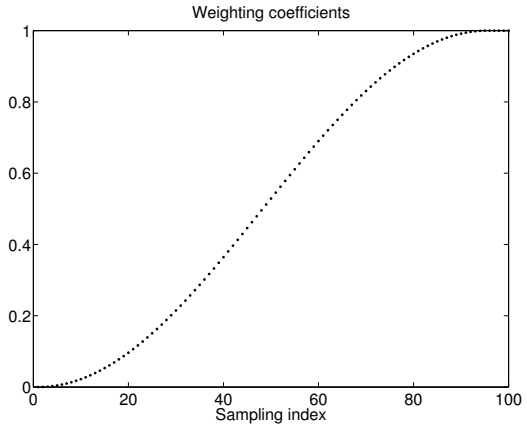


Figure 8.8: *Coefficients for time-domain weighting of the ends of the ILC input signal.*

8.6 Estimation of robot tool position

Two approaches for estimating the robot tool position will be evaluated:

- Complementary filtering.
- Kalman filtering.

Only the XY-direction is considered, motivated by the measurements available for evaluation.

Complementary filtering For the system with moderate load considered in this chapter, experimental results show that the calculated tool position $z_c(t)$ is a reasonable tool-position estimate for low frequencies. The signal $z_a(t)$, obtained by double integration of the measured tool acceleration $a(t)$, is a reasonably accurate tool-position estimate for higher frequencies. These two signals can be fused together by a complementary filter[6], giving the tool-position estimate

$$\hat{z}(t) = G(q)z_c(t) + \Big(1 - G(q)\Big)z_a(t) \tag{8.12}$$

In the experiments, the filter $G(q)$ is chosen as a second-order low-pass Butterworth filter, applied to give zero-phase characteristics by using `filtfilt` in MATLAB. The cutoff frequency of the filter is tuned to give the smallest maximum estimation error $e(t) = z(t) - \hat{z}(t)$ for the rectangular motion.

It can be noted that a similar approach with double integration and high-pass filtering of the measured tool acceleration is used in Nordström [2006]. Evaluation in both simulations and experiments conclude improved feedback control of a commercial industrial robot using the accelerometer as an additional sensor.

[6] See Section 3.2.3 for the details.

Kalman filtering The tool position is estimated using a stationary Kalman filter[7], based on the linear model (8.5) with input $z_c(t)$ and output $z(t)$. The predicted position is numerically differentiated twice, and the tool position is estimated by a stationary Kalman filter using the calculated tool position $z_c(t)$ and the measured tool acceleration $a(t)$. The covariance matrix for the process noise is chosen as $R_v = \alpha I$, with the factor α determined by minimising the maximum estimation error for the rectangular motion. The covariance matrix R_w for the measurement noise is determined from experiments.

In Figure 8.9, the estimation error $e(t) = z(t) - \hat{z}(t)$ is shown for the rectangular motion, with the measured tool position $z(t)$. It can be seen that the estimation error is decreased when using the complementary filter estimate (8.12) compared to using the estimate $\hat{z}(t) = z_c(t)$, especially in the Y-direction. This can be explained by the fact that the robot is flexible in this direction and thereby the accelerometer mounted on the end-effector plate provides more information than for the stiffer X-direction. The Kalman filter estimate performs slightly better than the complementary filter estimate. It is likely that the quality of the tool-position estimate can be further increased by using a better model for the specific working point in for example an EKF.

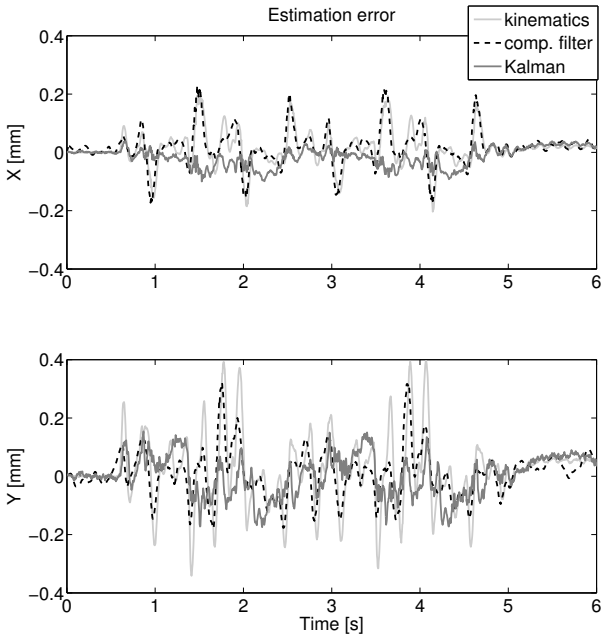


Figure 8.9: *Estimation error $e(t) = z(t) - \hat{z}(t)$ for the rectangular motion. Result for the calculated tool position $z_c(t)$ from motor angular positions transformed by the forward kinematics (kinematics) compared to results from the complementary filter (comp. filter) and the Kalman filter (Kalman).*

[7] See Algorithm 2 in Chapter 3.

8.7 Conditions for ILC experiments

In the experiments, the rectangular motion is used with a programmed tool velocity of $v(t) = 100\,\text{mm/s}$. The experiments are categorised into the following cases with the ILC algorithm (8.7) using the following errors:

1) Measured motor angular position, giving $\epsilon_k(t) = r_m(t) - q_m(t)$.
2) Estimated tool position, giving $\epsilon_k(t) = r(t) - \hat{z}_k(t)$.
3) Measured tool position, giving $\epsilon_k(t) = r(t) - z_k(t)$.

Case 2 is thereafter divided into:

A) Estimate $\hat{z}_k(t)$ derived using complementary filter.
B) Estimate $\hat{z}_k(t)$ derived using Kalman filter.

ILC is only performed in the XY-direction due to the experimental setup, and the error in the Z-direction is neglected. For the same reason, the tool performance is evaluated in the XY-plane. Before each iteration the motors are driven to their initial positions to minimise the effect of initial state errors.

For all experiments, the filter $L(q)$ in (8.7) has a time shift of $\delta = 5$ samples and a learning gain of $\gamma = 0.9$, based on knowledge of time delay and static gain of the closed-loop system. The filter $Q(q)$ is chosen differently for each specific case.

8.7.1 Evaluation measures

The experimental evaluation is based on the nominal error when no ILC algorithm is applied ($k = 0$). The vector of nominal errors for the motor angular positions for the three carts is given by

$$\begin{pmatrix} \boldsymbol{e}^1_{m,0} \\ \boldsymbol{e}^2_{m,0} \\ \boldsymbol{e}^3_{m,0} \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}^1_m \\ \boldsymbol{r}^2_m \\ \boldsymbol{r}^3_m \end{pmatrix} - \begin{pmatrix} \boldsymbol{q}^1_{m,0} \\ \boldsymbol{q}^2_{m,0} \\ \boldsymbol{q}^3_{m,0} \end{pmatrix} \tag{8.13}$$

with $\boldsymbol{e}^i_{m,0}$ denoting the motor angular position error and $\boldsymbol{q}^i_{m,0}$ being the measured motor angular position for cart i at iteration 0 and $\boldsymbol{r}^i_m$ being the corresponding reference. The vector of nominal tool-position errors is analogously defined as

$$\begin{pmatrix} \boldsymbol{e}^X_{z,0} \\ \boldsymbol{e}^Y_{z,0} \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}^X \\ \boldsymbol{r}^Y \end{pmatrix} - \begin{pmatrix} \boldsymbol{z}^X_0 \\ \boldsymbol{z}^Y_0 \end{pmatrix} \tag{8.14}$$

in the X- and Y-direction.

The reduction of the 2-norm of the motor angular position error at iteration k for

each of the three carts is given in percentage of the nominal error (8.13), as in

$$\bar{e}^1_{m,k} = 100 \cdot \frac{\|\boldsymbol{e}^1_{m,k}\|_2}{\|\boldsymbol{e}^1_{m,0}\|_2} \quad [\%] \tag{8.15a}$$

$$\bar{e}^2_{m,k} = 100 \cdot \frac{\|\boldsymbol{e}^2_{m,k}\|_2}{\|\boldsymbol{e}^2_{m,0}\|_2} \quad [\%] \tag{8.15b}$$

$$\bar{e}^3_{m,k} = 100 \cdot \frac{\|\boldsymbol{e}^3_{m,k}\|_2}{\|\boldsymbol{e}^3_{m,0}\|_2} \quad [\%] \tag{8.15c}$$

The reduction of the tool error is similarly normalised by (8.14), resulting in

$$\bar{e}^X_{z,k} = 100 \cdot \frac{\|\boldsymbol{e}^X_{z,k}\|_2}{\|\boldsymbol{e}^X_{z,0}\|_2} \quad [\%] \tag{8.16a}$$

$$\bar{e}^Y_{z,k} = 100 \cdot \frac{\|\boldsymbol{e}^Y_{z,k}\|_2}{\|\boldsymbol{e}^Y_{z,0}\|_2} \quad [\%] \tag{8.16b}$$

The quantities (8.15) and (8.16) are in the evaluation of the experimental results based on the part of the trajectory to be corrected by the ILC algorithm. See Section 8.3.1 for how the trajectory is divided into a learning part and lead-in/lead-out parts.

8.8 Experimental results

First, the resulting tool performance is investigated for case 1. This result is then compared to the result when the ILC algorithm uses an estimate of the tool position, case 2, with estimation alternative A and B. Finally, the result when the measured tool position is used in the ILC algorithm, case 3, is discussed.

8.8.1 Case 1 — ILC using measurements of motor angular position

For low velocities, most of the components of the tool-position error can be corrected by an ILC algorithm using motor angular positions, since the influence of the dynamic effects in the robot structure is small, see Figure 8.3a. This can be regarded as if the ILC algorithm is based on the estimate $\hat{z}(t) = z_c(t)$, the motor angular positions transformed by the forward kinematics to a calculated tool position, which is a reasonable estimate for low-velocity motions and moderate load. When performing ILC experiments for a motion with higher velocity, a smaller error reduction (8.16) of the robot tool is expected than for low-velocity motions. This can be explained by dynamic effects introduced in the system[8] due to the robot structure containing mechanical flexibilities, as is discussed in detail in Chapter 5.

[8] See Figure 8.3b.

For case 1, an ILC algorithm is implemented independently for each motor. The same ILC design variables are applied to all motors, since the models (8.2) of the motor dynamics for the carts are close to each other. The causal filter $\bar{Q}(q)$ is chosen as a second-order low-pass Butterworth filter with cutoff frequency $f_c = 10\,\text{Hz}$, and is implemented to give a zero-phase filter $Q(q)$ as described in Section 8.7. The ILC input signal is added to the motor angular position reference, giving $T_{uy}(q)$ in (8.6) equal to $G_{m,i}(q)$ from (8.2) for the three motors $i = 1, 2, 3$. Using the measured motor angular positions in the ILC update equation results in $F_r(q) = F_u(q) = 0$, $F_y(q) = 1$ in (8.9). The criterion (8.10) for motor i is given by

$$\bar{\sigma}_{i,\text{case 1}} = \bar{\sigma}\Big(\boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_{uy})\Big) < 1$$

For the three motors it results in

$$\bar{\sigma}_{1,\text{case 1}} \approx 0.90, \quad \bar{\sigma}_{2,\text{case 1}} \approx 0.91, \quad \bar{\sigma}_{3,\text{case 1}} \approx 0.93$$

This analysis results in stability and monotone convergence when each of the three motor models (8.2) is controlled independently by the ILC algorithm. From these values, it can also be seen that there is a robustness margin to model errors.

Some results from the experimental investigation of the properties of the algorithm have already been shown in Chapter 5, and are therefore only briefly summarised here. The reader is directed to Chapter 5 for the relevant diagrams. In Figure 5.12 it can be seen that the error measure (8.15) for each motor is reduced to nearly 2 % after only five iterations. The large error reduction can also be concluded from Figure 5.13, showing small remaining motor angular position errors compared to the nominal errors. In Figure 5.14 it is illustrated that the measured tool position has a larger deviation from the tool-position reference compared to what is the case for the calculated tool position $z_c(t)$, as is expected due to the flexible robot structure. Still, the tool performance is improved compared to the nominal tool performance shown in Figure 8.3b.

The effect of the time-weighting of the ILC input signal, discussed in Section 8.5, can also be seen in Figure 5.13. The curves illustrate how the learning is smoothly increased until the learning part starts ($t = 1.6\,\text{s}$) and is smoothly decreased at the start of the lead-out part ($t = 4.1\,\text{s}$).

Repeatability

To examine the repeatability properties, the same ILC experiment is repeated five times under as identical conditions as possible. An ILC input signal $u_1(t)$ to the first iteration is calculated from the nominal motor angular position errors. The signal is applied to the system and the iterations $k = 1, \ldots, 5$ are performed. This is repeated five times with the same input signal $u_1(t)$ at the first iteration. The error measure (8.15) for each motor and iteration is shown in Figure 8.10 for the five repeated experiments. It can be seen that the behaviour with slightly non-monotone convergence in Figure 5.12 may be explained by this spread of the error measure (8.15) for each iteration. The spread of the error measure (8.15) may be even larger due to varying experimental conditions, for example temperature.

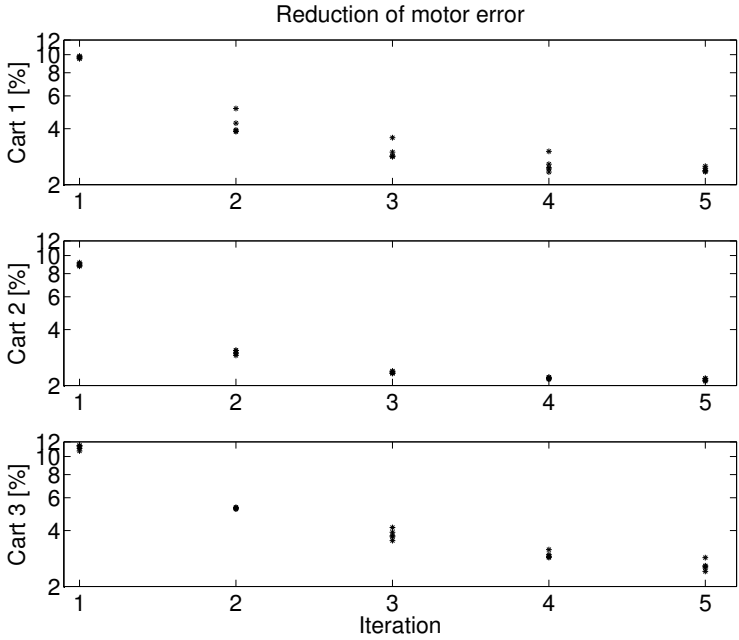


Figure 8.10: *Case 1: Error measure* (8.15) *for each cart when performing five experiments with identical ILC input signal at first iteration.*

8.8.2 Case 2 — ILC using estimates of tool position

Next, results are compared for experiments with an ILC algorithm using tool-position estimates from the complementary filter, case 2A, and the Kalman filter, case 2B. From the identification experiments discussed in Section 8.4.2 it is concluded that the robot system has a distinct resonance frequency of around 11.5 Hz in the Y-direction and not so pronounced resonance frequencies at approximately 7.4 Hz and 10.5 Hz in the X-direction. To be able to learn error components around and above the dominating resonance frequencies of the system, it is not sufficient to simply choose a low-pass filter $Q(q)$ with cutoff frequency below the lowest resonance frequency. Instead, the filter design has to be based on a model from the input point of the ILC input signal, here the reference $r(t)$, to the estimate $\hat{z}_k(t)$ used in the algorithm. For case 2A, the tool-position estimate is derived from (8.12) by using complementary filtering, which gives

$$\begin{aligned}\hat{z}_k(t) &= \begin{pmatrix} G(q) & 1-G(q) \end{pmatrix} \begin{pmatrix} z_{c,k}(t) \\ z_{a,k}(t) \end{pmatrix} = \begin{pmatrix} G(q) & 1-G(q) \end{pmatrix} \begin{pmatrix} G_{rc}(q) \\ G_{rt}(q) \end{pmatrix} r(t) \\ &= F_y(q) T_{uy}(q) r(t)\end{aligned}$$

from the measurement vector consisting of $z_{c,k}(t)$ and $z_{a,k}(t)$, with the representation (8.9) of the estimate and the system description (8.6). The measurement $z_{c,k}(t)$ relates to the reference $r(t)$ by the model (8.4). $z(t)$ is obtained

from $r(t)$ by (8.3), where numerical double differentiation followed by numerical double integration gives $z_{a,k}(t)$.

Case 2B is treated similarly with the measurement vector consisting of $z_{c,k}(t)$ and $a_k(t)$,

$$\hat{z}_k(t) = F_y(q)\begin{pmatrix} z_{c,k}(t) \\ a_k(t) \end{pmatrix}$$

where the estimation filter $F_y(q)$ is given from the relations for the Kalman estimator, similar to the observer in Example (6.5).

The tuning of the diagonal filter $Q(q)$ aims at being robust to large model errors especially around the resonance frequencies. The filter is obtained by filtering the signal forwards and backwards in time through a causal filter $\bar{Q}(q)$. The filter $\bar{Q}(q)$ is designed from the magnitude of the X- and Y-elements of the relation $1 - L(q)\Big(F_u(q) + F_y(q)T_{uy}(q)\Big)$, depicted in Figure 8.11. Due to high-frequency measurement noise, learning of the error components up to 30 Hz is chosen, which is above the dominating resonance frequencies of the system. Therefore, $Q(q)$ is also designed to have large attenuation for frequencies above 30 Hz. In Figure 8.11 the robustness properties of the algorithm around the resonance frequencies of the system can be seen, together with the attenuation for higher frequencies (over 30 Hz). The choice of filter $Q(q)$ is then experimentally evaluated for cases 2A and 2B to give a good error reduction, with the same filter

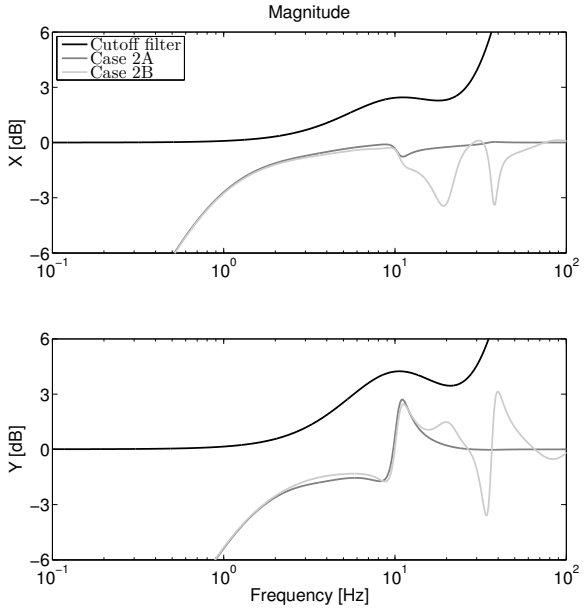


Figure 8.11: *Case 2A, 2B: The magnitude $|\bar{Q}^{-1}(e^{i\omega T_s})|$ of the filter $\bar{Q}(q)$ cutting off the learning over about 30 Hz illustrated together with the magnitude $|1 - L(e^{i\omega T_s})(F_u(e^{i\omega T_s}) + F_y(e^{i\omega T_s})T_{uy}(e^{i\omega T_s}))|$ in the X- and Y-direction.*

used for both cases. The criterion (8.10) then results in

$$\bar{\sigma}_{\text{case 2A}} \approx 0.86$$
$$\bar{\sigma}_{\text{case 2B}} \approx 0.93$$

with robustness to model errors.

The behaviour of the ILC algorithm applied to the system is then evaluated experimentally. By comparing the resulting tool path shown in Figure 8.12 to the result for case 1 shown in Figure 5.14, it is seen that the tool performance after 10 iterations are improved for case 2A compared to case 1. Similar results are achieved for case 2B, and the corresponding figure is omitted here. The same conclusion of smaller tool-position error for cases 2A and 2B compared to case 1 can be made from Figure 8.13, where the error measure (8.16) of the tool position can be seen for all cases. For illustration, the mean of the error reduction (8.16) for iteration 5 to 10 is given in Table 8.2. Case 2A seems to give a slightly larger error reduction than case 2B. Since different ILC algorithms are tuned and applied to the robot, it is difficult to compare the results quantitatively. Another tuning could for example give a larger error in the X-direction, maybe resulting in smaller error in the Y-direction. However, the resulting overall behaviour is an improved tool performance for cases 2A and 2B compared to case 1, especially in the Y-direction. Since the robot is stiff in the X-direction compared to the Y-direction, most of the errors in the X-direction can be compensated already by using motor angular position measurements. The accelerometer signal gives additional information about the tool position in the Y-direction compared to when only measuring motor angular positions, which makes it possible to improve the performance in that direction when using tool-position estimates in the ILC algorithm.

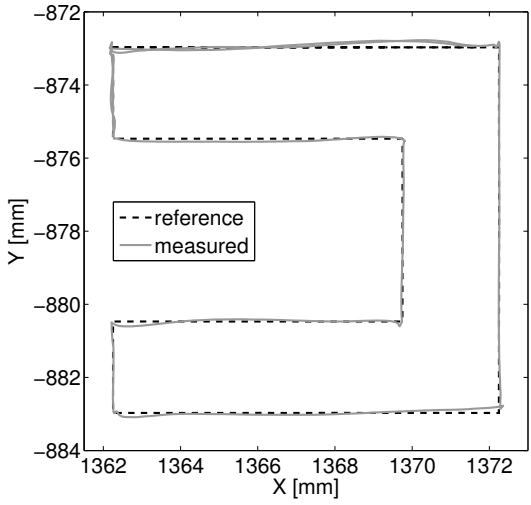


Figure 8.12: *Case 2A: Tool performance after 10 iterations, with reference path compared to measured tool path.*

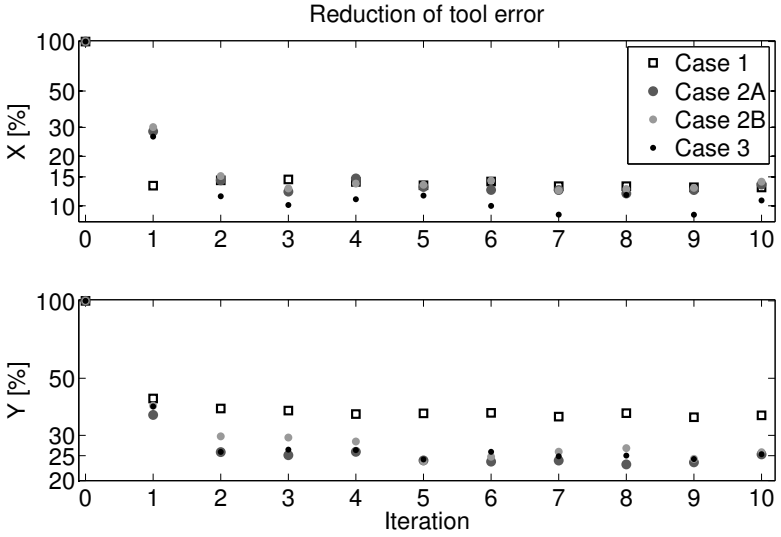


Figure 8.13: *Comparison of error measure* (8.16) *of the robot tool position for the cases 1, 2A, 2B, and 3 in the X- and Y-direction, respectively..*

Table 8.2: *Comparison of mean value for iteration 5 to 10 of error measure* (8.16) *of the robot tool position for the cases 1, 2A, 2B, and 3 in the X- and Y-direction, respectively.*

	Case 1	Case 2A	Case 2B	Case 3
X	13.3 %	12.6 %	13.3 %	10.3 %
Y	36.1 %	23.9 %	25.2 %	24.9 %

8.8.3 Case 3 — ILC using measurements of tool position

Finally, the ILC algorithm uses the measured tool position in the XY-direction. This case illustrates the improvement in the (ideal) case when measurements of the robot tool position are available for usage in the ILC algorithm. The design of the filter $\bar{Q}(q)$ follows the reasoning for case 2, with a need for a filter giving an algorithm robust to large model errors and with learning up to 30 Hz to avoid high-frequency measurement noise to be included in the learning. For case 3, with the ILC input signal added to the reference $r(t)$, it gives $T_{uy}(q) = G_{rt}(q)$ in (8.3) and $F_r(q) = F_u(q) = 0$, $F_y(q) = 1$ in (8.9). The model-based design of $\bar{Q}(q)$ is followed by experimental fine-tuning to give a stable ILC system having good error reduction. The resulting filter $\bar{Q}(q)$ is identical to the filter for case 2 in the Y-direction, while the gain around the resonance frequency is only slightly larger in the X-direction. The resulting magnitude diagrams in the X-and Y-directions are very similar to the diagrams for case 2 shown in Figure 8.11. This is natural,

since the relation $\big(1 - G(q)\big)G_{rt}(q)$ is close to $G_{rt}(q)$ for higher frequencies. The corresponding figure is therefore omitted here. The criterion (8.10) results in

$$\bar{\sigma}_{\text{case 3}} \approx 0.95$$

The resulting error measure (8.16) of the tool is shown in Figure 8.13 and the mean for iterations 5 to 10 is given in Table 8.2. It can be seen that the error reduction (8.16) is slightly larger for case 3 compared to case 2 in the Y-direction. As mentioned earlier, it is difficult to compare the experimental results quantitatively. It is for example possible that there is a more aggressive tuning of the ILC algorithm using the measured tool position that results in a larger error reduction for case 3. Still, it can be concluded that an ILC algorithm using estimates of the tool position is able to improve the tool performance compared to an ILC algorithm using measured motor angular positions directly.

8.9 Conclusions

In this chapter, three different approaches to ILC have been experimentally evaluated when applied to the Gantry-Tau parallel robot. First, the ILC algorithm uses measured motor angular positions, which usually are the only measurements available in commercial industrial robot systems. Second, the ILC algorithm uses tool-position estimates. The estimates are obtained by using complementary and Kalman filtering, based on motor angular position measurements transformed by the forward kinematics and tool-acceleration measurements. Third, the ILC algorithm uses the measured tool position. This approach is for evaluation purposes only, as it is difficult to measure the actual tool position in industrial applications. The ILC filters are designed to enable learning error components also above the dominant resonance frequencies of the robot system. Linear models identified in the specific working point is used for tuning of the ILC filters and analysis of the resulting stability.

In the chapter some practical aspects have also been discussed. The different ILC approaches are evaluated on a high-velocity rectangular trajectory, preceded and followed by lead-in/lead-out parts of the trajectory. Therefore, the beginning and end of the ILC input signal are weighted in the time domain to give a smooth transition from the lead-in/lead-out parts to the parts of the trajectory to be learned.

From the experiments it can be concluded that an ILC algorithm using estimates of the robot tool position improves the tool performance compared to an ILC algorithm using the measured motor angular positions directly. The experimental evaluation illustrates the principle of estimation-based ILC. Better robot models might improve the results, due to a more efficient model-based design of the ILC filters and more accurate tool-position estimates.

9

Implementation aspects

Some implementation aspects of ILC are considered in this chapter. An ILC algorithm involves filtering of various signals over finite-time intervals, often using non-causal filters. It is therefore important that the boundary effects of the filtering operations are handled in an appropriate way when implementing the ILC algorithm. It is illustrated in both theoretical analysis, using a matrix description in the time domain of the system and ILC algorithm, and in simulations of a two-mass system that the method of implementation for handling the boundary effects can have large influence over stability and convergence properties of the ILC algorithm. The main reference for the chapter is Wallén et al. [2010].

9.1 Introduction

The purpose of the chapter is twofold. First, to study some implementation alternatives for handling the boundary effects that occur in the filtering operations in the ILC algorithm. Second, to show how to systematically analyse their effects on the performance. The analysis is performed in the time domain by involving the implementation aspects of the filtering in the matrix formulation of the ILC algorithm.

The topic regarding implementation of ILC algorithms appears to have received fairly limited attention in the literature, even though it can be of large practical importance. Some comments are given in the literature on implementation aspects and boundary effects, for example a short note about extending the finite-time error signal with the last value at the end [Moore, 1998a, Wang and Zhang, 2009] or briefly discussing extensions at the beginning and the end of the signal before filtering [Elci et al., 2002]. However, the group around Longman has

studied various aspects of ILC/RC, including filtering and implementation. An overview of many of the ideas is given in Longman [2000], including references to relevant publications. One of the most important publications related to the results presented in this chapter is Plotnik and Longman [1999]. In the paper different aspects of filtering are discussed in detail; which type of filter to use, which signal to filter and how to perform the filtering when implementing the ILC algorithm. The resulting performance when using the different filtering and extension alternatives are then compared in simulations of a model of a single robot joint controlled using an ILC algorithm. The subject of implementation is also discussed in Longman and Songchon [1999]. There it is assumed that the boundary effects can be ignored, so that it is possible to analyse the ILC updating law in the frequency domain. In the publications mentioned above the effects of the filtering operations are discussed with focus on zero-phase filtering of the signal in order to cut off the learning above a desired frequency. The problem of initial conditions in the zero-phase filtering operation is highlighted, and it is discussed how to perform the filtering so that the choice of these initial values has negligible influence on the resulting behaviour. Stability and monotone convergence properties of the ILC algorithm are analysed in the frequency domain, since the effects of the filtering operations are made as small as possible so that frequency-domain analysis can be applied. This can be contrasted with the work in this chapter, where the main objective is to investigate the actual effects of the chosen implementation, with the analysis performed in the time domain.

9.2 System and ILC algorithm

The types of systems and ILC algorithms considered in this chapter are given here, together with a short repetition of some convergence results that will be used in the analysis[1].

9.2.1 System description

Consider the linear discrete-time system

$$y_k(t) = T_r(q)r(t) + T_u(q)u_k(t) \tag{9.1}$$

where load and measurement disturbances are omitted for simplicity. All signals are defined on a finite time interval $t = nT_s$, $n \in [0, N-1]$ with N number of samples. The sampling interval $T_s = 1$ is used for brevity in this chapter, if nothing else is stated. Parallel to the system description (9.1) in filter form, the matrix description[2] is used. The system description (9.1) is then reformulated into

$$\boldsymbol{y}_k = \boldsymbol{T}_r \boldsymbol{r} + \boldsymbol{T}_u \boldsymbol{u}_k \tag{9.2}$$

with the lower-triangular Toeplitz matrices $\boldsymbol{T}_r$ and $\boldsymbol{T}_u$ derived from the pulse-response coefficients of the causal transfer operators $T_r(q)$ and $T_u(q)$.

[1] The reader is directed to Chapter 4 for a detailed description.
[2] See Section 4.4 for the details.

9.2.2 ILC algorithm

Considering linear discrete-time ILC algorithms, two main alternative ways to implement a particular algorithm are discussed in the thesis. The first alternative is the matrix formulation (4.13), in which the ILC input signal vector for the next iteration is computed according to the expression

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}(\boldsymbol{u}_k + \boldsymbol{L}\boldsymbol{e}_k) \tag{9.3}$$

where the error used in the algorithm is given by

$$\boldsymbol{e}_k = \boldsymbol{r} - \boldsymbol{y}_k \tag{9.4}$$

and where $\boldsymbol{Q}$ and $\boldsymbol{L}$ are $N \times N$ matrices, not necessarily lower triangular. One way to determine $\boldsymbol{Q}$ and $\boldsymbol{L}$ is by using an optimisation approach, see Section 4.8.2.

Another implementation alternative is the filter formulation (4.12), where the ILC input signal is computed as

$$u_{k+1}(t) = Q(q)\Big(u_k(t) + L(q)e_k(t)\Big) \tag{9.5}$$

with the error $e_k(t)$ is given by

$$e_k(t) = r(t) - y_k(t) \tag{9.6}$$

and where $Q(q)$ and $L(q)$ may be non-causal filters.

Each of the two alternatives has advantages and disadvantages. The matrix formulation (9.3) is in general more computational demanding, but covers a larger class of algorithms. The filter form (9.5) is on the other hand less general and requires less computations. For both alternatives it is necessary to take care of the boundary effects in the implementation of the ILC algorithm. The main purpose of this chapter is to illustrate that the matrix formulation offers a systematic framework for analysing how this will affect the algorithm properties. Therefore, the implementation of the ILC algorithm in filter form (9.5) is interpreted using the matrix formulation (9.3). This will be discussed in more detail in Sections 9.3 and 9.4.

9.2.3 Stability and convergence properties

Controlling the system (9.2) by the ILC algorithm (9.3) based on the error (9.4) gives the ILC system equation (4.21),

$$\boldsymbol{u}_{k+1} = \boldsymbol{Q}\Big(I - \boldsymbol{L}\boldsymbol{T}_u\Big)\boldsymbol{u}_k + \boldsymbol{Q}\boldsymbol{L}\Big(I - \boldsymbol{T}_r\Big)\boldsymbol{r} \tag{9.7}$$

Theorem 4.5 states that the ILC system (9.7) is stable if and only if

$$\rho\Big(\boldsymbol{Q}(\boldsymbol{I} - \boldsymbol{L}\boldsymbol{T}_{\boldsymbol{u}})\Big) < 1 \tag{9.8}$$

A useful result from Theorem 4.6 concerning the properties of the ILC input signal $\boldsymbol{u}_k$ is that if the system (9.2) is controlled by the ILC algorithm (9.3) and

$$\bar{\sigma}\Big(\boldsymbol{Q}(\boldsymbol{I} - \boldsymbol{L}\boldsymbol{T}_{\boldsymbol{u}})\Big) \leq \lambda < 1 \tag{9.9}$$

that is, the largest singular value is less than one, then the ILC system is stable and

$$\|\boldsymbol{u}_\infty - \boldsymbol{u}_k\|_2 \leq \lambda^k \|\boldsymbol{u}_\infty - \boldsymbol{u}_0\|_2 \tag{9.10}$$

with the asymptotic control signal $\boldsymbol{u}_\infty$ defined as

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

Satisfying the condition (9.9) results in monotone convergence of $\boldsymbol{u}_k$ to the limit value $\boldsymbol{u}_\infty$. Using the asymptotic control signal $\boldsymbol{u}_\infty$ from (9.2.3) it is then straightforward to derive the asymptotic error as

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

Now study the corresponding ILC system in filter form, and assume that the length of the finite time interval goes to infinity. From (4.33) it is stated that if

$$\sup_{\omega\in[-\pi,\pi]} |Q(e^{i\omega})\Big(1 - L(e^{i\omega})T_u(e^{i\omega})\Big)| < 1 \tag{9.11}$$

then the ILC system is stable with monotone convergence of the ILC input signal. This is an approximation of the ILC system operating in finite time, as will be discussed in more detail in Section 9.5.2.

The relation between the time-domain criterion (9.9) and the frequency-domain criterion (9.11), given by Theorem 4.4, is as follows. Study a causal system given by the relation $F(q) = Q(q)\Big(1 - L(q)T_u(q)\Big)$. Now, if

$$\sup_{\omega\in[-\pi,\pi]} |F(e^{i\omega})| < 1 \tag{9.12}$$

then

$$\bar{\sigma}(\boldsymbol{F}_N) < 1 \tag{9.13}$$

where the lower-triangular matrix $\boldsymbol{F}_N$ consists of the N first pulse-response coefficients of $F(q)$.

9.3 Motivating example

In order to illustrate the alternatives for handling the boundary effects of the filtering operations, now assume that the ILC algorithm is given by (9.5),

$$u_{k+1}(t) = Q(q)\Big(u_k(t) + L(q)e_k(t)\Big) \tag{9.14}$$

where

$$L(q) = \gamma q^\delta \tag{9.15}$$

with the integer $\delta > 0$ and the scalar $\gamma > 0$. This means that the error signal $e_k(t)$ is shifted δ steps and scaled by a factor γ when filtered using the filter $L(q)$. Since $e_k(t)$ only is defined on a finite time interval $t \in [0, N-1]$, an assumption

has to be made concerning the values of $e_k(t)$ outside this time interval. As an illustration, two alternatives will be studied. The first alternative is to let

$$e_k(t) = 0, \quad t > N - 1 \tag{9.16}$$

where it is implicitly assumed that the error has reached zero at the end of the given time interval. This implies that filtering $e_k(t)$ using the filter $L(q) = \gamma q^{\delta}$ with the condition (9.16) corresponds to multiplying the error signal vector

$$\boldsymbol{e}_k = \begin{pmatrix} e_k(0) & \dots & e_k(N-1) \end{pmatrix}^T$$

by the matrix

$$\boldsymbol{L} = \begin{pmatrix} 0 & \dots & 0 & \gamma & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & \gamma & \dots & 0 \\ \vdots & & & \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & 0 & 0 & \dots & \gamma \\ 0 & \dots & 0 & 0 & 0 & \dots & 0 \\ \vdots & & & \vdots & & & \vdots \\ 0 & \dots & 0 & 0 & 0 & \dots & 0 \end{pmatrix} \tag{9.17}$$

A second alternative to handle the boundary effect is to put

$$e_k(t) = e_k(N-1), \quad t > N - 1 \tag{9.18}$$

This alternative can be motivated by situations where the error has not reached zero by the end of the movement. It can also be noted that the alternative (9.18) is used in for instance Moore [1998a], Longman [2000] and Wang and Zhang [2009]. This alternative corresponds to the matrix $\boldsymbol{L}$ given by

$$\boldsymbol{L} = \begin{pmatrix} 0 & \dots & 0 & \gamma & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & \gamma & \dots & 0 \\ \vdots & & & \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & 0 & 0 & \dots & \gamma \\ 0 & \dots & 0 & 0 & 0 & \dots & \gamma \\ \vdots & & & \vdots & & & \vdots \\ 0 & \dots & 0 & 0 & 0 & \dots & \gamma \end{pmatrix} \tag{9.19}$$

Since the two alternatives (9.16) and (9.18) for handling the boundary effects correspond to different matrices $\boldsymbol{L}$, it is clear that this will influence the criteria (9.8) and (9.9), respectively. This is studied in a numerical illustration in Section 9.5.

9.4 Handling of boundary effects

In this section a systematic approach is presented for analysis of how different ways to handle the boundary effects of the filtering operations will influence the properties of the ILC algorithm. The key idea is to extend the signal to be filtered and include assumptions concerning the properties of the signal outside the

given time interval $t \in [0, N-1]$. The emphasis will be on handling the boundary effects when filtering using the filter $Q(q)$ or the matrix $\boldsymbol{Q}$, depending on the chosen implementation of the ILC algorithm. A similar procedure can however also be used to represent the filtering using the filter $L(q)$ or the matrix $\boldsymbol{L}$.

As mentioned in Section 9.2, the matrix form (9.3) of the ILC algorithm is more general and better suited for analysis, but the filter formulation (9.5) is less computational demanding. For analysis of the effects of the chosen implementation, it is therefore of interest to be able to interpret the filtering using the filter $Q(q)$ in the matrix formulation. An important special case is when implementing non-causal filters in order to obtain zero-phase shift. A standard way to carry out such filtering is to use a conventional causal filter, for example of Butterworth type, and carry out forward-backward filtering. Consider a causal filter $\bar{Q}(q)$ given by

$$\bar{Q}(q) = \sum_{n=0}^{\infty} g_{\bar{Q}}(n) q^{-n}$$

and the corresponding $N \times N$ Toeplitz matrix

$$\bar{\boldsymbol{Q}} = \begin{pmatrix} g_{\bar{Q}}(0) & 0 & \dots & 0 \\ g_{\bar{Q}}(1) & g_{\bar{Q}}(0) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_{\bar{Q}}(N-1) & g_{\bar{Q}}(N-2) & \dots & g_{\bar{Q}}(0) \end{pmatrix} \tag{9.20}$$

Now consider a signal $x(t)$, $t \in [0, N-1]$ with the elements stacked in a vector

$$\boldsymbol{x} = \begin{pmatrix} x(0) & \dots & x(N-1) \end{pmatrix}^T \tag{9.21}$$

Filtering the signal $x(t)$ through the filter $\bar{Q}(q)$ can then be interpreted as the following matrix-vector multiplication

$$\boldsymbol{x}_f = \bar{\boldsymbol{Q}} \boldsymbol{x}$$

Zero-phase filtering is then obtained by reversing the order of the data points in the vector $\boldsymbol{x}_f$, filtering the data once more through the causal filter $\bar{Q}(q)$, and finally reversing the order of the data points again. Using the matrix representation this corresponds to the operation

$$\boldsymbol{x}_{ff} = \boldsymbol{Q} \boldsymbol{x} \tag{9.22}$$

with the matrix $\boldsymbol{Q}$ given by

$$\boldsymbol{Q} = \bar{\boldsymbol{Q}}^T \bar{\boldsymbol{Q}} \tag{9.23}$$

which can be used in the ILC update equation (9.3).

Now consider again the vector $\boldsymbol{x}$ defined by (9.21). Introduce a corresponding extended vector $\boldsymbol{x}_e$, where the original vector $\boldsymbol{x}$ is extended by m samples at the

beginning and at the end of the time interval, that is,

$$\boldsymbol{x}_e = \begin{pmatrix} x(-m) & \dots & x(0) & \dots & x(N-1) & \dots & x(N-1+m) \end{pmatrix}^T \tag{9.24}$$

where m is a design variable. The extension of the signal vector can be generated by the multiplication

$$\boldsymbol{x}_e = \boldsymbol{Q}_e \boldsymbol{x} \tag{9.25}$$

where the $N \times (N + 2m)$ matrix $\boldsymbol{Q}_e$ can be used to impose assumptions about the properties of the signal vector $\boldsymbol{x}$ outside the given time interval. One alternative is to assume that the signal has the same value outside the time interval as at the end points, similar to the assumption in (9.18). This corresponds to the matrix

$$\boldsymbol{Q}_e = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ \vdots & & & & & & \vdots \\ 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 & 0 \\ \vdots & & & \ddots & & & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 1 \\ \vdots & & & & & & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 & 1 \end{pmatrix} \tag{9.26}$$

Another alternative is to do a linear extrapolation of the signal outside the given time interval, and for example let

$$x(-l) = x(0) + \Big(x(0) - x(l)\Big) = 2x(0) - x(l) \tag{9.27}$$

at the beginning of the signal and

$$\begin{aligned} x(N-1+l) &= x(N-1) + \Big(x(N-1) - x(N-1-l)\Big) \\ &= 2x(N-1) - x(N-1-l) \end{aligned} \tag{9.28}$$

at the end of the signal. This corresponds to the matrix

$$\boldsymbol{Q}_e = \begin{pmatrix} \vdots & & & \iddots & & & \vdots \\ 2 & 0 & -1 & \dots & 0 & 0 & 0 \\ 2 & -1 & 0 & \dots & 0 & 0 & 0 \\ 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 & 0 \\ \vdots & & & \ddots & & & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 1 \\ 0 & 0 & 0 & \dots & 0 & -1 & 2 \\ 0 & 0 & 0 & \dots & -1 & 0 & 2 \\ \vdots & & & \iddots & & & \vdots \end{pmatrix} \tag{9.29}$$

It can be noted that this method is used in the `filtfilt` command [MATLAB Signal Processing Toolbox, 2010], where the signal is extended by m samples at the beginning and at the end of the given time interval. The extension length is given as three times the filter order, as in

$$m = 3\Big(\max\{\text{length}(a), \text{length}(b)\} - 1\Big) \tag{9.30}$$

for a filter with denominator coefficients a and numerator coefficients b. This way of reflecting the signal at the end points is also discussed in Elci et al. [2002].

In the next step the extended vector $\boldsymbol{x}_e$, given by (9.25), is filtered via the matrix-vector multiplication

$$\boldsymbol{y}_e = \tilde{\boldsymbol{Q}}\boldsymbol{x}_e$$

where $\tilde{\boldsymbol{Q}}$ is the $(N + 2m) \times (N + 2m)$ matrix generated from the pulse-response coefficients $\{g_{\bar{Q}}(0), \ldots, g_{\bar{Q}}(N - 1 + 2m)\}$ of the corresponding filter $\bar{Q}(q)$, similarly to how $\boldsymbol{Q}$ is generated in (9.23). In the last step the signal $\boldsymbol{y}_e$ is truncated by removing the first and last m samples of the filtered vector. This is carried out via the relation

$$\boldsymbol{y} = \boldsymbol{Q}_t\boldsymbol{y}_e$$

where

$$\boldsymbol{Q}_t = \begin{pmatrix} 0_{N\times m} & I_{N\times N} & 0_{N\times m} \end{pmatrix}$$

The entire filtering can hence be summarised as

$$\boldsymbol{y} = \boldsymbol{Q}_t\tilde{\boldsymbol{Q}}\boldsymbol{Q}_e\boldsymbol{x} = \boldsymbol{Q}\boldsymbol{x} \tag{9.31}$$

with the matrix $\boldsymbol{Q}$, which can be used in the ILC update equation (9.3). A similar procedure can be used to represent the filtering using the filter $L(q)$ in (9.5), but for simplicity this study will be limited to the cases represented by (9.16) and (9.18).

9.5 Numerical illustration

The purpose is now to provide a qualitative illustration that the implementation of the ILC algorithm can play an important role for the algorithm behaviour. The case where an ILC algorithm is applied to a two-mass model is used as an illustrative example of the implementation aspects.

9.5.1 System description

The two-mass system used for the illustration consists of two masses connected by a spring-damper pair[3]. The input to the system is the torque $\tau(t)$ applied to the first mass. In the simulation study the continuous-time two-mass system is discretised with the MATLAB command `c2d` using the zero-order hold method with a sampling interval $T_s = 0.01$ s. The motor angular position $q_m(t)$ of the first

[3] See Section 2.3.3 for the details, and an illustration of the model in Figure 2.4.

mass is the output of the discrete-time system, denoted $G(q)$ in Figure 9.1, and it is controlled by using a discrete-time PD-controller $F(q)$ including a low-pass filter. The controller is obtained by manual tuning, and is given by

$$F(q) = K_1 + \frac{K_2 q - K_3}{q - K_4}$$

The model parameter values in Table 9.1 and the controller parameter values in Table 9.2 will be used in the simulations. The structure of the control system is depicted in Figure 9.1, where it is seen that the ILC input signal $u_k(t)$ is added to the reference signal $r_m(t)$ of the existing control system. In the simulation example it means that, referring to the relation (9.1),

$$T_r(q) = T_u(q) = \frac{F(q)G(q)}{1 + F(q)G(q)} \tag{9.32}$$

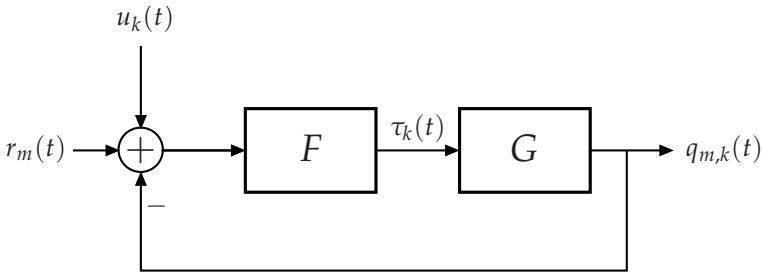


Figure 9.1: *Simulation setup with the two-mass system represented by $G(q)$ in discrete time. The variable $\tau_k(t)$ denotes the motor torque from the feedback controller $F(q)$ at iteration k. The ILC input signal $u_k(t)$ is added to the reference signal $r_m(t)$ of the feedback control system. The system output in the simulation study is the motor angular position $q_{m,k}(t)$ at iteration k.*

The numerical illustration will be based on the ILC algorithm (9.14) with the filter $L(q)$ given as in (9.15), that is,

$$u_{k+1}(t) = Q(q)\Big(u_k(t) + \gamma q^{\delta} e_k(t)\Big) \tag{9.33}$$

with the motor angular position error at iteration k given by

$$e_k(t) = r_m(t) - q_{m,k}(t) \tag{9.34}$$

The filter $Q(q)$ is chosen as a zero-phase low-pass filter. The zero-phase filtering is carried out via forward-backward filtering using a second-order low-pass Butterworth filter with cutoff frequency $f_c = 10\,\text{Hz}$. The ILC design parameter values are given in Table 9.3, together with the length m of the signal extension seen in (9.24). The choice $m = 6$ corresponds to the signal extension given by (9.30) when using the MATLAB function `filtfilt` and having a second-order filter.

Table 9.1: *Model parameter values.*

$\eta = 0.2$	$M_m = 0.0021$	$M_a = 0.0991$	$k = 8$
$d = 0.0924$	$f_m = 0.0713$	$k_\tau = 0.122$	

Table 9.2: *Controller parameter values.*

$K_1 = 5$	$K_2 = 2$	$K_3 = 2$	$K_4 = 0.905$

Table 9.3: *ILC design parameter values.*

$\gamma = 0.9$	$\delta = 10$	$f_c = 10$	$m = 6$

9.5.2 Analysis

First, stability of the ILC algorithm (9.33) applied to the discrete-time description of the two-mass system in previous section with parameter values given in Tables 9.1 to 9.3 is analysed in the frequency domain. The convergence condition (9.11) results in

$$\sup_{\omega\in[-\pi,\pi]} |Q(e^{i\omega})\Big(1 - L(e^{i\omega})T_u(e^{i\omega})\Big)| = 0.904 \tag{9.35}$$

from which it can be concluded that the ILC system is stable, and results in monotone convergence of the ILC input signal to the limit $u_\infty(t)$.

However, the relation (9.35) cannot reflect any implementation issues regarding filtering over finite-time intervals of the signals through the (possibly non-causal) ILC filters involved. The frequency-domain analysis is an approximation, and only reflects the properties under the assumption of infinite time horizon. Therefore, the criterion (9.35) cannot take the boundary effects into account, which is an important part of the ILC system performance since the time horizon is finite in all practical applications of ILC.

The time-domain analysis of how the boundary effects will affect the properties of the ILC algorithm will be carried out using the matrix representation of the discrete-time system and ILC algorithm. Therefore, the matrix $\boldsymbol{T}_u$ is generated from the pulse-response coefficients of the transfer operator $T_u(q)$ in (9.32) of the two-mass system described in Section 9.5.1. In a similar way, the ILC algorithm (9.33) is converted to matrix form (9.3) represented by the matrices $\boldsymbol{Q}$ and $\boldsymbol{L}$, respectively.

The cases to be studied in the numerical example are the following. When filtering the error signal $e_k(t)$ using the filter $L(q)$, the error signal is assumed to be:

A: zero outside the time interval, that is, $\boldsymbol{L}$ is interpreted according to (9.17).

B: extended with the last value, that is, $\boldsymbol{L}$ is interpreted according to (9.19).

When filtering through $Q(q)$, the signal is assumed to be:

I: not extended.

II: extended using the first and last value, that is, $\boldsymbol{Q}_e$ is interpreted according to (9.26).

III: extended according to extrapolation, that is, $\boldsymbol{Q}_e$ is interpreted according to (9.29).

First the time-domain stability criterion (9.8) is investigated. Inserting the matrices $\boldsymbol{T}_u$, $\boldsymbol{Q}$, and $\boldsymbol{L}$ into (9.8) gives the results shown in Table 9.4 for the combinations of the cases A-B and I-III. $N = 400$ number of samples is used in the calculations, motivated by simulation of the system during 4 s with a sampling interval of $T_s = 0.01$ s in the forthcoming section. The values of the stability criterion (9.8) presented in Table 9.4 show that the implementation plays a crucial role for the properties of the ILC algorithm. Alternatives A-II, A-III and B-III give a divergent algorithm with $\rho\Big(\boldsymbol{Q}(\boldsymbol{I} - \boldsymbol{L}\boldsymbol{T}_u)\Big) > 1$, while the other alternatives imply convergence. For alternative II it can also be seen that it plays an important role how the filtering via $L(q)$ is handled.

Since the implementation aspects now are involved in the matrix formulation of the individual ILC filters, it could give a possibly different result compared to the resulting frequency-domain convergence criterion (9.35). Table 9.5 shows the singular value condition (9.9) for the different implementation alternatives. The pattern is similar to the one seen in Table 9.4, with monotone convergence according to (9.10) for case I. Results for case I also show that it can be expected that the choice of $\boldsymbol{L}$ will have influence on the convergence rate of the ILC algorithm. This will be further investigated in simulations.

Table 9.4: *Stability criterion* (9.8) *for the extension alternatives A-B for filter* $L(q)$ *and I-III for filter* $Q(q)$*. Alternatives A-II, A-III and B-III give a divergent algorithm, while the other alternatives imply convergence.*

	I	*II*	*III*
A	0.895	1.007	1.101
B	0.891	0.931	1.080

Table 9.5: *Singular value condition* (9.9) *for the extension alternatives A-B for filter* $L(q)$ *and I-III for filter* $Q(q)$*. Alternatives A-I and B-I gives monotone convergence.*

	I	*II*	*III*
A	0.944	1.048	1.371
B	0.904	1.018	1.397

9.5.3 Simulation results

The properties of the ILC algorithm (9.33) applied the discrete-time two-mass system are evaluated using simulations in MATLAB for the different implementation alternatives A-B and I-III. The motor angular position reference $r_m(t)$ for the two-mass system is shown in Figure 9.2 with a duration of 4 s and sampling interval $T_s = 0.01$ s. For the simulations, the ILC algorithm (9.33) is converted to the matrix form (9.3) according to the descriptions in Sections 9.3 and 9.4. The system performance is evaluated using the 2-norm of the vector of the error (9.34),

$$\boldsymbol{e}_k = \begin{pmatrix} e_k(0) & \dots & e_k((N-1)T_s) \end{pmatrix}^T \tag{9.36}$$

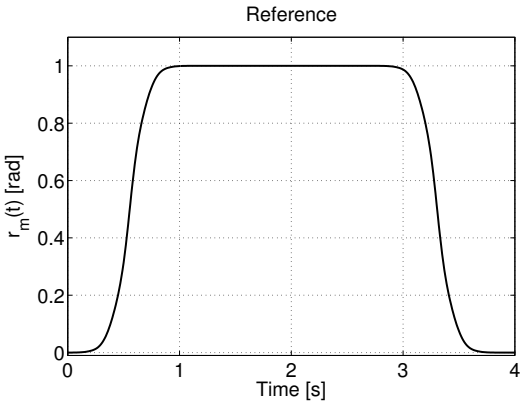


Figure 9.2: *Motor angular position reference $r_m(t)$ used in the simulations.*

Case I: Figure 9.3 shows the 2-norm of the error (9.36) as a function of iteration number for the cases A-I and B-I. The curves can be compared to the first column of Tables 9.4 and 9.5, respectively. The values for the spectral radius are about the same for the two cases. The largest singular value is smaller for case B-I, which results in a better guaranteed convergence rate than for case A-I, see (9.9) to (9.10). A better convergence rate for case B-I compared to case A-I can also be noticed in Figure 9.3. Furthermore, it is seen that the magnitude of the final error is affected by how the boundary conditions are handled.

Case II: Figure 9.4 shows the corresponding curves for cases A-II and B-II, respectively. In case B-II the largest eigenvalue is slightly larger than for cases A-I and B-I, while the eigenvalue for case A-II is just outside the stability boundary. This property with $\rho\big(\boldsymbol{Q}(I - \boldsymbol{L}\boldsymbol{T}_u)\big) > 1$ is seen in the simulation results, where the norm of the error for case A-II decreases during the first 50 iterations and then starts to increase.

Case III: Figure 9.5 illustrates the behaviour for cases A-III and B-III, respectively. Here the eigenvalues shown in the third column of Table 9.4 indicate that the ILC system is unstable for both cases, and this is confirmed by the simulation results in Figure 9.5.

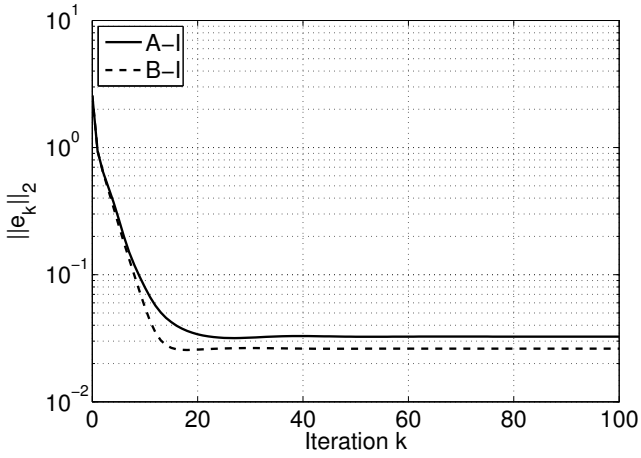


Figure 9.3: *2-norm of the error as function of iteration index for the cases A-I and B-I.*

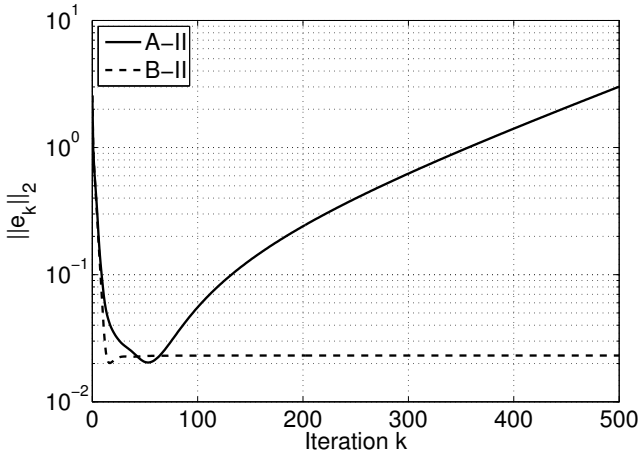


Figure 9.4: *2-norm of the error as function of iteration index for the cases A-II and B-II.*

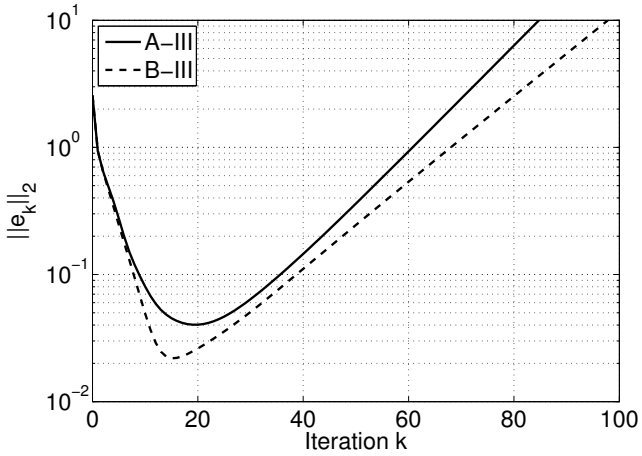


Figure 9.5: *2-norm of the error as function of iteration index for the cases A-III and B-III.*

The function `filtfilt` handles boundary conditions similar to the implementation alternative III above, see MATLAB Signal Processing Toolbox [2010]. The function also involves a way of handling initial conditions that depends on the signal itself, which implies that the behaviour is not predictable without knowledge of the filtered signal. In order to illustrate some properties, the `filtfilt` alternative is also investigated. Figure 9.6 shows the 2-norm of the error (9.36) when the filter form (9.33) of the ILC algorithm is used together with the `filtfilt` command, combined with the two alternative ways A and B of handling the filtering with $L(q)$. Also here it is beneficial to use alternative B. Running the algorithm for more iterations gives that the error signal settles at a constant value after approximately 1000 iterations for alternative B.

9.6 Conclusions

Some implementation aspects related to the non-causal filtering operations of the ILC algorithms have been discussed. Different ways to treat the boundary effects have been analysed both theoretically, by interpreting the filtering in the matrix description, and by using simulations of a two-mass system controlled by an ILC algorithm. The results indicate that the implementation method and how to handle boundary effects can play an important role for the behaviour of the ILC algorithm in terms of stability, convergence and final error level. Moreover, the conventional frequency-domain convergence criterion only reflects the properties of the system controlled by ILC under the assumption of infinite time horizon. It is illustrated that the frequency-domain criterion cannot generally be used for analysis of how the boundary effects influence the properties of the ILC algorithm.

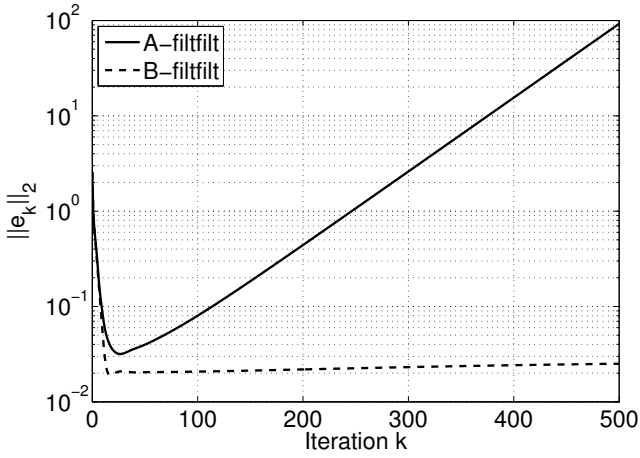


Figure 9.6: *2-norm of the error as function of iteration index when the filter form* (9.33) *of the ILC algorithm is used for the cases A and B together with the* `filtfilt` *command.*

10

Concluding remarks

THIS CHAPTER PROVIDES a summary of the results presented in Part II of the thesis. Possible directions for future work are also discussed.

10.1 Conclusions

In Chapter 5 the case is studied when an ILC algorithm is applied to a system containing mechanical flexibilities. Of particular interest is the situation where the controlled variable is not the measured variable. Experiments are performed on both a serial and a parallel robot, using the measured motor angular positions (measured variables) directly in the ILC algorithm. The main result from the experiments is that although the motor performance is improved, it does not necessarily imply an improved tool position. The underlying problem is analysed by a simulation study of a flexible two-mass model. The study illustrates the difficulties in improving performance of the controlled variable by ILC algorithms using only the measured variable directly. The results serve as a motivation to include additional sensors and use estimates of the controlled variable in the ILC algorithm.

A framework for analysis of estimation-based ILC is presented in Chapter 6. The system controlled using ILC is a natural extension of the system description in Norrlöf and Gunnarsson [2002a], where now also the dynamic relationship between the measured and controlled variable is taken into account. The focus in the analysis is the performance of the controlled variable, and a general expression for the asymptotic error of the controlled variable is given under the assumption that the ILC input converges to a bounded signal. The asymptotic error depends both on the ILC operators, the controller of the system, the esti-

mation algorithm and the model errors. The dependence on the model errors is discussed for three cases of possible estimates of the controlled variable, and is exemplified by an ILC algorithm applied to a flexible two-mass model. It can be seen that an ILC algorithm using an estimate of the controlled variable can improve the system performance. The resulting performance relies on the quality of the estimate, as expected.

The idea of estimation-based ILC is evaluated in Chapter 7 for a nonlinear two-link robot model having flexible joints and with an additional accelerometer on the robot tool. The ILC algorithm uses estimates of the robot tool position derived from an extended Kalman filter (EKF). The work serves as a case study for estimation-based ILC when combining EKF and ILC. Compared to an ILC algorithm using only the error of the measured variable directly, it is possible to improve the performance of the controlled variable (here robot tool position) when an estimate is used in the ILC algorithm.

Estimation-based ILC is experimentally evaluated on the Gantry-Tau parallel robot in Chapter 8, where an additional accelerometer is mounted on the end-effector plate. Estimates of the robot tool position are derived from complementary filtering and Kalman filtering, respectively. First, the ILC algorithm uses only the measured motor angular positions directly, the original measurements available in the robot system. Next, tool-position estimates are used in the ILC algorithm, with the algorithm tuned such that learning of frequency components of the error up to and above the dominating resonance frequencies of the system is possible. From the experiments it can be concluded that the tool performance can be improved by using estimation-based ILC.

In Chapter 9 time-domain analysis is performed of how different ways to treat the boundary effects in the filtering operations affect the convergence properties of the ILC algorithm. A numerical example is studied, where it is illustrated that the frequency-domain criterion cannot generally be used for analysis of how the boundary effects will influence the resulting performance — depending on the handling of the boundary effects, the frequency-domain convergence criterion can be valid or not valid. Simulation results for cases when handling the boundary effects in different ways indicate that the implementation method can have large influence on the system performance.

10.2 Future work

In the presented framework for analysis of estimation-based ILC, the individual contributions — choice of controller for the system, ILC filters and estimation filters — to the resulting system performance can be seen. This gives possibilities to analyse how different aspects of control, estimation and ILC affects the whole system. From a user perspective, the asymptotic error of the controlled variable is to be minimised under the constraint that the largest singular value of the ILC system is smaller than one. A future research problem is to formally formulate the optimisation problem and to find effective solution methods.

In Chapter 8 an ILC algorithm is applied to the Gantry-Tau parallel robot, using an estimate of the robot tool position. The choice of estimation algorithm, as well as the quality of the estimate, rely on how accurately the robot model can describe the essential dynamics of the robot system. During the experiments the need for a more comprehensive and nonlinear model has been evident, including for example flexible dynamics, friction and backlash. This requires comprehensive work regarding the choice of model structure, the design of identification experiments and suitable identification methods.

The quality of the robot tool-position estimate is fundamental for a successful application of estimation-based ILC to the system. One possibility is to derive a nonlinear observer, utilising the particular structure of the nonlinear dynamic model. Worth discussing is estimation by using measurements from different types of sensors, for example accelerometers and vision systems, and also utilise other measurements available in the robot system, for example the motor torque values. One question to be answered is where to place the sensors and how many sensors that are needed in order to achieve a sufficiently good estimate to be used in the ILC algorithm.

A natural continuation of the work presented in Chapter 9 is to experimentally investigate the effects of different choices of how the boundary effects of the filtering operations are handled. The effects could possibly be decreased by using a lead-in/lead-out part of the trajectory with time-domain weighting of the ILC input signal.

There is a constant need for development of control methods to improve the performance of the robot tool. The trend towards more sensors in the robot system also makes it possible to form estimates of the relevant signals by using information from the additional sensors. Hence, commercialisation of estimation-based ILC is of interest to many robot manufacturers.

Bibliography

ABB Robotics, 2007. Open public archive of robot images. URL: `http://www.abb.se`, accessed August, 2007.

Houssem Abdellatif and Bodo Heimann. Advanced model-based control of a 6-DOF hexapod robot: A case study. *IEEE/ASME Transactions on Mechatronics*, 15(2):269–279, 2010.

Houssem Abdellatif, Matthias Feldt, and Bodo Heimann. Application study on iterative learning control of high speed motions for parallel robotic manipulator. In *Proceedings of IEEE International Conference on Control Applications*, pages 2528–2533, Munich, Germany, October 2006.

Hyo-Sung Ahn, YangQuan Chen, and Kevin L. Moore. Iterative learning control: Brief survey and categorization. *IEEE Transactions on Systems, Man, and Cybernetics — Part C: Applications and Reviews*, 37(6):1099–1121, November 2007.

Tarek Al-Towaim, Andrew D. Barton, Paul L. Lewin, Eric Rogers, and David H. Owens. Iterative learning control — 2D control systems from theory to application. *International Journal of Control*, 77(9):877–893, 2004.

James S. Albus. Outline for a theory of intelligence. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):473–509, 1991.

Lawrence J. Alder and Stephen M. Rock. Experiments in control of a flexible-link robotic manipulator with unknown payload dynamics: An adaptive approach. *The International Journal of Robotics Research*, 13(6):481–495, 1994.

Lawrence J. Alder and Stephen M. Rock. Frequency-weighted state estimation with application to estimation in the presence of sensor bias. *IEEE Transactions on Control Systems Technology*, 4(4):427–436, 1996.

Notker Amann, David H. Owens, and Eric Rogers. Iterative learning control using optimal feedback and feedforward actions. *International Journal of Control*, 65(2):277–293, 1996a.

Notker Amann, David H. Owens, and Eric Rogers. Iterative learning control for discrete-time systems with exponential rate of convergence. *IEE Proceedings, Part D, Control Theory and Applications*, 143(2):217–224, 1996b.

Brian D. O. Anderson and John B. Moore. *Optimal Filtering*. Prentice-Hall, Inc, Englewood Cliffs, NJ, USA, 1979.

Suguru Arimoto. Mathematical theory of learning with applications to robot control. In *Proceedings of 4th Yale Workshop on Applications of Adaptive Systems*, New Haven, CN, USA, May 1985.

Suguru Arimoto. Learning control theory for robotic motion. *International Journal of Adaptive Control and Signal Processing*, 4(6):543–564, 1990.

Suguru Arimoto. A brief history of iterative learning control. In Zeungnam Bien and Jian-Xin Xu, editors, *Iterative Learning Control: Analysis, Design, Integration and Applications*, pages 3–7. Kluwer Academic Publishers, Boston, MA, USA, 1998.

Suguru Arimoto, Sadao Kawamura, and Fumio Miyazaki. Iterative learning control for robot systems. In *Proceedings of Annual Conference of the IEEE Industrial Electronics Socitey, IECON*, Tokyo, Japan, October 1984a.

Suguru Arimoto, Sadao Kawamura, and Fumio Miyazaki. Bettering operation of robots by learning. *Journal of Robotic Systems*, 1(2):123–140, 1984b.

Brian Armstrong-Hélouvry. *Control of Machines with Friction*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

Isaac Asimov. Runaround. *Astounding Science-Fiction*, pages 94–103, March 1942.

Isaac Asimov. *Robots and Empire*. Grafton Books, London, England, 1985.

Kira Barton, Jeroen van de Wijdeven, Andrew Alleyne, Okko Bosgra, and Maarten Steinbuch. Norm optimal cross-coupled iterative learning control. In *Proceedings of IEEE Conference on Decision and Control*, Cancun, Mexico, December 2008.

Kira L. Barton and Andrew G. Alleyne. A cross-coupled iterative learning control design for precision motion control. *IEEE Transactions on Control Systems Technology*, 16(6):1218–1231, 2008.

Zeungnam Bien and Kyung M. Huh. Higher-order iterative learning control algorithm. *IEE Proceedings, Part D, Control Theory and Applications*, 136(3): 105–112, May 1989.

Zeungnam Bien and Jian-Xin Xu, editors. *Iterative Learning Control: Analysis, Design, Integration and Applications*. Kluwer Academic Publishers, Boston, MA, USA, 1998.

Mattias Björkman, Torgny Brogårdh, Sven Hanssen, Sven-Erik Lindström, Stig Moberg, and Mikael Norrlöf. A new concept for motion control of industrial

robots. In *Proceedings of IFAC World Congress*, pages 15714–15715, Seoul, Korea, July 2008.

Anders Blomdell, Gunnar Bolmsjö, Torgny Brogårdh, Per Cederberg, Mats Isaksson, Rolf Johansson, Mathias Haage, Klas Nilsson, Magnus Olsson, Tomas Olsson, Anders Robertsson, and Jianjun Wang. Extending an industrial robot controller: Implementation and applications of a fast open sensor interface. *IEEE Robotics and Automation Magazine*, 12(3):85–94, September 2005.

Gunnar S. Bolmsjö. *Industriell robotteknik*. Studentlitteratur, Lund, 2nd edition, 1992. In Swedish.

Paola Bondi, Giuseppe Casalino, and Lucia Gambardella. On the iterative learning control theory for robotic manipulators. *IEEE Journal of Robotics and Automation*, 4(1):14–22, February 1988.

Douglas A. Bristow, Marina Tharayil, and Andrew G. Alleyne. A survey of iterative learning control. *IEEE Control Systems Magazine*, 26(3):96–114, June 2006.

Torgny Brogårdh. PKM research — important issues, as seen from a product development perspective at ABB Robotics. In *Proceedings of Workshop on Fundamental issues and future research directions to parallel mechanisms and manipulators*, Quebec City, Canada, October 2002.

Torgny Brogårdh. Present and future robot control development — An industrial perspective. *Annual Reviews in Control*, 31(1):69–79, 2007.

Torgny Brogårdh. Robot control overview: An industrial perspective. *Modeling, Identification and Control*, 30(3):167–180, 2009.

Etienne Burdet, Laurent Rey, and Alain Codourey. A trivial and efficient learning method for motion and force control. *Engineering Applications of Artificial Intelligence*, 14(4):487–496, 2001.

Giuseppe Casalino and Giorgio Bartolini. A learning procedure for the control of movements of robotic manipulators. In *Proceedings of IASTED Symposium on Robotics and Automation*, pages 108–111, San Francisco, CA, USA, May 1984.

Marzia Cescon, Isolde Dressler, Rolf Johansson, and Anders Robertsson. Subspace-based identification of compliance dynamics of parallel kinematic manipulator. In *Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 1028–1033, Singapore, Singapore, July 2009.

Hong-Jen Chen and Richard W. Longman. The importance of smooth updates in producing good error levels in repetitive control. In *Proceedings of IEEE Conference on Decision and Control*, pages 258–263, Phoenix, AZ, USA, December 1999.

YangQuan Chen and Changyun Wen. *Iterative Learning Control: Convergence,*

Robustness and Applications, volume 248 of *Lecture Notes in Control and Information Sciences*. Springer Verlag, London, England, 1999.

YangQuan Chen, Zhiming Gong, and Changyun Wen. Analysis of a high-order iterative learning control algorithm for uncertain nonlinear systems. *Automatica*, 34(3):345–353, March 1998.

Yangquan Chen, Leeling Tan, Kiankeong Ooi, Qiang Bi, and Kokhiang Cheong. Repeatable runout disturbance compensation with a new data collection method for hard disk drive. United States Patent 6437936, 2002.

YuangQuan Chen and Kevin L. Moore. Comments on United States Patent 3555252 — learning control of actuators in control systems. In *Proceedings of The Sixth International Conference on Control, Automation, Robotics and Vision*, Singapore, Singapore, December 2000.

Jacob W. F. Cheung and Yeung Sam Hung. Robust learning control of a high precision planar parallel manipulator. *Mechatronics*, 19(1):42–55, 2009.

Hua-Yi Chuang and Yung-Chih Chang. Dynamics analysis and learning control for 3-PRPS platform. *International Journal of Computer Applications in Technology*, 14(4–6):204–214, 2001.

Peter Corke. An inertial and visual sensing system for s small autonomous helicopter. *Journal of Robotic Systems*, 21(2):43–51, 2004.

John J. Craig. Adaptive control of manipulators through repeated trials. In *Proceedings of American Control Conference*, pages 1566–1572, San Diego, CA, USA, June 1984.

John J. Craig. *Introduction to robotics mechanics and control*. Addison-Wesley Publishing Company, Inc, Reading, MA, USA, 2nd edition, 1989.

Phil Crothers, Philip Freeman, Torgny Brogårdh, Isolde Dressler, Klas Nilsson, Anders Robertsson, Walter Zulauf, Beat Felder, Raimund Loser, and Knut Siercks. Characterisation of the Tau parallel kinematic machine for aerospace application. *SAE International Journal of Aerospace*, 2(1):205–213, March 2010.

Pawel Dabkowski, Krysztof Gałkowski, Biswa Datta, and Eric Rogers. LMI based stability and stabilization of second-order linear repetitive processes. *Asian Journal of Control*, 12(2):136–145, March 2010.

Alessandro De Luca and Wayne Book. Robots with flexible elements. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 287–319. Springer Verlag, Berlin Heidelberg, Germany, 2008.

Alessandro De Luca and Giovanni Ulivi. Iterative learning control of robots with elastic joints. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1920–1926, Nice, France, May 1992.

Alessandro De Luca, Dierk Schröder, and Michael Thümmel. An acceleration-based state observer for robot manipulators with elastic joints. In *Proceedings*

of IEEE International Conference on Robotics and Automation, pages 3817–3823, Roma, Italy, April 2007.

Dick de Roover. Synthesis of a robust iterative learning controller using an H_∞ approach. In *Proceedings of IEEE Conference on Decision and Control*, pages 3044–3049, Kobe, Japan, December 1996.

Jacques Denavit and Richard S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *ASME Journal of Applied Mechanics*, pages 215–221, June 1955.

Branko G. Dijkstra and Okko H. Bosgra. Extrapolation of optimal lifted system ILC solution with application to a waferstage. In *Proceedings of American Control Conference*, pages 2595–2600, Anchorage, AK, USA, May 2002.

DLR, 2010. German Aerospace Center, open public archive of robot images. URL: `http://www.dlr.de/rm/en/`, accessed December, 2010.

Isolde Dressler, Mathias Haage, Klas Nilsson, Rolf Johansson, Anders Robertsson, and Torgny Brogårdh. Configuration support and kinematics for a reconfigurable Gantry-Tau manipulator. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2957–2962, Roma, Italy, April 2007a.

Isolde Dressler, Anders Robertsson, and Rolf Johansson. Accuracy of kinematic and dynamic models of a Gantry-Tau parallel kinematic robot. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 883–888, Roma, Italy, April 2007b.

Isolde Dressler, Torgny Brogårdh, and Anders Robertsson. A kinematic error model for a parallel Gantry-Tau manipulator. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3709–3714, Anchorage, AK, USA, May 2010.

Hugh Durrant-Whyte and Thomas C. Henderson. Multisensor data fusion. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 585–610. Springer Verlag, Berlin Heidelberg, Germany, 2008.

Haluk Elci, Richard W. Longman, Minh Q. Phan, Jer-Nan Juang, and Roberto Ugoletti. Discrete frequency based learning control for precision motion control. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, pages 2767–2773, San Antonio, TX, USA, October 1994.

Haluk Elci, Richard W. Longman, Minh Q. Phan, Jer-Nan Juang, and Roberto Ugoletti. Simple learning control made practical by zero-phase filtering: Applications to robotics. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 49(6):753–767, June 2002.

Christopher T. Freeman, Paul L. Lewin, Eric Rogers, and James D. Ratcliffe. Iterative learning control applied to a gantry robot and conveyor system. *Transactions of the Institute of Measurement and Control*, 32(3):251–264, June 2010.

Freescale. Product documentation for accelerometer MMA7361L, 2010. URL: http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=KIT3376MMA73x1L, accessed September, 2010.

Murray Garden. Learning control of actuators in control systems. United States Patent 03555252, January 1971. Leeds & Northrup Company, Philadelphia, USA.

Peter B. Goldsmith. On the equivalence of causal LTI iterative learning control and feedback control. *Automatica*, 38(4):703–708, 2002.

Herbert Goldstein, Charles Poole, and John Safko. *Classical mechanics*. Addison-Wesley Publishing Company, Inc, San Francisco, CA, USA, 3rd edition, 2002.

Clement Gosselin and Jorge Angeles. Singularity analysis of closed-loop kinematic chains. *IEEE Transactions on Robotics and Automation*, 6(3):281–290, 1990.

Mohinder S. Grewal and Angus P. Andrews. *Kalman filtering: theory and practice using MATLAB*. John Wiley & Sons, Ltd, Hoboken, NJ, USA, 3rd edition, 2008.

Svante Gunnarsson and Mikael Norrlöf. On the design of ILC algorithms using optimization. *Automatica*, 37(12):2011–2016, December 2001.

Svante Gunnarsson and Mikael Norrlöf. On the disturbance properties of high order iterative learning control algorithms. *Automatica*, 42(11):2031–2034, November 2006.

Svante Gunnarsson, Mikael Norrlöf, Geir Hovland, Ulf Carlsson, Torgny Brogårdh, Tommy Svensson, and Stig Moberg. Pathcorrection for an industrial robot. United States Patent 7130718, October 2006.

Svante Gunnarsson, Mikael Norrlöf, Enes Rahic, and Markus Özbek. On the use of accelerometers in iterative learning control of a flexible robot arm. *International Journal of Control*, 80(3):363–373, March 2007.

Fredrik Gustafsson. *Adaptive filtering and change detection*. John Wiley & Sons, Ltd, Chichester, England, 2000.

Shinji Hara, Yutaka Yamamoto, Tohru Omata, and Michio Nakano. Repetitive control system: A new type servo system for periodic exogeneous signals. *IEEE Transactions on Automatic Control*, 33(7):659–668, July 1988.

Jari J. Hätönen, David H. Owens, and Kevin L. Moore. An algebraic approach to iterative learning control. *International Journal of Control*, 77(1):45–54, January 2004.

Heidenhain. Product documentation for length gauge ST 3078, 2010. URL: http://www.heidenhain.com, accessed September, 2010.

Robert Henriksson, Mikael Norrlöf, Stig Moberg, Thomas B. Schön, and Erik Wernholt. Experimental comparison of observers for tool position estimation

of industrial robots. In *Proceedings of IEEE Conference on Decision and Control*, pages 8065–8070, Shanghai, China, December 2009.

Walter T. Higgins, Jr. A comparison of complementary and Kalman filtering. *IEEE Transactions on Aerospace and Electronic Systems*, AES-11(3):321–325, May 1975.

Lukasz Hladowski, Krzysztof Galkowski, Zonglun Cai, Eric Rogers, Chris T. Freeman, and Paul L. Lewin. Experimentally supported 2D systems based iterative learning control law design for error converence and performance. *Control Engineering Practice*, 18(4):339–348, 2010.

Geir Hovland, Matthew Murray, and Torgny Brogårdh. Experimental verification of friction and dynamic models of a parallel kinematic machine. In *Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 82–87, Zürich, Switzerland, September 2007.

International Federation of Robotics. World Robotics 2006, Executive Summary, 2006. URL: `http://www.ifrstat.org/downloads/2007_Executive_Summary.pdf`, accessed December 2010.

International Federation of Robotics. World Robotics 2010, Executive Summary, 2010. URL: `http://www.worldrobotics.org/downloads/2010_Executive_Summary.pdf`, accessed December 2010.

ISO. Svensk standard SS-EN ISO 8373. Manipulating industrial robots — Vocabulary (ISO 8373:1994), 1996.

Mrdjan Jankovic. Observer based control for elastic joint robots. *IEEE Transactions on Robotics and Automation*, 11(4):618–623, 1995.

Lars Johannesson, Viktor Berbyuk, and Torgny Brogårdh. Gantry-Tau — a new three degrees of freedom parallel kinematic robot. In *Proceedings of 4th Chemnitz Parallel Kinematics Seminar*, pages 731–734, Chemnitz, Germany, April 2004.

Thomas Kailath, Ali H. Sayed, and Babak Hassibi. *Linear estimation*. Prentice-Hall, Inc, Upper Saddle River, NJ, USA, 2000.

Rudolph E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82:35–45, 1960.

Rickard Karlsson and Mikael Norrlöf. Position estimation and modeling of a flexible industrial robot. In *Proceedings of IFAC World Congress*, Prague, Czech Republic, July 2005.

Sadao Kawamura, Fumio Miyazki, and Suguru Arimoto. Realization of robot motion based on a learning method. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):126–134, January 1988.

Jerzy E. Kurek and Marek B. Zaremba. Iterative learning control synthesis based

on 2-D system theory. *IEEE Transactions on Automatic Control*, 38(1):121–125, January 1993.

Kwang Soon Lee and Jay H. Lee. Design of quadratic criterion-based iterative learning control. In Zeungnam Bien and Jian-Xin Xu, editors, *Iterative Learning Control: Analysis, Design, Integration and Applications*, pages 165–192. Kluwer Academic Publishers, Boston, MA, USA, 1998a.

Kwang Soon Lee and Jay H. Lee. Model-based predictive control combined with iterative learning for batch or repetitive processes. In Zeungnam Bien and Jian-Xin Xu, editors, *Iterative Learning Control: Analysis, Design, Integration and Applications*, pages 313–334. Kluwer Academic Publishers, Boston, MA, USA, 1998b.

Kyoung H. Lee and Zeungnam Bien. Initial condition problem of learning control. *IEE Proceedings, Part D, Control Theory and Applications*, 138(6):525–528, 1991.

Leica Geosystems. Laser tracker systems, 2010. URL: `http://metrology.leica-geosystems.com/en/Laser-Tracker-Systems_69045.htm`, accessed April, 2010.

Vatchara Lertpiriyasuwat and Martin Berg. Extended Kalman filtering applied to a two-axis robotic arm with flexible links. *The International Journal of Robotics Research*, 19(3):254–270, 2000.

Youfu Li and Daniel X. Chen. End-point sensing and state observation of a flexible-link robot. *IEEE/ASME Transactions on Mechatronics*, 6(3):351–356, 2001. ISSN 1083-4435.

Lennart Ljung. *System Identification — Theory for the User*. Prentice-Hall, Inc, Upper Saddle River, NJ, USA, 2nd edition, 1999.

Lennart Ljung. *System Identification Toolbox 7. User's Guide*. The MathWorks, 2010.

Richard W. Longman. Designing iterative learning and repetitive controllers. In Zeungnam Bien and Jian-Xin Xu, editors, *Iterative Learning Control: Analysis, Design, Integration and Applications*, pages 107–146. Kluwer Academic Publishers, Boston, MA, USA, 1998.

Richard W. Longman. Iterative learning control and repetitive control for engineering practice. *International Journal of Control*, 73(10):930–954, July 2000.

Richard W. Longman and Thuanthong Songchon. Trade-offs in designing learning/repetitive controllers using zero-phase filtering for long term stabilization. *Advances in the Astronautical Sciences*, 102:243–262, 1999.

Richard W. Longman and Sven-Lennart Wirkander. Automated tuning concepts for iterative learning and repetitive control laws. In *Proceedings of IEEE Conference on Decision and Control*, pages 192–198, Tampa, FL, USA, December 1998.

Ola Markusson, Håkan Hjalmarsson, and Mikael Norrlöf. Iterative learning control of nonlinear non-minimum phase systems and its application to system and model inversion. In *Proceedings of IEEE Conference on Decision and Control*, pages 4481–4482, Orlando, FL, USA, December 2001.

Jean-Pierre Merlet. *Parallel Robots*. Springer, Dordrecht, The Netherlands, 2nd edition, 2006.

Jean-Pierre Merlet and Clément Gosselin. Parallel mechanisms and robots. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 269–285. Springer Verlag, Berlin Heidelberg, Germany, 2008.

Stig Moberg. *Modeling and Control of Flexible Manipulators*. Dissertation No. 1349, Department of Electrical Engineering, Linköping University, Linköping, Sweden, December 2010.

Stig Moberg and Sven Hanssen. A DAE approach to feedforward control of flexible manipulators. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3439–3444, Roma, Italy, April 2007.

Stig Moberg, Jonas Öhr, and Svante Gunnarsson. A benchmark problem for robust control of a multivariable nonlinear flexible manipulator. In *Proceedings of IFAC World Congress*, Seoul, Korea, July 2008. See: `http://www.robustcontrol.org`.

Kevin L. Moore. *Iterative learning control for deterministic systems*. Springer Verlag, London, England, 1993.

Kevin L. Moore. Iterative learning control — an expository overview. *Applied and Computational Controls, Signal Processing and Circuits*, 1(1):425–488, 1998a.

Kevin L. Moore. Multi-loop control approach to designing iterative learning controllers. In *Proceedings of IEEE Conference on Decision and Control*, pages 151–214, Tampa, FL, USA, December 1998b.

Kevin L. Moore. An observation about monotonic convergence in discrete-time, P-type iterative learning control. In *Proceedings of IEEE International Symposium on Intelligent Control*, pages 45–49, Mexico City, Mexico, September 2001.

Kevin L. Moore and YangQuan Chen. On monotonic convergence of high order iterative learning update laws. In *Proceedings of IFAC World Congress*, Barcelona, Spain, July 2002.

Kevin L. Moore, YangQuan Chen, and Vikas Bahl. Monotonically convergent iterative learning control for linear discrete-time systems. *Automatica*, 41(9): 1529–1537, September 2005.

Salvatore Nicosia, Patrizio Tomei, and Antonio Tornambé. A nonlinear observer for elastic robots. *IEEE Journal of Robotics and Automation*, 4(1):45–52, 1988.

Alexander Nordström. Identifiering och reglering av industrirobot med hjälp av accelerometer. Master's thesis No. LiTH-ISY-EX-06/3785, Department of Electrical Engineering, Linköping University, Linköping, Sweden, April 2006.

Mikael Norrlöf. *Iterative Learning Control: Analysis, Design and Experiments*. Dissertation No. 653, Department of Electrical Engineering, Linköping University, Linköping, Sweden, October 2000.

Mikael Norrlöf. Iteration varying filters in iterative learning control. In *Proceedings of 4th Asian Control Conference*, pages 2124–2129, Singapore, Singapore, September 2002.

Mikael Norrlöf and Svante Gunnarsson. A model based ILC method applied to a commercial industrial robot. In *Proceedings of IFAC 6th symposium on robot control, SYROCO*, pages 477–482, Vienna, Austria, September 2000.

Mikael Norrlöf and Svante Gunnarsson. Time and frequency domain convergence properties in iterative learning control. *International Journal of Control*, 75(14):1114–1126, 2002a.

Mikael Norrlöf and Svante Gunnarsson. Experimental comparison of some classical iterative learning control algorithms. *IEEE Transactions on Robotics and Automation*, 18(4):636–641, 2002b.

Mikael Norrlöf and Svante Gunnarsson. A note on causal and CITE iterative learning control algorithms. *Automatica*, 41(2):345–350, 2005.

NyTeknik. Stopp för slängar, October 2007. Swedish technical magazine.

Henrik Olsson, Karl Johan Åström, Carlos Canudas de Wit, Magnus Gäfvert, and Pablo Lischinsky. Friction models and friction compensation. *European Journal of Control*, 4(3):176–195, December 1998.

David H. Owens and Jari Hätönen. Iterative learning control — an optimization paradigm. *Annual Reviews in Control*, 29(1):57–70, 2005.

Antonio Pascoal, Isaac Kaminer, and Paulo Oliveira. Navigation system design using time-varying complementary filters. *IEEE Transactions on Aerospace and Electronic Systems*, 36(4):1099–1114, 2000.

Burton Paul and Jacinto A. Rosa. Kinematics simulation of serial manipulators. *The International Journal of Robotics Research*, 5(2):14–31, 1986.

Minh Phan and Richard W. Longman. A mathematical theory of learning control of linear discrete multivariable systems. In *Proceedings of AIAA/AAS Astrodynamics Conference*, pages 740–746, Minneapolis, MN, USA, August 1988.

Aaron M. Plotnik and Richard W. Longman. Subtleties in the use of zero-phase low-pass filtering and cliff filtering in learning control. *Advances in the Astronautical Sciences*, 103:673–692, 1999.

James D. Ratcliffe, Jari J. Hätönen, Paul L. Lewin, Eric Rogers, Thomas J. Harte, and David H. Owens. P-type iterative learning control for systems that contain

resonance. *International Journal of Adaptive Control and Signal Processing*, 19(10):769–796, December 2005.

James D. Ratcliffe, Paul L. Lewin, Eric Rogers, Jari J. Hätönen, and David H. Owens. Norm-optimal iterative learning control applied to gantry robots for automation applications. *IEEE Transactions on Robotics*, 22(6):1303–1307, December 2006.

Wilson J. Rugh. *Linear system theory*. Prentice-Hall, Inc, Upper Saddle River, NJ, USA, 1996.

Samer S. Saab, William G. Vogt, and Marlin H. Mickle. Learning control algorithms for tracking "slowly" varying trajectories. *IEEE Transactions on Systems, Man, and Cybernetics*, 27(4):657–670, August 1997.

Angela Schöllig and Raffaello D'Andrea. Optimization-based iterative learning control for trajectory tracking. In *Proceedings of European Control Conference*, pages 1505–1510, Budapest, Hungary, August 2009.

Lorenzo Sciavicco and Bruno Siciliano. *Modelling and Control of Robot Manipulators*. Springer Verlag, London, England, 2nd edition, 2000.

Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer Verlag, Berlin Heidelberg, Germany, 2008.

SMErobot, 2010. URL: `http://www.smerobot.org`, accessed December, 2010.

Harold W. Sorenson, editor. *Kalman Filtering: Theory and Application*. IEEE Press, New York, NY, USA, 1985.

Mark W. Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot Modeling and Control*. John Wiley & Sons, Ltd, New York, NY, USA, 2006.

Abdelhamid Tayebi. Adaptive iterative learning control for robot manipulators. *Automatica*, 40(7):1195–1203, July 2004.

Abdelhamid Tayebi and Jian-Xin Xu. Observer-based iterative learning control for a class of time-varying nonlinear systems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 50(3):452–455, March 2003.

MATLAB Signal Processing Toolbox. *User's Guide ver. 6*. The MathWorks, 2010.

Marina Tharayil and Andrew Alleyne. A time-varying iterative learning control scheme. In *Proceedings of American Control Conference*, pages 3782–3787, Boston, MA, USA, June 2004.

Masaki Togai and Osamu Yamano. Analysis and design of an optimal learning control scheme for industrial robots: a discrete system approach. In *Proceedings of IEEE Conference on Decision and Control*, pages 1399–1404, Ft. Lauderdale, FL, USA, December 1985.

Patrizio Tomei. An observer for flexible joint robots. *IEEE Transactions on Automatic Control*, 35(6):739–743, June 1990.

Rob Tousain, Eduard van der Meché, and Okko Bosgra. Design strategy for iterative learning control based on optimal control. In *Proceedings of IEEE Conference on Decision and Control*, pages 4463–4468, Orlando, FL, USA, December 2001.

Lung-Wen Tsai. *Robot Analysis; The Mechanics of Serial and Parallel Manipulators*. John Wiley & Sons, Ltd, New York, NY, USA, 1999.

Ilya Tyapin, Geir Hovland, and Torgny Brogårdh. Kinematic and elastodynamic design optimisation of the 3-DOF Gantry-Tau parallel kinematic manipulator. In *Proceedings of Workshop on Fundamental issues and future research directions to parallel mechanisms and manipulators*, Quebec City, Canada, October 2002.

Masaru Uchiyama. Formulation of high-speed motion pattern of a mechanical arm by trial. *Transactions on Society of Instrumentation and Control Engineering*, 14(6):706–712, 1978. Published in Japanese.

Parishwad P. Vaidyanathan. *Multirate systems and filter banks*. Prentice-Hall, Inc, Englewood Cliffs, NJ, USA, 1993.

Jeroen van de Wijdeven and Okko Bosgra. Hankel iterative learning control for residual vibration suppression with MIMO flexible structure experiments. In *Proceedings of American Control Conference*, New York City, NY, USA, July 2007.

Casper L. van Oosten, Okko H. Bosgra, and Branko G. Dijkstra. Reducing residual vibrations through iterative learning control with application to a wafer stage. In *Proceedings of American Control Conference*, pages 5150–5155, Boston, MA, USA, June 2004.

Mark Verwoerd. *Iterative learning control – a critical review*. Dissertation, Electrical Engineering and Applied Mathematics, University of Twente, Enschede, The Netherlands, 2005.

Johanna Wallén. On robot modelling using MAPLE. Technical Report LiTH-ISY-R-2723, Department of Electrical Engineering, Linköping University, Linköping, Sweden, August 2007.

Johanna Wallén. *On Kinematic Modelling and Iterative Learning Control of Industrial Robots*. Licentiate thesis No. 1343, Department of Electrical Engineering, Linköping University, Linköping, Sweden, January 2008. Available at: `http://www.control.isy.liu.se/research/reports/LicentiateThesis/Lic1343.pdf`.

Johanna Wallén, Svante Gunnarsson, and Mikael Norrlöf. Derivation of kinematic relations for a robot using MAPLE. In *Proceedings of Reglermöte 2006*, Royal Institute of Technology, Stockholm, Sweden, May 2006.

Johanna Wallén, Mikael Norrlöf, and Svante Gunnarsson. Experimental evaluation of ILC applied to a six degrees-of-freedom industrial robot. In *Proceedings of European Control Conference*, pages 4111–4118, Kos, Greece, July 2007a.

Johanna Wallén, Mikael Norrlöf, and Svante Gunnarsson. Accelerometer based evaluation of industrial robot kinematics derived in MAPLE. In *Proceedings of Mekatronikmöte 2007*, Lund Institute of Technology, Lund, Sweden, October 2007b.

Johanna Wallén, Mikael Norrlöf, and Svante Gunnarsson. Comparison of performance and robustness for two classical ILC algorithms applied to a flexible system. Technical Report LiTH-ISY-R-2868, Department of Electrical Engineering, Linköping University, Linköping, Sweden, November 2008a.

Johanna Wallén, Mikael Norrlöf, and Svante Gunnarsson. Arm-side evaluation of ILC applied to a six-degrees-of-freedom industrial robot. In *Proceedings of IFAC World Congress*, pages 13450–13455, Seoul, Korea, July 2008b. Invited paper.

Johanna Wallén, Mikael Norrlöf, and Svante Gunnarsson. Performance and robustness for ILC applied to flexible systems. In *Proceedings of Reglermöte 2008*, Luleå University of Technology, Luleå, Sweden, June 2008c.

Johanna Wallén, Svante Gunnarsson, Robert Henriksson, Stig Moberg, and Mikael Norrlöf. ILC applied to a flexible two-link robot model using sensor-fusion-based estimates. In *Proceedings of IEEE Conference on Decision and Control*, pages 458–463, Shanghai, China, December 2009a.

Johanna Wallén, Mikael Norrlöf, and Svante Gunnarsson. Performance of ILC applied to a flexible mechanical system. In *Proceedings of European Control Conference*, pages 1511–1516, Budapest, Hungary, August 2009b.

Johanna Wallén, Mikael Norrlöf, and Svante Gunnarsson. A framework for analysis of observer-based ILC. In *Proceedings of Symposium on Learning Control at IEEE Conference on Decision and Control*, Shanghai, China, December 2009c.

Johanna Wallén, Isolde Dressler, Anders Robertsson, Mikael Norrlöf, and Svante Gunnarsson. Observer-based ILC applied to the Gantry-Tau parallel kinematic robot — modelling, design and experiments. Technical Report LiTH-ISY-R-2968, Department of Electrical Engineering, Linköping University, Linköping, Sweden, October 2010a.

Johanna Wallén, Isolde Dressler, Anders Robertsson, Mikael Norrlöf, and Svante Gunnarsson. Observer-based ILC applied to the Gantry-Tau parallel kinematic robot. Submitted to IFAC World Congress 2011, Milano, Italy, 2010b.

Johanna Wallén, Svante Gunnarsson, and Mikael Norrlöf. Some implementation aspects of iterative learning control. Technical Report LiTH-ISY-R-2967, Department of Electrical Engineering, Linköping University, Linköping, Sweden, September 2010. Submitted to IFAC World Congress 2011, Milano, Italy.

Johanna Wallén, Mikael Norrlöf, and Svante Gunnarsson. A framework for analysis of observer-based ILC. *Asian Journal of Control*, 2011. Accepted for publication in special issue of Iterative Learning Control.

Danwei Wang and Yongqiang Ye. Multi-channel learning using anticipatory ILCs. *International Journal of Control*, 77(13):1189–1199, 2004.

Danwei Wang and Bin Zhang. Extension of learnable bandwidth in iteative learning control. In *Proceedings of Symposium on Learning Control at IEEE Conference on Decision and Control*, Shanghai, China, December 2009.

Danwei Wang, Yongqiang Ye, and Chien Chern Cheah. Analysis and design of anticipatory learning control. In *Proceedings of IEEE Conference on Decision and Control*, pages 4434–4439, Maui, HI, USA, December 2003.

Lars Westerlund. *The Extended Arm of Man – A History of the Industrial Robot*. Informationsförlaget, Stockholm, Sweden, 2000.

Jian-Xin Xu. Direct learning of control efforts for trajectories with different magnitude scales. *Automatica*, 33(12):2191–2195, 1997.

Jian-Xin Xu and Zenn Z. Bien. The frontiers of iterative learning control. In Zeungnam Bien and Jian-Xin Xu, editors, *Iterative Learning Control: Analysis, Design, Integration and Applications*, pages 9–35. Kluwer Academic Publishers, Boston, MA, USA, 1998.

Jian-Xin Xu and Yanbin Song. Direct learning control of non-uniform trajectories. In Zeungnam Bien and Jian-Xin Xu, editors, *Iterative Learning Control: Analysis, Design, Integration and Applications*, pages 261–283. Kluwer Academic Publishers, Boston, MA, USA, 1998.

Jian-Xin Xu, Tong Heng Lee, and Heng-Wei Zhang. Analysis and comparison of iterative learning control schemes. *Engineering Applications of Artificial Intelligence*, 17(6):675–686, September 2004.

Jian-Xin Xu, Rui Yan, and YangQuan Chen. On initial conditions in iterative learning control. In *Proceedings of American Control Conference*, pages 220–225, Minneapolis, MN, USA, June 2006.

PhD Dissertations
Division of Automatic Control
Linköping University

M. Millnert: Identification and control of systems subject to abrupt changes. Thesis No. 82, 1982. ISBN 91-7372-542-0.

A. J. M. van Overbeek: On-line structure selection for the identification of multivariable systems. Thesis No. 86, 1982. ISBN 91-7372-586-2.

B. Bengtsson: On some control problems for queues. Thesis No. 87, 1982. ISBN 91-7372-593-5.

S. Ljung: Fast algorithms for integral equations and least squares identification problems. Thesis No. 93, 1983. ISBN 91-7372-641-9.

H. Jonson: A Newton method for solving non-linear optimal control problems with general constraints. Thesis No. 104, 1983. ISBN 91-7372-718-0.

E. Trulsson: Adaptive control based on explicit criterion minimization. Thesis No. 106, 1983. ISBN 91-7372-728-8.

K. Nordström: Uncertainty, robustness and sensitivity reduction in the design of single input control systems. Thesis No. 162, 1987. ISBN 91-7870-170-8.

B. Wahlberg: On the identification and approximation of linear systems. Thesis No. 163, 1987. ISBN 91-7870-175-9.

S. Gunnarsson: Frequency domain aspects of modeling and control in adaptive systems. Thesis No. 194, 1988. ISBN 91-7870-380-8.

A. Isaksson: On system identification in one and two dimensions with signal processing applications. Thesis No. 196, 1988. ISBN 91-7870-383-2.

M. Viberg: Subspace fitting concepts in sensor array processing. Thesis No. 217, 1989. ISBN 91-7870-529-0.

K. Forsman: Constructive commutative algebra in nonlinear control theory. Thesis No. 261, 1991. ISBN 91-7870-827-3.

F. Gustafsson: Estimation of discrete parameters in linear systems. Thesis No. 271, 1992. ISBN 91-7870-876-1.

P. Nagy: Tools for knowledge-based signal processing with applications to system identification. Thesis No. 280, 1992. ISBN 91-7870-962-8.

T. Svensson: Mathematical tools and software for analysis and design of nonlinear control systems. Thesis No. 285, 1992. ISBN 91-7870-989-X.

S. Andersson: On dimension reduction in sensor array signal processing. Thesis No. 290, 1992. ISBN 91-7871-015-4.

H. Hjalmarsson: Aspects on incomplete modeling in system identification. Thesis No. 298, 1993. ISBN 91-7871-070-7.

I. Klein: Automatic synthesis of sequential control schemes. Thesis No. 305, 1993. ISBN 91-7871-090-1.

J.-E. Strömberg: A mode switching modelling philosophy. Thesis No. 353, 1994. ISBN 91-7871-430-3.

K. Wang Chen: Transformation and symbolic calculations in filtering and control. Thesis No. 361, 1994. ISBN 91-7871-467-2.

T. McKelvey: Identification of state-space models from time and frequency data. Thesis No. 380, 1995. ISBN 91-7871-531-8.

J. Sjöberg: Non-linear system identification with neural networks. Thesis No. 381, 1995. ISBN 91-7871-534-2.

R. Germundsson: Symbolic systems – theory, computation and applications. Thesis No. 389, 1995. ISBN 91-7871-578-4.

P. Pucar: Modeling and segmentation using multiple models. Thesis No. 405, 1995. ISBN 91-7871-627-6.
H. Fortell: Algebraic approaches to normal forms and zero dynamics. Thesis No. 407, 1995. ISBN 91-7871-629-2.
A. Helmersson: Methods for robust gain scheduling. Thesis No. 406, 1995. ISBN 91-7871-628-4.
P. Lindskog: Methods, algorithms and tools for system identification based on prior knowledge. Thesis No. 436, 1996. ISBN 91-7871-424-8.
J. Gunnarsson: Symbolic methods and tools for discrete event dynamic systems. Thesis No. 477, 1997. ISBN 91-7871-917-8.
M. Jirstrand: Constructive methods for inequality constraints in control. Thesis No. 527, 1998. ISBN 91-7219-187-2.
U. Forssell: Closed-loop identification: Methods, theory, and applications. Thesis No. 566, 1999. ISBN 91-7219-432-4.
A. Stenman: Model on demand: Algorithms, analysis and applications. Thesis No. 571, 1999. ISBN 91-7219-450-2.
N. Bergman: Recursive Bayesian estimation: Navigation and tracking applications. Thesis No. 579, 1999. ISBN 91-7219-473-1.
K. Edström: Switched bond graphs: Simulation and analysis. Thesis No. 586, 1999. ISBN 91-7219-493-6.
M. Larsson: Behavioral and structural model based approaches to discrete diagnosis. Thesis No. 608, 1999. ISBN 91-7219-615-5.
F. Gunnarsson: Power control in cellular radio systems: Analysis, design and estimation. Thesis No. 623, 2000. ISBN 91-7219-689-0.
V. Einarsson: Model checking methods for mode switching systems. Thesis No. 652, 2000. ISBN 91-7219-836-2.
M. Norrlöf: Iterative learning control: Analysis, design, and experiments. Thesis No. 653, 2000. ISBN 91-7219-837-0.
F. Tjärnström: Variance expressions and model reduction in system identification. Thesis No. 730, 2002. ISBN 91-7373-253-2.
J. Löfberg: Minimax approaches to robust model predictive control. Thesis No. 812, 2003. ISBN 91-7373-622-8.
J. Roll: Local and piecewise affine approaches to system identification. Thesis No. 802, 2003. ISBN 91-7373-608-2.
J. Elbornsson: Analysis, estimation and compensation of mismatch effects in A/D converters. Thesis No. 811, 2003. ISBN 91-7373-621-X.
O. Härkegård: Backstepping and control allocation with applications to flight control. Thesis No. 820, 2003. ISBN 91-7373-647-3.
R. Wallin: Optimization algorithms for system analysis and identification. Thesis No. 919, 2004. ISBN 91-85297-19-4.
D. Lindgren: Projection methods for classification and identification. Thesis No. 915, 2005. ISBN 91-85297-06-2.
R. Karlsson: Particle Filtering for Positioning and Tracking Applications. Thesis No. 924, 2005. ISBN 91-85297-34-8.
J. Jansson: Collision Avoidance Theory with Applications to Automotive Collision Mitigation. Thesis No. 950, 2005. ISBN 91-85299-45-6.
E. Geijer Lundin: Uplink Load in CDMA Cellular Radio Systems. Thesis No. 977, 2005. ISBN 91-85457-49-3.
M. Enqvist: Linear Models of Nonlinear Systems. Thesis No. 985, 2005. ISBN 91-85457-64-7.
T. B. Schön: Estimation of Nonlinear Dynamic Systems — Theory and Applications. Thesis No. 998, 2006. ISBN 91-85497-03-7.

I. Lind: Regressor and Structure Selection — Uses of ANOVA in System Identification. Thesis No. 1012, 2006. ISBN 91-85523-98-4.

J. Gillberg: Frequency Domain Identification of Continuous-Time Systems Reconstruction and Robustness. Thesis No. 1031, 2006. ISBN 91-85523-34-8.

M. Gerdin: Identification and Estimation for Models Described by Differential-Algebraic Equations. Thesis No. 1046, 2006. ISBN 91-85643-87-4.

C. Grönwall: Ground Object Recognition using Laser Radar Data – Geometric Fitting, Performance Analysis, and Applications. Thesis No. 1055, 2006. ISBN 91-85643-53-X.

A. Eidehall: Tracking and threat assessment for automotive collision avoidance. Thesis No. 1066, 2007. ISBN 91-85643-10-6.

F. Eng: Non-Uniform Sampling in Statistical Signal Processing. Thesis No. 1082, 2007. ISBN 978-91-85715-49-7.

E. Wernholt: Multivariable Frequency-Domain Identification of Industrial Robots. Thesis No. 1138, 2007. ISBN 978-91-85895-72-4.

D. Axehill: Integer Quadratic Programming for Control and Communication. Thesis No. 1158, 2008. ISBN 978-91-85523-03-0.

G. Hendeby: Performance and Implementation Aspects of Nonlinear Filtering. Thesis No. 1161, 2008. ISBN 978-91-7393-979-9.

J. Sjöberg: Optimal Control and Model Reduction of Nonlinear DAE Models. Thesis No. 1166, 2008. ISBN 978-91-7393-964-5.

D. Törnqvist: Estimation and Detection with Applications to Navigation. Thesis No. 1216, 2008. ISBN 978-91-7393-785-6.

P-J. Nordlund: Efficient Estimation and Detection Methods for Airborne Applications. Thesis No. 1231, 2008. ISBN 978-91-7393-720-7.

H. Tidefelt: Differential-algebraic equations and matrix-valued singular perturbation. Thesis No. 1292, 2009. ISBN 978-91-7393-479-4.

H. Ohlsson: Regularization for Sparseness and Smoothness — Applications in System Identification and Signal Processing. Thesis No. 1351, 2010. ISBN 978-91-7393-287-5.

S. Moberg: Modeling and Control of Flexible Manipulators. Thesis No. 1349, 2010. ISBN 978-91-7393-289-9.

Department of Electrical Engineering
Linköping University
SE–581 83 Linköping, Sweden

Linköping 2011