ITERATIVE LEARNING CONTROL FOR LTV SYSTEMS WITH APPLICATIONS TO AN INDUSTRIAL ROBOT

Wouter Hakvoort

This research was carried out under project number MC8.03161 in the framework of the Strategic Research Program of the Materials Innovation Institute (M2i) in the Netherlands (www.M2i.nl).

Iterative Learning Control for LTV Systems
with Applications to an Industrial Robot
Hakvoort, Wouter Bernardus Johannes
ISBN 978-90-77172-44-5

Printed by Ipskamp Drukkers BV.

ITERATIVE LEARNING CONTROL FOR LTV SYSTEMS WITH APPLICATIONS TO AN INDUSTRIAL ROBOT

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. H. Brinksma,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op vrijdag 29 mei 2009 om 13:15 uur

door

Wouter Bernardus Johannes Hakvoort
geboren op 16 februari 1979
te Doetinchem

Dit proefschrift is goedgekeurd door
prof.dr.ir. J.B. Jonker, promotor
prof.ir. O.H Bosgra, promotor
dr.ir. R.G.K.M Aarts, assistent-promotor

Voorwoord

Dit proefschrift bevat de belangrijkste resultaten van vijf jaar promotieonderzoek. Het onderzoek is uitgevoerd bij de vakgroep Werktuigbouwkundige Automatisering aan de Universiteit Twente binnen het kader van het onderzoeksprogramma van het Netherlands Institute for Metals Research (NIMR), later het Materials Innovation Institute (M2i). De aanleiding voor het onderzoek was een praktisch probleem; de nauwkeurigheid van industriële robots met standaard regeling bleek onvoldoende voor het laserlassen met hoge snelheid van complexe geometriën. Het heeft mij veel voldoening gegeven om dit praktische probleem met behulp van regeltechniek te kunnen oplossen. De beschikbaarheid van een experimentele opstelling in het laboratorium was hiervoor zeer waardevol. Vaak kon ik een nieuw stuk theorie, vertaald in een regelalgoritme, nog dezelfde middag testen op deze opstelling.

De resultaten zouden niet tot stand gekomen zijn zonder de bijdrage van velen. Het M2i wil ik bedanken voor de mogelijkheid om dit onderzoek te doen binnen het grotere kader van het onderzoeksprogramma. Mijn begeleiders Ben Jonker, Ronald Aarts en Johannes van Dijk wil ik bedanken voor hun bijdrage, de discussies, het doorlezen van mijn publicaties en de prettige samenwerking. In het bijzonder wil ik Ronald Aarts bedanken voor de inzet bij de dagelijkse begeleiding. Tevens wil ik Okko Bosgra bedanken voor zijn bijdrage aan de totstandkoming van dit werk. Toon Hardeman wil ik bedanken voor de goede en plezierige samenwerking binnen ons gezamenlijke project. Bob Ransijn en Dirk Tjepkema wil ik bedanken voor de bijdrage die zij tijdens hun afstuderen aan dit onderzoek hebben geleverd. Verder wil ik alle collega's bij de vakgroep WA bedanken voor de hulp bij de experimenten, de inhoudelijke discussies en de gezelligheid tijdens de koffiemomenten, uitjes en stapavonden, die allemaal hebben bijgedragen aan de totstandkoming van dit resultaat.

Mijn vrienden, waaronder de (oud-)leden van het onafhankelijk dispuut WAZIG, wil ik bedanken voor de gezellige uren buiten het werk. Een discussie in de late uurtjes was soms een mooie gelegenheid om de relevantie van het onderzoek nog eens te overdenken. Mijn vader, moeder, zus en broer wil ik bedanken voor alle ondersteuning en interesse tijdens dit promotietraject en de studie die eraan vooraf ging. Als laatste, maar zeker niet op de laatste plaats, wil ik Katja bedanken voor al haar steun in de afgelopen jaren.

Wouter Hakvoort, Enschede, april 2009

Samenvatting

Industriële robots worden veelvuldig gebruikt vanwege de flexibele inzetbaarheid, de hoge manipulatiesnelheid en de relatief lage prijs. De toepassing van deze robots wordt echter beperkt door de matige volgnauwkeurigheid tengevolge van de lage bandbreedte van standaard industriële regelaars. Gelukkig is de repeteernauwkeurigheid van industriële robots meestal veel beter dan de volgnauwkeurigheid. Deze eigenschap kan worden benut voor het verbeteren van de volgnauwkeurigheid door het toepassen van iteratief lerend regelen (ILC). ILC verkleint de volgfout langs een traject dat herhaaldelijk wordt afgelegd door iteratief een vooruitgekoppeld stuursignaal aan te passen.

De volgnauwkeurigheid van industriële robots kan aanzienlijk worden verbeterd door met ILC de frequentiecomponenten van de volgfout boven de bandbreedte van de standaard regelaar te reduceren. Beneden de bandbreedte wordt de niet-lineaire dynamica van het mechanisme gelineariseerd door de regelaar, maar boven de bandbreedte hangt de gesloten-lus dynamica af van de configuratie van het mechanisme. Deze standsafhankelijke dynamica kan worden benaderd als lineair tijdsvariërende (LTV) dynamica voor kleine afwijkingen ten opzichte van de repeterende grote beweging. In dit proefschrift worden daarom twee ILC algoritmen voor systemen met LTV dynamica ontwikkeld.

Het norm-optimale ILC algoritme berekent iteratief het stuursignaal dat een gewogen som van de norm van de volgfout en de groei van het stuursignaal minimaliseert. De fout wordt voorspeld met behulp van een LTV dynamisch model. De berekening van het optimale stuursignaal is geformuleerd als een optimaal regelprobleem met een eindige tijd. Dit regelprobleem kan worden opgelost met behulp van een bestaand, efficiënt algoritme.

Het robuuste ILC algoritme berekent iteratief het stuursignaal dat de reductie van de volgfout optimaliseert voor een LTV dynamisch model met een gegeven modelonzekerheid. Er wordt een voldoende voorwaarde afgeleid waaronder dit stuursignaal een bepaalde reductie van de volgfout realiseert voor de slechtst mogelijke invloed van de modelonzekerheid. Deze voorwaarde houdt rekening met de LTV dynamica en de eindige lengte van het traject. De berekening van het optimale stuursignaal is geformuleerd als een dynamisch spel en de controle van de voldoende voorwaarde voor convergentie als een anti-causaal

optimaal regelprobleem. Dit anti-causale regelprobleem en het dynamische spel kunnen worden opgelost met behulp van bestaande, efficiënte algoritmen.

Convergentie analyse laat zien, dat de voorgestelde ILC algoritmen de volgfout met een instelbare convergentiesnelheid naar nul laten convergeren mits het dynamische model voldoende nauwkeurig is. Een verhoging van de convergentiesnelheid verlaagt de toelaatbare modelfout. Een te grote modelfout resulteert in divergentie van de volgfout. De toelaatbare modelfout kan worden vergroot door het toepassen van een robuustheidsfilter dat de componenten van het stuursignaal verwijdert waarop de dynamische respons niet voldoende nauwkeurig is gemodelleerd. De verwijderde componenten van het stuursignaal kunnen echter niet worden gebruikt om de fout te verkleinen, waardoor de uiteindelijke fout meestal ongelijk is aan nul.

De voorgestelde ILC algoritmen zijn geschikt voor systemen met LTV dynamica, ze zijn rekenefficiënt en verminderen de volgfout monotoon met een instelbare convergentiesnelheid. Deze unieke combinatie van eigenschappen maakt de ILC algoritmen toepasbaar in de praktijk om de volgnauwkeurigheid van industriële robots en andere systemen met LTV dynamica te verbeteren.

De prestaties van de voorgestelde ILC algoritmen zijn getest door ze toe te passen op de industriële Stäubli RX90 robot. Het referentie traject voor de positie van de robot is iteratief aangepast om de volgfout aan het uiteinde van deze robot te verminderen. Deze volgfout is gemeten met een optische sensor. De experimentele resultaten laten zien dat de volgfout aanzienlijk kan worden verkleind door het toepassen van de ILC algoritmen, vooral door gebruik te maken van een LTV model van de standsafhankelijke hoogfrequente dynamica van de robot. De reductie van de volgfout is voldoende om de robot te kunnen gebruiken voor het laserlassen van complexe geometriën met hoge snelheid.

Summary

Industrial robots are widely used in industry because of their dexterity, the high manipulation speed and the relatively low price. However, the applicability of these robots is limited by the mediocre accuracy resulting from the low bandwidth of standard industrial controllers. Fortunately, the repeatability of industrial robots is often much better than their tracking accuracy, which can be exploited to improve the accuracy by the application of Iterative Learning Control (ILC). ILC is a control technique that reduces the tracking error along a trajectory that is traced repeatedly by the iterative refinement of a feedforward signal.

The tracking accuracy of industrial robots can be improved substantially with ILC by reducing the frequency components of the tracking error beyond the low bandwidth of the standard industrial controller. Below this bandwidth the non-linear dynamics of the robot mechanism are linearised by the controller, but at higher frequencies the closed-loop dynamics depend on the configuration of the robot mechanism. These configuration dependent dynamics can be approximated as linear time-varying (LTV) for small deviations from the repetitive large-scale motion. Therefore, two ILC algorithms for systems with LTV dynamics are developed in this thesis.

The norm-optimal ILC algorithm iteratively computes the feedforward that minimises a weighted sum of the norm of the error and the growth of the feedforward. The error is predicted from an LTV dynamic model. The computation of the optimal feedforward is formulated as a finite-time optimal control problem and it is shown that this optimisation problem can be solved using an existing, computationally efficient algorithm.

The robust ILC algorithm iteratively computes the feedforward that optimises the reduction of the error for an LTV dynamic model with a given uncertainty. A sufficient condition is derived under which the feedforward reduces the error with a specified fraction for the worst case effect of the uncertainty. This condition takes the finite length of the iteration and the LTV dynamics into account. The computation of the optimal feedforward is formulated as a finite-time dynamic game and the check of the convergence condition is formulated as an anti-causal optimal control problem. It is shown that the dynamic game

and the optimal control problem can be solved using existing, computationally efficient algorithms.

Convergence analysis shows that the proposed ILC algorithms make the error converge to zero with an adjustable convergence rate if the dynamic model is sufficiently accurate. Increasing the convergence rate reduces the allowable model error. A model error that is too large results in divergence of the tracking error. The allowable model error can be increased by using a robustness filter that removes the components of the feedforward to which the dynamic response is not modelled sufficiently accurate. However, the removed components of the feedforward cannot be used to compensate for the error, which typically results in a non-zero error after convergence.

The proposed algorithms are suited for systems with LTV dynamics, they are computationally efficient and they are able to reduce the error monotonically with an adjustable convergence rate. This unique combination of properties makes the algorithms suited for improving the tracking accuracy of industrial robots and other systems with LTV dynamics in practice.

The performance of the ILC algorithms is tested experimentally by the application to the industrial Stäubli RX90 robot. The setpoints for the position of the robot are adjusted with ILC to reduce the tracking error at its end-effector, which is measured with an optical sensor. The experimental results show that the proposed ILC algorithms are able to reduce the measured tracking error substantially, especially if an LTV model of the configuration dependent high-frequency dynamics of the robot is used. The reduction of the tracking error is sufficient for the application of the robot to laser welding of complex trajectories at high speed.

Contents

Nomenclature

List of conventions

a_i	vector a at time step i
A_i	matrix A at time step i
a_i^k	vector a at time step i in iteration k
a_i^∞	limit value of vector a_i after infinite iterations
$\boldsymbol{a}$	lifted vector
$\boldsymbol{A}$	lifted matrix
a_i^k	element i of lifted vector $\boldsymbol{a}$ in iteration k
$\bar{\boldsymbol{a}}$	transformation of vector $\boldsymbol{a}$ based on the singular value decomposition of the system matrix
$\bar{\boldsymbol{A}}$	transformation of matrix $\boldsymbol{A}$ based on the singular value decomposition of the system matrix
$\hat{a}$	estimated value
$\check{a}$	optimal value
$\bar{\bar{a}}$	variable related to the TVARX model structure
$\tilde{a}$	variable in the alternative formulation of a set of equations
$\tilde{\tilde{a}}$	variable in the alternative formulation of a set of equations
A'	alternative definition of matrix A
$a^{(b)}$	vector a related to variable b
$\overrightarrow{a}$	variable related to the causal part of the robustness filter
$\overleftarrow{a}$	variable related to the anti-causal part of the robustness filter
$\mathcal{O}_{xyz}$	xyz-coordinate system
x'	direction in the local coordinate system $\mathcal{O}_{x'y'z'}$
$\|\boldsymbol{a}\|_\lambda$	λ-norm of vector a
$\|\boldsymbol{a}\|_\infty$	maximum of vector a
$\|\boldsymbol{a}\|_2$	2-norm of vector a
$\|\boldsymbol{A}\|_{i2}$	norm of matrix $\boldsymbol{A}$ induced by the vector 2-norm

List of symbols

Latin symbols

A	state-transition matrix of the state-space model
B	input matrix of the state-space model
C	output matrix of the state-space model
D	feedthrough matrix the state-space model
d	iteration-invariant disturbance
e	tracking error
f	feedforward input manipulated by ILC
$\boldsymbol{G}$	system matrix
$\boldsymbol{H}$	system matrix of the generalised plant including the learning controller
I	unit matrix of appropriate dimensions
J	objective function
$\boldsymbol{L}$	learning matrix
L	auxiliary matrix in the solution of the optimal control problem
l_i	ith component on the diagonal of $\bar{\boldsymbol{L}}$
$\boldsymbol{M}$	uncertainty pre-weighting matrix
m_i	ith component on the diagonal of $\bar{\boldsymbol{M}}$
$\boldsymbol{N}$	uncertainty post-weighting matrix
N_a	number of poles of the TVARX model
N_b	number of zeros of the TVARX model
N_c	number of delays of the TVARX model
N_e	dimension of the tracking error
N_f	dimension of the feedforward input
N_i	number of time-steps in the iteration
N_k	number of measurement series for the estimation of the TVARX model
N_n	number of parameter sets over which the TVARX model is interpolated
N_o	$\max(N_a, N_b + N_c)$
O	zero matrix of appropriate dimensions
$\boldsymbol{P}$	system matrix of the generalised plant
P	auxiliary matrix in the solution of the optimal control problem
p	input of the normalised model uncertainty
Q	weighting matrix related to the state vector
$\boldsymbol{Q}$	robustness filter for norm-optimal ILC
q	output of the normalised uncertainty or maximising input in the optimal control problem
q_i	ith component on the diagonal of $\bar{\boldsymbol{Q}}$
R	weighting matrix related to the input vector
$\boldsymbol{R}$	robustness filter for robust ILC
r	output of the robustness filter
r_i	ith component on the diagonal of $\bar{\boldsymbol{R}}$
S	non-stationary Riccati matrix

$\boldsymbol{S}$	diagonal matrix containing the singular values of the system matrix
s_i	ith component on the diagonal of $\boldsymbol{S}$
$\boldsymbol{T}$	orthogonal matrix with the right singular vectors of the system matrix
$\boldsymbol{U}$	orthogonal matrix with the left singular vectors of the system matrix
u	update of the feedforward manipulated by ILC or minimising input in the optimal control problem
V	weighting matrix related to the error
v	uncontrolled input
W	weighting matrix related to the feedforward input update
w	weight related to the feedforward input update
w	compensable sum of the error over the iterations
$\boldsymbol{X}$	matrix related to the objective function for robust ILC
x	state vector
x	a horizontal direction in the $\mathcal{O}_{xyz}$ coordinate system
y	a horizontal direction in the $\mathcal{O}_{xyz}$ coordinate system
z	the vertical direction in the $\mathcal{O}_{xyz}$ coordinate system
z	sum of the error over the iterations

Greek symbols

β	weight ratio
γ	maximum convergence ratio for robust ILC
$\boldsymbol{\Delta}$	the normalised model uncertainty ($\|\boldsymbol{\Delta}\|_{i2} < 1$)
δ_i	ith component on the diagonal of $\bar{\boldsymbol{\Delta}}$
η	co-state vector
$\boldsymbol{\Theta}$	model error matrix
θ_i	ith component on the diagonal of $\bar{\boldsymbol{\Theta}}$
λ	auxiliary co-state vector in the solution of the optimal control problem
μ	state-variable related to $\boldsymbol{M}$
ν	state-variable related to $\boldsymbol{N}$
ξ	normalised distance along the trajectory
ρ	state-variable related to $\boldsymbol{R}$
ψ	interpolation parameter for the definition of the TVARX model structure

List of abbreviations

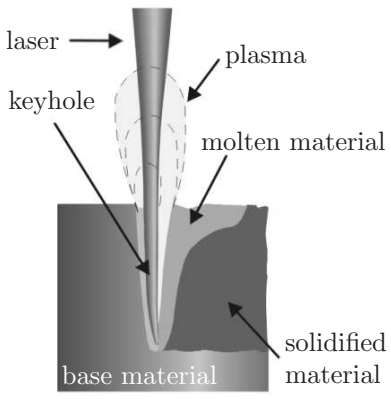
ARX	Auto-Regressive model with eXogenous inputs
CITE	Current Iteration Tracking Error
CTF	Close To Focus
DOF	Degrees Of Freedom
ILC	Iterative Learning Control
LTV	Linear Time-Varying
LTI	Linear Time Invariant
MAX	MAXimum absolute value of the subsequent variable
NILC	Norm-optimal Iterative Learning Control
PC	Personal Computer
RILC	Robust Iterative Learning Control
RMS	Root Mean Square value of the subsequent variable
SISO	Single Input Single Output
SCRC	the Sufficient Condition for Robust Convergence of the summed error for robust ILC
TVARX	Time-Varying Auto-Regressive model with eXogenous inputs

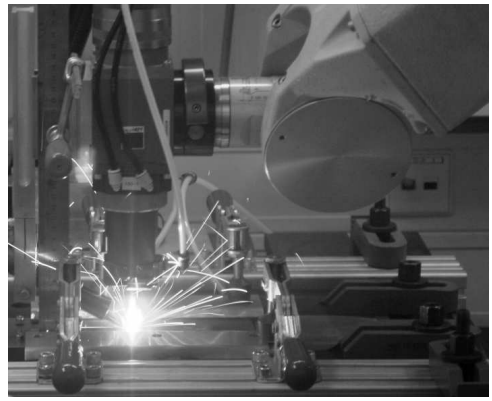
Chapter 1

Introduction

1.1 Background

Laser welding is the joining of parts through melting of the interface with a high-power laser beam. A typical application is the joining of sheet metal parts by keyhole laser welding (see figure 1.1). The keyhole is obtained by focussing the high-power laser beam (>1 kW) to a small spot (<1 mm^2). The seam can be welded at high speed (>100 mm/s) due to the high power density. Moreover, the heat affected zone is only small due to the small spot size. However, the laser beam needs to be manipulated accurately along the weld seam to obtain defect free welds. Typically, the tracking error should be less than ±0.1 mm in the





(a) Schematic overview (b) Robotised laser welding

source: (a) Materials Innovation Institute, (b) Laser Applicatie Centrum

Figure 1.1: Keyhole laser welding

directions perpendicular to the welding direction (Duley, 1998; Olde Benneker and Gales, 2007; Römer, 2002). The combination of the small allowable tracking error and the high welding speed puts high demands on the manipulation of the laser beam. The trend towards higher power densities and smaller focal spot sizes even increases the demands on the manipulator further.

From an industrial perspective it is attractive to use industrial robot arms of the elbow type for the manipulation of the laser beam (see figures 1.1(b) and 1.2). These robots give access to complex seam geometries, because they are able to manipulate the welding head in six directions (three linear and three rotational directions). Moreover, these robots are produced in large quantities, which makes them less expensive than dedicated manipulators. However, industrial robots with conventional industrial controllers do not meet the accuracy requirements imposed by many high-speed laser welding tasks. The limited tracking accuracy is the result of the low bandwidth of conventional industrial controllers. These controllers compute the setpoints for the position of the robot axes from the desired trajectory for the robot tip using a kinematic model of the robot mechanism. Next, the axes are controlled along these setpoints, which should result in the desired motion of the tip. However, errors in the kinematic model and flexibilities in the links and joints of the robot mechanism result in tracking errors at the robot tip, even if the axes trace the setpoints accurately. In particular, the excitation of resonance vibrations resulting from the flexibilities may result in high-frequency tracking errors. The frequency of the resonance vibrations depends on the load and configuration of the robot mechanism. To avoid the excitation of the resonance vibrations, the bandwidth of standard industrial controllers is taken below the first resonance frequency of the robot mechanism for the worst case load and configuration. Thereby the controlled robots only trace the low-frequency components of the trajectory setpoints, which may result in a considerable tracking error.

Fortunately, the repeatability of industrial robots is good, which means that the tracking error is approximately the same for each repetitive movement along the same trajectory. This can be exploited to improve the accuracy of the robot motion by repeatedly moving along the same trajectory and using the measured tracking error for the iterative refinement of some input signal, e.g., the trajectory setpoints or a torque feedforward. This control technique, known as Iterative Learning Control (ILC), allows reduction of repeatable tracking errors, even at frequencies beyond the bandwidth of the feedback controller. Academic research in the field of ILC started in the 1980s (Arimoto et al., 1984). Since then, numerous ILC algorithms have been proposed and many applications have been investigated. A considerable part of this research considers the application of ILC to robotic manipulators. However, most of the research focusses on improving the tracking accuracy of the robot's axes assuming a robot mechanism without flexibilities. As mentioned before, accurate tracking of the axes may still result in a considerable tracking error of the tip due to errors in the kinematic model and flexibilities in the robot mechanism. Moreover, the flexi-

bilities affect the dynamic response of the robot mechanism at high frequencies, where they induce resonance vibrations. The effect of flexibilities on the robot dynamics has to be taken into account in the design of ILC to be able to compensate the high-frequency components of the tracking error at the robot tip. Only few publications on ILC consider the reduction of the tracking error at the tip of a robot and even fewer consider the compensation of the high-frequency components of the error. The algorithms that consider the compensation of high-frequency tracking errors suffer from drawbacks that limit the applicability of the algorithms in practice. Those drawbacks will be discussed in more detail in chapter 2. A practical ILC algorithm for realising high-accuracy motion of an industrial robot requires further research.

1.2 Objective

The aim of this thesis is the development of ILC algorithms for realising high-accuracy motion at the tip of an industrial robot. In this section the requirements on the ILC algorithms following from this objective are formulated using the application to the Stäubli RX90 robot, which is used for laser welding, as a reference. Although the requirements are related to this specific example, they are formulated sufficiently general to be suited for many other applications of ILC.

Figure 1.2 shows a picture of the Stäubli RX90 robot carrying a laser welding head. The laser welding head focusses the high-power laser beam and the focus point should trace the weld seam of the product. The tracking error of the focus point with respect to the weld seam is measured with a seam-tracking sensor, which is integrated in the welding head. The robot and the sensor are described in more detail in chapter 5. As mentioned previously, the tracking error should be less than ± 0.1 mm in the directions perpendicular to the welding direction to obtain defect free welds. Measurements of the tracking error show that the Stäubli RX90 robot controlled by the standard industrial CS8 controller does not meet the required accuracy along typical weld seam trajectories at typical welding velocities (> 50 mm/s). The tracking error is mainly the result of the low bandwidth of the CS8 controller. Reducing the tracking error to the required level calls for an ILC algorithm that reduces the frequency components of the tracking error beyond this bandwidth. Below the bandwidth the closed-loop dynamics of the robot and controller are linearised by the high gain of the controller, but at higher frequencies the closed-loop dynamics depend on the configuration of the mechanism. For example, the closed-loop bandwidth and the resonance frequencies of the mechanism depend on the robot configuration. The ILC algorithm should be able to cope with these configuration dependent dynamics to reduce the tracking error to the required level. In this work the non-linear robot dynamics are approximated as linear time-varying (LTV) dynamics to confine the complexity of the ILC design. The non-linear dynamics of the

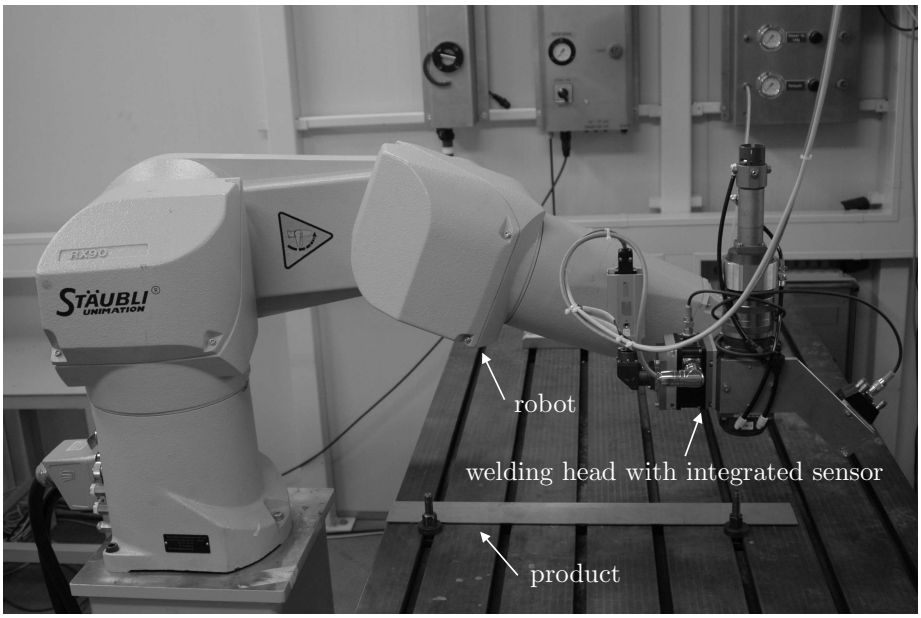


Figure 1.2: The Stäubli RX90 robot carrying a welding head with integrated seam-tracking sensor

robot can be approximated as LTV, because only small deviations from the large-scale repetitive motion are considered. Thus, it is demanded that the ILC algorithm should be applicable to a system with LTV dynamics to be able to reduce the tracking error to the required level.

The reduction of the tracking error of the Stäubli RX90 robot by the application of ILC should be realised under several constraints resulting from practical considerations. Those practical considerations put additional requirements on the ILC algorithm, which are discussed hereafter. The measurement range of the seam-tracking sensor is limited to $\pm$ 4 mm. The accuracy of the Stäubli RX90 robot with its standard controller is in the order of several millimetres and thus the sensor can just measure the tracking error if no ILC is applied. The tracking error should also be measurable during the ILC iterations, which means that the tracking error should not increase during the ILC iterations. Therefore, it is demanded that the ILC algorithm should reduce the tracking error monotonically. During the iterations in which the error is not reduced sufficiently by ILC, the Stäubli RX90 robot cannot be used for welding. Therefore it is demanded that the tracking error is reduced to the desired level in a limited number of iterations, preferably less than 10 iterations. The standard industrial CS8 controller for the Stäubli RX90 robot is tuned for reliable, durable and stable robot motion. ILC is intended as an add-on to this industrial controller

to improve the tracking accuracy. It is thus demanded that the ILC algorithm can be applied to the Stäubli RX90 robot operating in closed-loop with the standard industrial CS8 controller without adding feedback action. Finally, the ILC algorithm should be implementable on a contemporary PC to obviate the need of dedicated (expensive) computation hardware. Therefore it is demanded that the ILC algorithm is computationally efficient.

Summarising, the following requirements are imposed on the ILC algorithm:

- The ILC algorithm should be applicable to systems with LTV dynamics,
- The ILC algorithm should reduce the tracking error monotonically to a small final value,
- The ILC algorithm should reduce the tracking error to the desired level in a limited number of iterations,
- The ILC algorithm should be applicable to an industrial robot operating in closed-loop with its standard controller without adding feedback,
- The ILC algorithm should be computationally efficient.

An ILC algorithm that meets these requirements is suited for application to the Stäubli RX90 robot that is used for laser welding. Moreover, such ILC algorithm is more generally applicable, e.g., to other types of industrial robots, to other applications of industrial robots and to other (mechanical) systems with similar dynamic behaviour.

1.3 Outline

Chapter 2 starts with a discussion of some general properties of ILC and an introduction of the terminology that is used throughout the thesis. The chapter continues with a review of existing literature on ILC, in particular literature on the application of ILC to robotic manipulators is considered. Subsequently, the suitability of the existing ILC algorithms for satisfying the objective of this work is discussed. Finally it is concluded which developments are required to obtain ILC algorithms that satisfy the objective of this work. Based on those requirements two model-based ILC algorithms are developed in chapters 3 and 4.

In chapter 3 a norm-optimal ILC algorithm for LTV dynamic systems is developed. The objective of the norm-optimal ILC algorithm is formulated as the iterative minimisation of an objective function that is related to the norm of the error in the next iteration, which is predicted from an LTV dynamic model. The growth of the feedforward is limited by including the norm of the feedforward update in the objective function. After the formulation of the objective, a computationally efficient implementation of the norm-optimal ILC algorithm for LTV dynamic systems is proposed. Finally, the convergence properties of

the norm-optimal ILC algorithm are analysed and used to formulate guidelines for the tuning of its parameters.

In chapter 4 a robust ILC algorithm for LTV dynamic systems is developed. The objective of the robust-ILC algorithm is formulated as the reduction of the tracking error of a system with LTV dynamics and a specified (bounded) model uncertainty. Furthermore, a condition is derived under which the reduction of the error is guaranteed even for the worst case effect of the bounded model uncertainty. Thereafter, a computationally efficient implementation of the robust ILC algorithm for LTV dynamic systems is proposed. Moreover, an efficient algorithm for checking the convergence of the error is derived. Finally, the convergence properties of the robust ILC algorithm are analysed and used to formulate guidelines for the tuning of its parameters.

The contribution of chapters 3 and 4 is not limited to the specific application considered in this thesis. The algorithms are applicable to any system with LTV dynamics. In the subsequent part of the thesis the application of the developed algorithms to the Stäubli RX90 robot is considered. In chapter 5 the robot is described in detail and its dynamics are modelled. These models are used for the implementation of the developed norm-optimal and robust ILC algorithms. The experimental results from the application of those algorithms to the Stäubli RX90 robot are described and discussed in chapter 6. The reported results show the suitability of the proposed algorithms for satisfying the objective of this thesis.

Finally, in chapter 7, conclusions are drawn from the work presented in the preceding chapters and several directions for further research are discussed.

Chapter 2

Iterative Learning Control

In this chapter existing literature on ILC is reviewed to find ILC algorithms that are suited for the objective of this work. Previous to the literature review, in section 2.1, some general properties of ILC are discussed along with an introduction of the terminology that is used throughout the literature review and the rest of the thesis. Section 2.2 gives an overview of the different types of ILC algorithms that have been proposed in literature. The literature that considers the application of ILC to robotic manipulators is reviewed in section 2.3. In section 2.4, the advantages and disadvantages of the reviewed ILC algorithms are summarised. Two algorithms are selected that satisfy most of the requirements following from the objective of this work, though further development of those algorithms is needed to satisfy all requirements. The discussion is closed with a preview of the steps that are taken in this thesis to realise these developments.

2.1 Terminology

2.1.1 General

Iterative Learning Control (ILC) is a control technique to reduce the tracking error of systems that trace the same trajectory repeatedly or systems that are affected by the same disturbance repeatedly. In each iteration a feedforward input signal is applied which is computed by the *ILC algorithm* from recordings of the tracking error and the feedforward in the previous iteration(s). A well designed ILC algorithm makes the tracking error decrease over the iterations. The ILC algorithm is also referred to as the *learning operator*. Signals or systems that do not change over the iterations are referred to as *iteration-invariant*. Commonly, ILC is applied in addition to a feedback controller. The feedback controller stabilises the system and ILC improves the tracking performance. Mostly, the feedforward, which is updated by the ILC algorithm, is either an

addition to the output of the feedback controller or a correction of the setpoints for the feedback controlled system.

The iteration for which the feedforward is computed is referred to as the *current iteration*. The tracking error that is measured in the previous iteration(s) provides information to predict the tracking error in the current iteration at future time steps, which clearly distinguishes ILC from conventional feedback control. An ILC algorithm that computes the feedforward input at a certain time instance in the current iteration using only recordings of the feedforward and the tracking error in previous iterations up to that time instance is called a *causal ILC algorithm*. Causality limits the performance of ILC, e.g., for proper systems the performance of causal ILC is limited by Bode's integral theorem (see Norrlöf and Gunnarsson, 2005).

ILC is closely related to *repetitive control* (RC), which is also a control technique for reducing repetitive errors. The main difference between RC and ILC is the initialisation of the system's state. The state is assumed to be identical at the beginning of each iteration for ILC, while for RC the iteration starts with the state from the end of the previous repetition.

2.1.2 Classification of ILC algorithms

ILC algorithms can be classed according to the use of data from the current and previous iterations for the computation of the feedforward update. This classification is commonly used in literature. A *first order ILC algorithm* only uses recordings of the feedforward and the measured tracking error from the previous iteration. A *higher-order ILC algorithm* uses recordings of the feedforward and the measured tracking error from multiple previous iterations. A *Current Iteration Tracking Error* (CITE) ILC algorithm also uses recordings of the tracking error in the current iteration for the computation of the feedforward. CITE ILC thus includes a feedback loop and it is not a pure feedforward control technique. Higher-order ILC or CITE ILC are useful to reduce the effect of iteration-varying disturbances or dynamics; CITE ILC is able to respond directly to iteration-varying disturbances and higher-order ILC is able to average iteration-varying disturbances over multiple iterations. If all disturbances and the dynamics are iteration-invariant, then any higher-order ILC algorithm and any CITE ILC algorithm can be converted to an equivalent first-order ILC algorithm as shown by Phan et al. (2000). In this thesis it is assumed that all disturbances and the system dynamics are iteration-invariant and therefore only first-order ILC is considered.

Alternatively, ILC algorithms can be classed according to the use of model information. This classification is used for the discussion of ILC algorithms in section 2.2, but it is not commonly used in literature. *Gain-type* ILC algorithms update the feedforward with a gain times the error, the derivative of the error or the integral of the error in the previous iteration(s). The gain is selected such that the error converges for the dynamics of the controlled system. *Model-*

type ILC algorithms employ a model of the dynamics of the controlled system for the computation of the feedforward update from the error in the previous iteration(s). Commonly, the feedforward is computed from the error using some kind of inverse of the modelled dynamics. *Adaptive-type* algorithms do not only update the feedforward, but also the algorithm itself from the recordings of the feedforward and the tracking error in previous iteration(s). Mostly, adaptive-type ILC algorithms are gain-type or model-type ILC algorithms, where the gain or the model is updated after each iteration.

2.1.3 Types of convergence

Consider the sampled vector e_i^k of the tracking error, where the superscript k denotes the iteration index and the subscript i denotes the time-index with $i = 1 \dots N_i$. The error *converges* if it approaches a finite value when the iterations go to infinity, i.e., $\lim_{k\to\infty} e_i^k = e_i^\infty$ is finite for all i. The error *diverges* if the error does not approach a finite value. The limit value of the error (e_i^∞) is referred to as the *final error*. A well-designed ILC algorithm should result in convergence of the tracking error. Furthermore, a well-designed CITE ILC algorithm should also result in stable closed-loop dynamics.

In literature the convergence of the error is commonly proved by showing that some norm of the difference between the error and its final value converges to zero. The error *converges monotonically* if for every iteration this norm is smaller than the norm in the previous iteration. The ratio between the norm in one iteration and the previous iteration is referred to as the *convergence ratio* and its inverse as the *convergence rate*. Two norms are frequently used in ILC literature; the 2-norm and the λ-norm. Hereafter those norms are defined for discrete time signals, similar norms for continuous time signals are used in ILC literature as well. The 2-norm of the error in iteration k is defined as

$$\left\|e^k\right\|_2 = \sqrt{\sum_{i=1}^{N_i} e_i^{kT} e_i^k}. \tag{2.1}$$

The 2-norm equals $\sqrt{N_i}$ times the root mean square (RMS) value of the error. The λ-norm is defined as

$$\|e^k\|_\lambda = \sup_{0\le i\le N_i} \exp(-\lambda i)\|e_i^k\|_\infty, \tag{2.2}$$

where $\|\cdot\|_\infty$ denotes the maximum absolute component of a vector. The λ-norm thus weights the maximum components of the error at each time instant with a weight that decreases exponentially over time and takes the supremum of the result. The λ-norm of the error could decrease if the error decreases only slightly at low i while it increases considerably at large i. Thus, even if the λ-norm of the error converges monotonically to zero, then the error can grow very large at some time instances before it converges to zero. This property of the

λ-norm has been pointed out by several authors (Elci et al., 2002; Harte et al., 2005; Hätönen et al., 2004; Longman, 2000; Norrlöf and Gunnarsson, 2002c; Owens et al., 2000; Songschon and Longman, 2003). In this work the described convergence behaviour of the λ-norm of the error is considered unacceptable and the term *monotonic convergence* is used to refer to monotonic convergence of the 2-norm of the difference between the error and its final value.

Even monotonic convergence of the 2-norm of the error does not imply that the maximum absolute value (MAX) of the error converges monotonically. Considering the objectives of this thesis (see section 1.2), it would thus be better to demand monotonic convergence of the ∞-norm of the error than its 2-norm. Still, convergence the 2-norm of the error is demanded for the design of the ILC algorithms in this thesis, because the use of the 2-norm facilitates the computation of the optimal feedforwards for the ILC algorithms developed in chapters 3 and 4. Moreover, large errors contribute more to the value of the 2-norm than small errors due to the quadratic nature of the norm and thus the reduction of large errors is prioritised over the reduction of small errors.

2.1.4 Convergence analyses

The application of ILC to many kinds of systems has been considered in literature, e.g., continuous-time or discrete-time systems, linear time-invariant (LTI), linear time-varying (LTV) or non-linear systems. The proof of convergence of the error is mostly related to the kind of system to which ILC is applied.

The proof of convergence for ILC applied to LTI and LTV systems is sometimes based on the use of the λ-norm (e.g., Arif et al., 2003; Arimoto et al., 1984). It is shown that the application of the proposed ILC algorithm results in monotonic convergence of the λ-norm over the iterations. As discussed previously, monotonic convergence of the λ-norm may result in undesirable convergence behaviour of the error, where the error grows large at some time-instances before it converges.

The proof of convergence for ILC applied to LTI systems is often based on the *frequency domain* transform (e.g., Bukkems et al., 2005; Van Dijk et al., 2001; Kavli, 1993; De Luca et al., 1992; De Luca and Ulivi, 1992). This allows concepts from the conventional frequency domain analysis of feedback controlled LTI systems to be used for the analysis of ILC. However, the frequency domain analysis implicitly assumes that signals are periodic or their length is infinite, while ILC deals with non-periodic finite iterations. Nevertheless, the convergence of the error away from the boundaries of long iterations can be predicted reasonably well from the convergence of its frequency domain transform as discussed by, e.g., Dijkstra (2004); Longman (2000). Transients at the start and end of the trajectory should be considered carefully. Convergence of the error for all frequencies implies convergence of the 2-norm of the error by Parseval's theorem (Norrlöf and Gunnarsson, 2002c).

The proof of convergence for ILC applied to discrete-time LTI and LTV

systems is often based on a matrix description (e.g., Beigi, 1997; Phan et al., 2000). In this description all time instances of a signal in one trial are concatenated into a single vector. A linear dynamic system is represented by a matrix that relates a vector containing the system's inputs to the vector containing the system's outputs. Convergence can be derived from properties of the matrix that relates the error in one iteration to the error in the next iteration. The error converges if the spectral radius of this matrix is less than 1 and the error converges monotonically if its spectral norm is less than 1. The spectral norm is the matrix-norm induced by the 2-norm of a vector and is denoted as $\|\cdot\|_{i2}$ in this thesis. Several terms are used to refer to the matrix description, e.g., matrix form (Elci et al., 2002; Longman et al., 2003; Longman, 2000; Norrlöf and Gunnarsson, 2002c,a, 2005), super-vector notation (Hätönen et al., 2004) or lifted plant notation (Dijkstra, 2004; Harte et al., 2005; Hätönen et al., 2006; Tousain et al., 2001). In this thesis the term '*lifted*' is used. The lifted system description is used extensively in this thesis and is described in more detail in section 3.1.

The proof of convergence for ILC applied to non-linear systems is mostly based on some special property of the non-linear dynamics. Examples are positive systems, passive systems, or systems for which the adjoint dynamics equal the time-reverse dynamics. *Positive systems* (Arimoto et al., 1985; Hätönen et al., 2006; Owens and Feng, 2003) are systems where, for any input, the inner product of the input and output is larger than a finite positive constant times the norm of the input. *Passive systems* (Arimoto et al., 2000; Hamamoto and Sugie, 2002) are systems where, for any input, the inner product of the input with the output is larger than a positive constant times the norm of the output. Examples of systems with adjoint dynamics that equal the time-reverse dynamics are SISO LTI systems (Ye and Wang, 2005) and Hamiltonian systems with a Hamiltonian that is symmetric with respect to the mid-time of the iteration (Fujimoto and Sugie, 2003).

A special type of non-linear dynamics, which is often considered in ILC literature, is the dynamics of a rigid serial robot, i.e., a series of rigid bodies interconnected by actuated hinges (e.g., Bondi et al., 1998; Choi and Lee, 2000; Hamamoto and Sugie, 2002; Tayebi, 2004). The dynamics of a rigid robot are positive and passive. Furthermore the adjoint dynamics equal the time-reverse dynamics under certain conditions (Fujimoto and Sugie, 2003).

Incorrect modelling of the response of a system to part of the feedforward could lead to divergence of that part of the feedforward by the application of ILC. Two techniques are often applied to solve this problem; A *robustness filter* is used to filter out the part of the feedforward to which the response is unknown or the feedforward is limited to a certain set of *basis functions*. The consequence of the elimination of part of the feedforward is mostly that part of the error cannot be compensated, resulting in a non-zero final error. For example, if the high-frequency dynamics of a system are unknown, then a low-pass robustness filter could be applied to the feedforward, or the feedforward could be limited to

a low-frequency basis. This approach leads to convergence of the feedforward, but, because the feedforward does not contain high-frequency components, the high-frequency part of the error cannot be compensated, which typically results in a non-zero final error.

2.2 Algorithms

Although the principle of ILC is straightforward, the development of the method started only in the last decades of the twentieth century. A US patent on 'Learning control of actuators in control systems' by Garden, accepted in 1971, patented the idea to store a 'command signal' in a computer memory and to update this command signal iteratively using the error between the actual response and the desired response of the error. The first academic contribution on ILC is a Japanese paper by Uchiyama in 1978. The real start of academic research on ILC was the paper of Arimoto et al. (1984), which is generally considered to be the first academic publication on ILC in English. In this publication the concept of ILC was inspired by the human ability to learn from mistakes. Since this initial publication, ILC has been a very active research area and many algorithms and applications have been investigated. The ILC literature review of Moore (1998) lists 254 references and the literature on ILC has grown ever since. The following discussion of ILC algorithms gives an overview of the most important types of ILC algorithms found in literature. The ILC algorithms are classed according to the use of model information.

2.2.1 Gain-type ILC algorithms

The algorithm proposed in the initial publication by Arimoto et al. (1984) is a gain-type algorithm. The feedforward is updated by a gain times the derivative of the error at the same time instant in the previous iteration. Notwithstanding the simple implementation, convergence of the λ-norm of the error is proved if the gain satisfies some condition in relation to the system dynamics. For the application to a mechanical system Arimoto et al. (1984) update the feedforward current by the derivative of the velocity error. The algorithm thus requires the measurement of the acceleration, which could be noisy. Kawamura et al. (1988) propose a modification of the algorithm where the feedforward update is proportional to the derivative of the position error. Several other modifications to the original algorithm have been studied. Arimoto et al. (1985) update the feedforward with a term proportional to the error and its derivative. Arif et al. (2003); Wang (1995) update the feedforward with a term proportional to the error, its time derivative and its second order time derivative. Arimoto (1990) uses the error and its integral instead. Arif et al. (1999, 2000) add a term proportional to the tracking error of a model of the system. Driessen and Sadegh (2002) consider constraints on the feedforward and its time derivative and Xu and Yan (2003) consider the application to singular systems. A discrete

time equivalent of the algorithm of Arimoto et al. (1984) is analysed by Huang et al. (2002). In all the aforementioned publications, the convergence of the λ-norm of the error is proved, but monotonic convergence of (the 2-norm of) the error is not guaranteed.

Fang and Chow (1998); Galkowski et al. (2003); Kurek and Zaremba (1993); Kurek (2000); Li et al. (2005) propose discrete-time gain-type ILC algorithms that update the feedforward with a term proportional to the error in the previous iteration and a term proportional to the state (error) in the current iteration. These algorithms are thus CITE ILC algorithms. The evolution of the system state and the tracking error over time and over the iterations can be written as a Roesser-type model, which is a known model structure from 2D-systems theory. It is proved that the error converges to zero when the iteration number goes to infinity if the gains of the algorithm satisfy some condition in relation to the system dynamics. However, monotonic convergence of the error is not guaranteed, except for a special selection of the gains for which the error converges to zero in one trial.

Arimoto et al. (1985, 2000) show that monotonic convergence of the error can be achieved by a gain-type ILC algorithm if it is applied to a passive or positive system. Each iteration the feedforward is updated with a term proportional to the error in the output in the previous iteration. Since the dynamics of a rigid robot satisfy the passivity property its tracking can be improved by such gain-type ILC algorithm. Miyazaki et al. (1986) consider robotic systems with flexibility in the drives. The relation between the motor torque and the arm angle does not satisfy the passivity property, but the relation between the motor torque and the motor angle and the relation between the motor angle and the arm angle are passive. A two stage gain-type ILC procedure is proposed. In the inner loop the motor torque is updated to get the required motor position. In the outer loop the motor angle is updated to make the arm angle match the desired position.

Monotonic convergence of the error can also be achieved by gain-type ILC if the adjoint dynamics of the system equal the time-reverse dynamics. Each iteration the tracking error is reversed in time, multiplied by a suitable gain, applied as feedforward and the time-reverse of the resulting tracking error is the new feedforward update. This procedure is suited for self-adjoint Hamiltonian systems (Fujimoto and Sugie, 2003) and SISO LTI systems (Ye and Wang, 2005). In case the time-reversed tracking error is not acceptable as an feedforward input for the real system, a model of the system can be used to compute the feedforward update.

Frequency domain analysis can be used to design gain-type ILC algorithms that result in monotonic convergence of the error of an LTI system. Mita and Kato (1985); Togai and Yamano (1985); Wada et al. (1993) update the feedforward by a gain times the error in the previous iteration and use frequency domain analysis and a model of the system to tune the gain such that the error converges in the frequency range of interest (mostly the low-frequency range). A

robustness filter is applied to filter out the feedforward components at frequencies where the convergence condition is violated (mostly at high-frequencies). The frequency range of convergence can be extended by using a more complex transfer function instead of a single gain as proposed by, e.g., Elci et al. (2002); Longman (2000); De Luca et al. (1992). Again, frequency domain analysis can be used to tune the gains of the transfer function such that convergence of the error is obtained in the frequency range of interest.

2.2.2 Model-type ILC algorithms

Model-type ILC algorithms employ a model of the system dynamics in the ILC algorithm. These ILC algorithms are thus more complex than gain-type ILC algorithms, but model-type ILC algorithms have other advantageous properties. For example, monotonic convergence with a high-convergence rate can be realised by model type ILC.

The most straightforward implementation of model-type ILC is the computation the feedforward update that is required to eliminate the measured error by multiplication of the error with the inverse of a model of the system dynamics (see, e.g., Kavli, 1993; Lange and Hirzinger, 1995, 1999b; Markusson et al., 2002; Norrlöf and Gunnarsson, 2001; Pervozvanskii and Avrachenkov, 1997). For non-linear systems the inverse of the linearised dynamics of the model can be used (Avrachenkov, 1998; Avrachenkov and Longman, 2003). The inverse of a non-minimum-phase system can be obtained from non-causal filtering (Devasia et al., 1996; Ghosh and Paden, 2004; Markusson et al., 2002), the Zero Phase Error Tracking Control (ZPETC) algorithm (Bukkems et al., 2005; Van Dijk et al., 2001; Tomizuka, 1987) or the lifted system description (Harte et al., 2005). A pseudo-inverse of the system-dynamics can be used to handle over- or underdetermined systems (Avrachenkov and Longman, 2003). Commonly, the dynamics of the model do not exactly represent the dynamics of the real system. Using the inverse of the model to compensate for the error of the real system might result in divergence of the error if the model error is large. In most publications a bound on the model error is derived for which convergence of the error can be guaranteed. The error may diverge if this condition is violated. The allowable model error can be increased by the use of a robustness filter. Bukkems et al. (2005); Van Dijk et al. (2001); Kavli (1993); Markusson et al. (2002); Pervozvanskii and Avrachenkov (1997) apply a low-pass filter to obtain robustness to errors in the model of the high-frequency dynamics. The application of a low-pass filter also decreases the large high-frequency gain of the inverse dynamics of proper systems, which is useful to attenuate the effect of iteration-varying high-frequency load and measurement disturbances (Norrlöf and Gunnarsson, 2001). Alternative methods to decrease the high-frequency gain of the inverse dynamics are the application of an inverse Kallman filter (Lange and Hirzinger, 1995, 1999b) or regularisation of the pseudo-inverse (Avrachenkov and Longman, 2003; Ghosh and Paden, 2004).

Norm-optimal ILC (NILC) is a model-type ILC method that is based on the iterative minimisation of an objective function that is related to the 2-norm of the tracking error in the current iteration. The tracking error in the current iteration is predicted from the measurement of the error in the previous iteration(s) and a model of the dynamic response of the controlled system. Note that inversion-based ILC is a special type of NILC, because the feedforward update that minimises the 2-norm of the error for a non-singular system is equal to the error times the inverse of the system model. NILC provides an elegant method for the design of ILC for overdetermined, underdetermined and non-minimum-phase systems. Amann et al. (1996a,b, 1998); Buchheit et al. (1994); Gunnarsson and Norrlöff (2001); Gunnarsson et al. (2007); Norrlöf and Gunnarsson (2002a) propose NILC algorithms that include the feedforward (update) in the objective function, which limits the (growth of the) feedforward. The approach is similar to linear-quadratic feedback control. The minimisation of an objective function that includes the feedforward (update), yields a feedforward (update) that is equal to the error times a regularised pseudo-inverse of the system. The regularisation parameter depends on the weight on the feedforward (update) in the objective function. The convergence rate can be increased by including the prediction of the error in future iterations (Amann et al., 1998) or the feedforward in multiple previous iterations (Fang et al., 2005) in the objective function. Dijkstra (2004); Lee et al. (1999, 2000); Tousain et al. (2001) use NILC for the minimisation of the norm of the error of a system that is affected by iteration-varying noise. The noise is split in a part that should be compensated, typically the iteration-invariant part, and a part that should not be compensated, typically the iteration-varying part and/or the high-frequency part of the error. The minimisation results in an observer-based ILC algorithm, similar to a Kallman-filter. Gunnarsson and Norrlöff (2001) show that the rejection of iteration-varying disturbances is improved by including the error in multiple past iterations in the objective function. Lee et al. (2000) include constraints on the error, the feedforward, the feedforward update and the time-derivative of the feedforward in the design of NILC. The feedforward of NILC should minimise the specified objective function. This optimal feedforward is often computed using the lifted system description (see, e.g., Gunnarsson and Norrlöff, 2001; Gunnarsson et al., 2007; Lee et al., 1999, 2000; Tousain et al., 2001). The optimal feedforward is computed by multiplication of the lifted error vector with a lifted matrix that is derived from the lifted matrices associated with the system model and the weights in the objective function. The number of elements of the lifted matrices involved in the computations scales quadratically with the length of the iteration, resulting in a computationally inefficient algorithms for long iterations. The size of the optimisation problem can be reduced by using the singular value decomposition of lifted matrices as proposed by Kim et al. (2000). Another method to compute the optimal feedforward efficiently is derived from optimal control theory. The optimal feedforward is computed from the error using a series of causal and anti-causal convolutions in-

volving the state-space matrices associated with the system model, the weights in the norm and the solution of a Riccati matrix convolution. Such algorithm is proposed by Amann et al. (1996a,b, 1998); Lee et al. (1999) for the computationally efficient implementation of a CITE NILC algorithm and by Dijkstra (2004); Hakvoort et al. (2009); Kim et al. (2000) for the implementation of a first order ILC algorithm. Hatzikos et al. (2004) propose the use of a genetic algorithm for the computation of the optimal feedforward for non-linear systems. Whichever approach is used to compute the optimal feedforward for proper and non-minimum-phase systems, the resulting NILC algorithm is non-causal.

The application of model-type ILC may result in divergence of the error if the difference between the dynamics of the model and the real system is large. Several authors propose model-type ILC algorithms that explicitly take model uncertainty into account. These so-called *robust ILC* (RILC) algorithms guarantee convergence of the error for the specified bound on the model uncertainty. Amann et al. (1996c); Van Dijk et al. (2001); De Roover (1996); De Roover and Bosgra (2000) design RILC algorithms using conventional robust control design methods, which are developed for the design of robust feedback control algorithms. This approach results in causal RILC algorithms. As mentioned in section 2.1, causality puts a severe limitation on the performance of the ILC controller. Van Dijk et al. (2001) show that an inversion-based ILC algorithm outperforms such causal RILC algorithm. Van de Wijdeven and Bosgra (2007a) do not impose the causality constraint on their design of RILC for LTI systems, yielding an ILC algorithm with better performance. The algorithm results in guaranteed convergence of the error for systems with limited bounded uncertainty, though convergence of the error for systems with large bounded uncertainty cannot be guaranteed. Moore et al. (2005) employ the lifted system description for the design of ILC with robust performance in the presence of iteration-varying disturbances and model uncertainty. The approach results in a higher-order ILC algorithm that is not restricted to be causal. However, the use of the lifted description makes the implementation computationally inefficient.

2.2.3 Adaptive-type ILC algorithms

Gain-type and model-type ILC algorithms iteratively refine the feedforward using the same learning operator in each iteration, whereas each iteration yields new information for the refinement of the learning operator. Several ILC algorithms have been proposed in literature that update the learning operator as well. These adaptive-type ILC algorithms are discussed in this subsection. Besides, some adaptive feedback algorithms that improve the performance of systems tracing the same trajectory repeatedly have been proposed in literature. These algorithms are closely related to ILC and these algorithms are discussed in this subsection as well.

Hätönen et al. (2004); Owens and Feng (2003) propose an adaptive gain-type ILC algorithm, where the feedforward is updated by a gain times the error in

the previous iteration(s) one time step ahead. The value of the gain is adapted such that it minimises a weighted quadratic norm of the gain and the error in the next iteration. The error converges monotonically if the initial error is small and a perfect model of the system dynamics is available for the estimation of the error in the next iteration. Furthermore, the error converges to zero if the controlled system is positive. Several modifications have been proposed to improve the properties of the algorithm. Hätönen et al. (2004) propose a similar CITE ILC algorithm with improved computational efficiency. Hätönen et al. (2006) propose a modification that results in convergence of the error to zero for non-positive systems. Harte et al. (2005) propose a model-type version, where the feedforward update is obtained from the multiplication of the error with the inverse dynamics of a model and an adapted gain. The error reduces to zero if the multiplicative error between the real system dynamics and the model is positive. The proposed adaptive gain-type ILC algorithms are simple and robust, but the convergence rate slows down considerably when the error decreases.

A different kind of adaptive-type ILC algorithm is proposed by Norrlöf and Gunnarsson (2002a); Saab (2004). They assume that the measurement of the error is affected by iteration-varying noise and aim at minimising the norm of the undisturbed error. The algorithms result in a kind of Kallman-estimator for the undisturbed error. The part of the error that is compensated depends on the uncertainty in its estimation. The uncertainty in the estimation of the error decreases over the iterations and thus the compensated part of the error increases.

Another kind of adaptive-type ILC algorithms are model-type ILC algorithms that update the model using the measured dynamic response of the system to the feedforward applied in the past iteration(s). Kang et al. (2005) iteratively improve the estimation of the system's direct feedthrough. The estimated direct feedthrough is used for the implementation of an ILC algorithm resulting in convergence of the λ-norm of the error. The adaptive-type ILC algorithm proposed by Beigi (1997) employs the lifted system description. In each iteration the lifted system matrix of the model is updated with the generalised secant method and the updated lifted system matrix is used to update the feedforward. Longman et al. (2003) consider the numerical conditioning of the updating procedures and propose several modifications to improve the numerical conditioning of the update of the feedforward and the lifted system matrix. The method of Beigi (1997) becomes computationally inefficient for long trajectories because of the large dimensions of the lifted system matrix. Frueh and Phan (2000) propose a more efficient procedure. The feedforward is constructed from an orthogonal set of basis-vectors and in each iteration the response of the system to one of the basis-vectors is learned. In the subsequent iterations the learned dynamic response is used to optimise the contribution of the basis-vector to the feedforward that minimises the error. The number of independent basis-vectors is equal to the length of the lifted vector of the

feedforward. So, theoretically, the number of iterations to obtain the optimal feedforward is equal to the length of the lifted vector of the feedforward. Fortunately, most of the error can often be reduced in a limited number of iterations using a limited set of basis-vectors. The efficiency of the estimation of the system dynamics can also be improved by exploiting knowledge on the structure of the system dynamics. For example, Beigi (1997); Gorinevsky et al. (1997) estimate the dynamics of a time-invariant system, Oh et al. (1988) estimate the dynamics of a time-varying finite state system using full state measurements and Markusson et al. (2002) identify the dynamics of a finite-state system using system identification techniques. Another similar model-based adaptive-type ILC algorithm is the model reference adaptive learning controller proposed by Phan and Frueh (1999). The feedforward update is computed from the measured tracking error and the tracking error predicted by a model. The measured error converges as desired if the system model is able to predict the measured error accurately. The difference between the predicted error and the measured error is minimised in a separate step in which the system model is updated.

Choi and Lee (2000) combine ILC and adaptive control for the application to a rigid robot. The adaptive control part learns the uncertain parameters of the dynamic model of a robot and the learned parameters are transferred from one iteration to the next to compute the feedforward that compensates for the tracking error resulting from the robot dynamics along the repetitive trajectory. The ILC part learns a torque feedforward to compensate for iteration-invariant torque disturbances. The algorithm proposed by Hamamoto and Sugie (2002) only updates the parameters of the dynamic model of a rigid robot. The model is used to generate the feedforward for the next iteration, in which the robot may track a totally different trajectory. The learned parameters are thus reused, even if a different trajectory is traced in the next iteration. This method thus overcomes the adverse property of conventional ILC that can only reduce the tracking error if the same trajectory is traced repeatedly. Several other strategies were proposed in literature to overcome this property of ILC. Arif et al. (2002); Cheah (2001); Gorinevsky (1995); Gorinevsky et al. (1997); Lange and Hirzinger (1995, 1999a,b) apply conventional ILC to learn the feedforward that compensates for the error along several different trajectories. The learned feedforwards are used to construct a feedforward controller that is able to generate the feedforward that reduces the error along new trajectories.

Polushin and Tayebi (2004); Tayebi (2004); Tayebi and Islam (2006) propose adaptive feedback control algorithms for rigid robots that repeatedly trace the same trajectory. A PD-feedback controller is combined with a non-linear feedback controller of which a time-varying parameter is updated between the iterations. A similar algorithm is proposed by Xu and Xu (2004) for a special class of non-linear systems with full-state measurements. The parameters of the system are assumed to be unknown and iteration-invariant, though the trajectory may be iteration-varying.

French et al. (1999); French and Rogers (2000); French et al. (2001); Owens

and Munde (2000); Owens et al. (2000) propose adaptive feedback control algorithms for time-invariant SISO systems that trace the same trajectory repeatedly. The algorithms are based on conventional adaptive control concepts like universal adaptive stabilisation theory and tuning functions design. At the end of each iteration a set of parameters is transferred to the next iteration, resulting in a linear rate of convergence of the tracking error over the iterations. The adaptive algorithms are able to cope with large uncertainty in the model dynamics. The minimum-phase condition, which is conventionally imposed by the adaptive control algorithms, is not imposed by the adaptive-type ILC algorithms, because of the repetitive tracking. Nevertheless some other conditions are imposed on the system dynamics, e.g., the system should be SISO, have a known relative degree and an upper bound on the plant order should be known.

2.3 Application of ILC to robots

A considerable part of the literature on ILC considers the application to robotic manipulators. For two reasons ILC is an effective method to improve the tracking of industrial robots. In the first place, the repeatability of industrial robots is often much better than the tracking accuracy. Secondly, most robots in industry perform the same task repeatedly. Many different types of ILC algorithms are proposed for the application to robotic manipulators, including gain-type, model-type and adaptive-type ILC. Appendix A provides a list of publications that consider the application of ILC to robotic manipulators. In most papers it is shown by simulation or experiments, that ILC is able to reduce the tracking error of the robot at least one order of magnitude.

In some of the publications ILC algorithms for LTI systems are applied to robotic manipulators. The tracking error of a single robot link or the low-frequency components of the tracking error of a multi-axis robot can be reduced by those algorithms, because the dynamics of a single link and the low-frequency dynamics of a multi-axis robot operating in closed-loop can be approximated as LTI. However, at higher frequencies the dynamics of a multi-axis robot cannot be approximated as LTI. In most publications that describe the application of ILC to multi-axes robots, the robot is considered as a series of interconnected rigid bodies and the flexibilities in the robot mechanism is neglected. It is shown that the application of ILC results in convergence of the error for the dynamic equations of a series of rigid bodies or for a more general class of systems like LTV, passive or positive systems. The location of the tip of a rigid robot is related to the joint angles via a (known) kinematic transformation. Thus, reduction of the tracking error of the joints suffices to reduce the tracking error at the robot's end-effector. Real robots contain flexible components like joint and drive flexibility (see Hardeman, 2008). As a result, accurate tracking of the joints does not necessarily imply accurate tracking of the tip, which is clearly illustrated for an industrial robot by Norrlöf (2000). The flexibilities

only result in quasi-static deformations at low-frequencies. Deman et al. (1999); Lange and Hirzinger (1999b) show that this effect can be compensated effectively by assuming a rigid robot and measuring the position of the tip. However, the flexibilities affect the dynamic response of the robot at high-frequencies substantially as they induce mechanical resonance vibrations. Neglecting this effect for the design of the ILC algorithm may result in divergence of the high-frequency part of the error. Therefore, in most applications of ILC to a real robot the feedforward is limited to its low-frequency components, either by the application of a low-pass filter or by constructing the feedforward input from a set of low frequency basis functions.

Only few publications explicitly consider the effect of flexibilities on the robot dynamics for the design of the ILC algorithm. De Luca and Ulivi (1992); Velthuis et al. (1996); Wada et al. (1993) consider drive flexibility and demonstrate the necessity of a low-pass robustness filter for realising convergence of the error, though only the low-frequency part of the error is compensated. ILC algorithms for the compensation of the high-frequency components of the tracking are proposed in literature as well. Miyazaki et al. (1986) propose a gain-type ILC algorithm for robots with rigid links and flexibility in the transmission. The motion of the motor-side and the arm-side of the transmission is measured and improved in two alternating modes. Simulations on a 2DOF robot demonstrate the (slow) convergence of the tracking error. Cheng and Wen (1993); Gorinevsky (1995); Gorinevsky et al. (1997); Guglielmo and Sadegh (1996) measure the response of the robot tip to a set of basis-functions for the feedforward, while the robot is moving repeatedly along the same trajectory. The measured response is used for the implementation of model-type ILC algorithms that reduce the error including its high-frequency components. Results from simulation and experiments show that the error can be reduced successfully with this method. However, many experiments are needed to learn the robot's response for a set of basis-function with a wide frequency range and the set of basis-functions grows with the length of the iteration resulting in demanding computations for long iterations. Gunnarsson et al. (2007) propose a more efficient method for reducing the tracking error including its high-frequency components for a single link with drive flexibility. The tracking error at the link side of the transmission is estimated from measurements of the motor position and the measured acceleration of the arm. An LTI dynamic model of the single robot link is identified and this model is used to implement a computationally efficient ILC algorithm for LTI systems. It is shown that the method effectively reduces the tracking error at the tip, including its high-frequency components.

Thus, a wide range of ILC algorithms for robotic manipulators are proposed in literature. Most of them can be used only for reducing the low-frequency part of the tracking error. For the compensation of the high-frequency part of the tracking error the effect of flexibilities in the robot mechanism should be taken into account. ILC algorithms for the compensation of the high-frequency part of the tracking error of industrial robots have been proposed, but none of

them is computationally efficient, results in fast convergence of the error and is applicable to the configuration dependent dynamics of a multi-axis industrial robot.

2.4 Discussion

In section 1.2 the requirements on the ILC algorithm following from the objectives of this work are formulated. The ILC algorithm should be able to cope with the dynamics of an industrial robot, which are configuration dependent and significantly affected by the flexibilities in the robot mechanism. Furthermore, the algorithm should result in monotonic convergence of the error with a high convergence rate, the algorithm should be computationally efficient and not introduce any feedback action.

The literature that considers the application of ILC to robotic manipulators is reviewed in section 2.3. In none of the discussed publications an ILC algorithm is proposed that satisfies all the aforementioned requirements. Therefore, the potential of the different types of ILC algorithms described in section 2.2 for satisfying these requirements is evaluated in subsection 2.4.1 and it is discussed which developments are needed to obtain ILC algorithms that satisfy all requirements. In subsection 2.4.2 the steps taken in the subsequent chapters to develop such algorithms are outlined.

2.4.1 Existing ILC algorithms

Gain-type ILC algorithms are computationally efficient. No model is needed for the implementation of these algorithms, although model information is needed to tune the gains of the algorithm such that the error converges. Mostly convergence of the λ-norm of the error is proved for gain-type ILC algorithms, which, as discussed in subsection 2.1.3, may result in a growth of the 2-norm of the error in the initial iterations. Monotonic convergence of the error can be realised with gain-type ILC for systems with LTI, positive or passive dynamics, but the dynamics of a feedback controlled industrial robot are in general neither LTI, passive nor positive. Gain-type ILC is thus not suited to satisfy all requirements following from the objective of this work.

Model-type ILC can realise monotonic convergence of the tracking error with a high convergence rate, provided that the system dynamics are modelled sufficiently accurate. This has been demonstrated in literature by several successful applications of model-type ILC to robotic manipulators. Mostly, an LTI model of the closed-loop low-frequency dynamics of an industrial robot is used to reduce the low-frequency part of the tracking error. It is also shown that an LTI model of the dynamics of a single robot link can be used to reduce the tracking error of a single link including the high-frequency components. The advantage of using an LTI model for ILC is that computationally efficient algorithms of

model-type ILC for LTI systems are available. However, the closed-loop high-frequency dynamics of a multi-link industrial robot are not LTI. Those dynamics can be approximated as LTV for the design of ILC, because only small deviations from the iterative large scale motion are considered. Model-type ILC algorithms for LTV dynamics are proposed in literature, but these are not computationally efficient. The development of a computationally efficient model-type ILC algorithm for LTV systems would yield an algorithm that satisfies the requirements of this thesis, provided that this algorithm does not involve any feedback action.

A diverse range of adaptive-type ILC algorithms is proposed in literature. Most adaptive-type ILC algorithms are not suited for the application considered in this thesis, because they result in a low convergence rate, require feedback action or they are only suited for LTI dynamics. Two adaptive-type ILC methods are useful for the objective of this thesis. Firstly, an adaptive model-type ILC algorithm that updates the model during the iterations would be useful as it can be used to prevent divergence of the error as a result of model errors. Secondly, an adaptive-type ILC algorithm that uses the feedforwards learned by ILC to construct a feedforward controller that is able to reduce the tracking error along new trajectories would be useful. Both methods are adaptive add-ons to model-type ILC. A suitable model-type ILC algorithm should be developed prior to the addition of these adaptive elements. This thesis focusses on the development of a suitable model-type ILC algorithm. The development of additional adaptive elements is suggested for future research.

2.4.2 Developments in this thesis

In the previous subsection it is concluded, that model-type ILC is the most suited approach for realising the objectives of this work, because monotonic convergence of the tracking error with a high convergence rate can be realised with this approach. The development of a computationally efficient implementation of model-type ILC for LTV systems is needed to satisfy the other requirements imposed by the objectives of this work as formulated in section 1.2. The development of such model-type ILC algorithms is considered in the next two chapters.

In chapter 3 the design of an computationally efficient NILC algorithm for LTV systems is considered. NILC is a straightforward method for the design of ILC, that is considered frequently in literature (see section 2.2). However, the effect of modelling errors is not taken into account in the design of NILC, which may lead to divergence of the error. In chapter 4 the design of a computationally efficient RILC algorithm for LTV systems is considered. The developed RILC algorithm aims at realising convergence of the error with a specified convergence rate, even for the worst case effect of the model uncertainty. Thus, in contrast to NILC, the design of RILC takes the effect of model uncertainty into account. Inversion-based ILC, which is the third type of model-type ILC discussed in

section 2.2, is a special case of NILC and this method is not considered separately in this thesis.

To realise monotonic convergence of the error, the objectives for the NILC and RILC algorithms are formulated as the minimisation of objective functions related to the 2-norm of the error. The feedforwards of the ILC algorithms follow from the minimisation of these objective functions for a prediction of the error in the current iteration. The prediction of the error is based on a dynamic model and the measurement of the error in the previous iteration, such that no real-time feedback of the error is needed for the implementation of the ILC algorithms. Computationally efficient algorithms for the computation of the optimal feedforward update are obtained by the derivation of algorithms of which the number of computational operations scale linearly with the length of the iterations. Finally, the convergence properties of the algorithms are investigated, in particular the convergence rate, the robustness to modelling errors and the final error. The results from this analyses are used to formulate guidelines for selection of the parameters of the ILC algorithms that results in a high convergence rate and a small final error.

The suitability of the developed algorithms for realising the objective of this thesis is evaluated experimentally from the applications to the industrial Stäubli RX90 robot. The experimental setup is described in detail in chapter 5. This chapter also describes the dynamic model of the robot that is used for the implementation of the model-based ILC algorithms. The experimental results from the applications of the developed algorithms to the Stäubli RX90 robot are reported in chapter 6. Finally, in chapter 7, it is discussed whether the requirements imposed by the objectives of this work are satisfied by the developed ILC algorithms.

Chapter 3

Norm-optimal ILC

Norm-optimal ILC (NILC) is a model-type ILC method for the iterative minimisation of an objective function that is related to the norm of the predicted error. The prediction of the error follows from the error that is measured in the previous iteration and the change of the error resulting from the applied feedforward, which is predicted using a model of the dynamic response of the controlled system. The growth of the feedforward can be limited by including the norm of the feedforward update in the objective function.

Section 3.1 describes the linear time-varying system dynamics considered in this chapter and the lifted system description is defined. The objective function, defining the optimal feedforward update for NILC, is formulated in section 3.2. Algorithms for the computation of the optimal feedforward update are described in section 3.3. A computationally efficient algorithm of NILC for LTV systems is presented in subsection 3.3.2. In section 3.4 the convergence properties of the proposed NILC algorithm are analysed and used to formulate guidelines for the selection of the parameters of NILC.

The formulated objective function and the convergence analysis in this chapter have been presented in a similar form in literature (see, e.g., Amann et al., 1996a; Lee et al., 2000). The main contribution of this chapter is the derivation of a computationally efficient NILC algorithm for LTV systems.

3.1 System description

The computation of the feedforward of NILC is based on the minimisation of an objective function that is related to the prediction of the error. In this section a model-based prediction for the error in iteration $k+1$ is derived, which is based on the measurement of the error in iteration k. The error in iteration $k+1$, which is denoted as e_i^{k+1}, depends on the feedforward applied in iteration $k+1$, which is denoted as f_i^{k+1}. The subscript i is the time index and the number of time-samples in each iteration is N_i. The dynamics of the controlled system are

assumed to be linear time-varying, iteration-invariant and strictly proper. Thus, the effect of the feedforward in iteration $k+1$ on the error can be modelled by the following state-space equations

$$x_{i+1}^{k+1} = A_i x_i^{k+1} + B_i f_i^{k+1}, \tag{3.1a}$$
$$x_1^{k+1} = O, \tag{3.1b}$$
$$e_i^{k+1} = C_i x_i^{k+1} + d_i, \tag{3.1c}$$

where x_i^k is the state vector and $\{A_i, B_i, C_i\}$ are the time-varying state-space matrices of the model. Note that the error equals d_i if no feedforward is applied. This vector d_i thus describes the initial tracking error along the repetitively traced trajectory, including the effect of all disturbances and the effect of any non-zero initial state. The vector d_i is assumed to be trial-invariant, i.e., it is assumed that the error is not affected by iteration-varying effects.

The analysis of the iterative behaviour of discrete-time systems controlled by ILC is facilitated by the use of the lifted system description (see subsection 2.1.4). In this description, all time samples of a signal in one iteration are concatenated in a so-called lifted vector. For example, the lifted vector of the feedforward in iteration $k+1$ is defined as

$$\boldsymbol{f}^{k+1} = \begin{bmatrix} f_1^{k+1T} & f_2^{k+1T} & \dots & f_{N_i}^{k+1T} \end{bmatrix}^T. \tag{3.2}$$

In the lifted system description a dynamic system is represented by a matrix that maps the lifted vector of its inputs to the lifted vector of its outputs. The lifted system equation corresponding to the state-space equations (3.1) is

$$\boldsymbol{e}^{k+1} = \boldsymbol{G}\boldsymbol{f}^{k+1} + \boldsymbol{d}, \tag{3.3}$$

where the lifted system matrix $\boldsymbol{G}$ is expressed in terms of the state-space matrices $\{A_i, B_i, C_i\}$ as

$$\boldsymbol{G} = \begin{bmatrix} O & O & \dots & O & O \\ C_2B_1 & O & \dots & O & O \\ C_3A_2B_1 & C_3B_2 & \ddots & O & O \\ C_4A_3A_2B_1 & C_4A_3B_2 & \ddots & O & O \\ \vdots & \vdots & \ddots & \ddots & O \\ C_{N_i}\prod_{i=N-1}^{2} A_iB_1 & C_{N_i}\prod_{i=N-1}^{3} A_iB_2\dots & \dots & C_{N_i}B_{N_i-1} & O \end{bmatrix}. \tag{3.4}$$

Lifted matrices and lifted vectors are assumed to be real-valued in this thesis.

The ILC algorithm updates the feedforward $\boldsymbol{f}^k$ using measurements of the error $\boldsymbol{e}^k$ to compensate for the effect of the iteration-invariant disturbance $\boldsymbol{d}$. ILC can be considered as a feedback controller operating on the system in the

iteration-domain, in contrast to a conventional feedback controller that operates on a system in the time-domain. The internal model principle states that if a feedback system is to reject a certain disturbance, it should contain a model of the mechanism that generates the disturbance. A constant disturbance can be generated by an integrator. According to the internal-model principle the ILC algorithm should thus contain an integrator over the iterations to compensate for the iteration-invariant disturbance $\boldsymbol{d}$. This is realised by using the following feedforward update equation

$$\boldsymbol{f}^{k+1} = \boldsymbol{f}^{k} + \boldsymbol{u}^{k+1}, \tag{3.5}$$

where $\boldsymbol{u}^{k+1}$ denotes the feedforward update for iteration $k+1$. Combining this update equation with system equation (3.3) for iterations k and $k+1$, yields the following expression for the error in iteration $k+1$

$$\boldsymbol{e}^{k+1} = \boldsymbol{G}\boldsymbol{u}^{k+1} + \boldsymbol{e}^{k}. \tag{3.6}$$

The error in iteration $k+1$ is predicted using this expression. The system matrix of the real system is assumed to be unknown and the prediction of the error, which is denoted as $\hat{\boldsymbol{e}}^{k+1}$, is obtained by replacing the system matrix $\boldsymbol{G}$ by the estimated system matrix $\hat{\boldsymbol{G}}$, yielding

$$\hat{\boldsymbol{e}}^{k+1} = \hat{\boldsymbol{G}}\boldsymbol{u}^{k+1} + \boldsymbol{e}^{k}. \tag{3.7}$$

In section 3.3 this equation for the error estimate is used for the derivation of the optimal feedforward update. The difference between the error estimate and the real error resulting from the application of this feedforward depends on the difference between the real system dynamics and the estimated system dynamics. This difference is expressed by the additive model error matrix $\boldsymbol{\Theta}$, which is defined as

$$\boldsymbol{\Theta} = \boldsymbol{G} - \hat{\boldsymbol{G}}. \tag{3.8}$$

The additive model error matrix is used in section 3.4 for the analysis of the effect of a model error on the convergence of the error.

3.2 Objective

The objective of NILC is the monotonic reduction of the tracking error. This could be realised by the iterative application of the feedforward that minimises the 2-norm of the error in the next iteration as predicted by the model. However, this NILC strategy might lead to large feedforward updates if the modelled dynamic response of the tracking error to certain components of the feedforward is small, e.g., the response to the feedforward components at the anti-resonance frequencies or the response to the high-frequency components of the feedforward for a proper system model. Amann et al. (1996a,b, 1998) propose a strategy to overcome this problem, where the objective of NILC is defined as the iterative

minimisation of a weighted sum of the 2-norm of the error and the 2-norm of the applied feedforward update. This way, the tracking error is reduced while the growth of the feedforward is limited. This strategy is adopted in this work and thus the optimal feedforward update for NILC is defined as

$$\breve{\boldsymbol{u}}^{k+1} = \arg\min_{\boldsymbol{u}^{k+1}} {J^{(n)}}^{k+1}, \tag{3.9}$$

where ${J^{(n)}}^{k+1}$ is the following objective function

$${J^{(n)}}^{k+1} \triangleq \hat{\boldsymbol{e}}^{k+1T}\boldsymbol{V}\hat{\boldsymbol{e}}^{k+1} + \boldsymbol{u}^{k+1T}\boldsymbol{W}\boldsymbol{u}^{k+1}. \tag{3.10}$$

Matrices $\boldsymbol{V}$ and $\boldsymbol{W}$ are block-diagonal lifted matrices corresponding to the time-varying weights V_i and W_i. The effect of these weights on the convergence rate and the robustness of the algorithm is analysed in section 3.4. The error estimate $\hat{\boldsymbol{e}}^{k+1}$ is obtained from equation (3.7). Algorithms for the solution of the optimal feedforward update $\breve{\boldsymbol{u}}^{k+1}$ are considered in the next section.

3.3 Solutions of the optimal feedforward update

The optimal feedforward update should minimise the objective function in equation (3.10) according to equation (3.9) for the system model in equation (3.7). In this section two algorithms to solve the optimal feedforward are derived, yielding two algorithms for NILC.

An explicit expression for the optimal feedforward update can be derived using the lifted system description, which has been demonstrated previously by Lee et al. (2000). The derivation of this lifted expression for the optimal feedforward update is described in subsection 3.3.1 and the expression is used for the analysis of the convergence properties of NILC in section 3.4. However, the computation of the feedforward update from the lifted expression requires computations with lifted matrices, which makes the algorithm computationally demanding for long iterations.

A more efficient algorithm for the computation of the optimal feedforward update can be derived from the lifted algorithm by using the state-space matrices associated with the lifted system matrix, which has been demonstrated by Dijkstra (2004) for a system with LTI dynamics. An extension of this algorithm to LTV systems is derived by the author and is published previously (Hakvoort et al., 2009). An alternative computationally efficient algorithm for the computation of the optimal feedforward for NILC is proposed by Amann et al. (1996a). Their algorithm uses the state-space representation of the system and the derivation is based on optimal control theory. However, the algorithm proposed by Amann et al. (1996a) uses measurements of the error from the current iteration for the computation of the feedforward and thus it does not satisfy the requirements on the ILC algorithm formulated in section 1.2. A similar algorithm that only uses measurements of the error from the previous iteration

is proposed in subsection 3.3.2. This algorithm is comparable to the algorithm proposed by Yang et al. (2003), though it is developed independently. Yang et al. (2003) apply the algorithm to the control of the temperature of a wafer. This application is quite different from the application considered in this work, which made the publication unknown to the author at the time of development of the algorithm in section 3.3.2. The applicability to both the motion control of a robot and the temperature control of a wafer shows the wide range of possible applications of the algorithm. The algorithm proposed in subsection 3.3.2 is computationally efficient, suited for LTV systems and uses no current iteration data. These properties meet the requirements on the algorithm imposed by the objectives of this work (see section 1.2). The other requirements on the ILC algorithm following from the objective of this work concern the convergence of the error. The convergence properties of NILC are analysed in section 3.4. The suitability of the proposed NILC algorithm for the objectives of this work is evaluated experimentally by the application to the Stäubli RX90 robot. The results of these experiments are presented in chapter 6.

Summarising, subsection 3.3.1 describes the derivation of the optimal feedforward update from the lifted system equations, which yields an explicit expression for the optimal feedforward update, and subsection 3.3.2 describes the derivation of the optimal feedforward update using optimal control theory, yielding an efficient algorithm for the computation of the optimal feedforward update.

3.3.1 Solution using the lifted description

In this section the optimal feedforward update for NILC is solved using the lifted system description. Substitution of lifted equation (3.7) for error estimate in lifted equation (3.10) for the objective function yields

$$J^{(n)k+1} = \left(\hat{\boldsymbol{G}}\boldsymbol{u}^{k+1} + \boldsymbol{e}^k\right)^T \boldsymbol{V}\left(\hat{\boldsymbol{G}}\boldsymbol{u}^{k+1} + \boldsymbol{e}^k\right) + \boldsymbol{u}^{k+1T}\boldsymbol{W}\boldsymbol{u}^{k+1}. \tag{3.11}$$

This expression depends only on the error in the previous iteration $\boldsymbol{e}^k$, which is measured, and the feedforward update $\boldsymbol{u}^{k+1}$, which has to be computed. The objective function has a minimum with respect to $\boldsymbol{u}^{k+1}$ in case its second derivative with respect to $\boldsymbol{u}^{k+1}$ is positive definite, i.e.,

$$2\hat{\boldsymbol{G}}^T\boldsymbol{V}\hat{\boldsymbol{G}} + 2\boldsymbol{W} > 0. \tag{3.12}$$

This condition can be satisfied by selecting positive definite matrices $\boldsymbol{W}$ and $\boldsymbol{V}$. The minimising feedforward update is obtained by equating the derivative of $J^{(n)k+1}$ with respect to $\boldsymbol{u}^{k+1}$ to zero. This yields the following lifted expression for the optimal feedforward update

$$\check{\boldsymbol{u}}^{k+1} = -\boldsymbol{L}\boldsymbol{e}^k, \tag{3.13}$$

where the learning filter $\boldsymbol{L}$ is defined as

$$\boldsymbol{L} = \left(\hat{\boldsymbol{G}}^T\boldsymbol{V}\hat{\boldsymbol{G}} + \boldsymbol{W}\right)^{-1}\hat{\boldsymbol{G}}^T\boldsymbol{V} \tag{3.14}$$

The optimal feedforward update is thus equal to a regularised pseudo-inverse of $\hat{\boldsymbol{G}}$ times the tracking error in the previous iteration.

The explicit expression for the feedforward update in equation (3.13) is used for analysis of the robustness and convergence properties of NILC in section 3.4. However, the expression is not suited for practical implementation of NILC for long iterations, because the number of elements of the lifted matrices in this equation is proportional to the square of the length of the iteration and thus the computation is time-consuming for long iterations.

3.3.2 Solution using optimal control theory

In this section the equations that define the optimal feedforward update for NILC are rewritten to a finite-horizon optimal control problem. This control problem is solved using an existing optimal control algorithm, yielding a computationally efficient algorithm to compute the optimal feedforward update for NILC.

To formulate the optimal control problem, the lifted expressions for the optimal feedforward update in equation (3.9) and the objective function in equation (3.10) are expressed in terms of the time-samples of the various signals as

$$\breve{u}_i^{k+1} = \arg\min_{u_i^{k+1}} {J^{(n)}}^{k+1}, \tag{3.15}$$

$${J^{(n)}}^{k+1} \triangleq \sum_{i=1}^{N_i-1} \left(\hat{e}_{i+1}^{k+1T} V_{i+1} \hat{e}_{i+1}^{k+1} + u_i^{k+1T} W_i u_i^{k+1} \right). \tag{3.16}$$

The first time sample of the error estimate ($\hat{e}_1^{k+1}$) is removed from the objective function since it cannot be changed by any feedforward update. Similarly, the last time sample of the feedforward update ($u_{N_i}^{k+1}$) is removed from the objective function, because it does not change the error in the considered time-interval and thus its optimal value is zero. The error estimate is obtained from the state-space equations associated with the lifted expression (3.7). These state-space equations are

$$\hat{x}_{i+1}^{k+1} = \hat{A}_i \hat{x}_i^{k+1} + \hat{B}_i u_i^{k+1}, \tag{3.17a}$$

$$\hat{x}_1^{k+1} = O, \tag{3.17b}$$

$$\hat{e}_i^{k+1} = \hat{C}_i \hat{x}_i^{k+1} + e_i^k, \tag{3.17c}$$

where $\{\hat{A}_i, \hat{B}_i, \hat{C}_i\}$ and $\hat{x}_i^{k+1}$ are the state-space matrices and the state-vector associated with the system estimate $\hat{\boldsymbol{G}}$.

Equations (3.15), (3.16) and (3.17) define a finite-horizon optimal control problem. Başar and Olsder (1995) describe an algorithm to compute the optimal feedforward update for a finite-horizon optimal control problem with a quadratic objective function that only depends the system's input and state vector. Such

optimal-control problem is obtained by extending the state-vector with the error and expressing the objective function in terms of the state and the input of the extended state equation, yielding

$$\tilde{x}_{i+1} = \tilde{A}_i \tilde{x}_i + \tilde{B}_i \tilde{u}_i + \tilde{v}_i, \tag{3.18a}$$

$$\breve{\tilde{u}}_i = \arg\min_{\tilde{u}_i} \tilde{J^{(n)}}, \tag{3.18b}$$

$$\tilde{J^{(n)}} = \sum_{i=1}^{N_i-1} \left({\tilde{x}_{i+1}}^T \tilde{Q}_{i+1} \tilde{x}_{i+1} + {\tilde{u}_i}^T \tilde{R}_i \tilde{u}_i \right), \tag{3.18c}$$

where

$$\tilde{x}_i = \begin{bmatrix} \hat{x}_i^{k+1} \\ e_i^k \end{bmatrix}, \tag{3.19a}$$

$$\tilde{x}_1 = \begin{bmatrix} O \\ e_1^k \end{bmatrix}, \tag{3.19b}$$

$$\tilde{u}_i = u_i^{k+1}, \tag{3.19c}$$

$$\tilde{v}_i = \begin{bmatrix} O \\ e_{i+1}^k \end{bmatrix}, \tag{3.19d}$$

$$\tilde{A}_i = \begin{bmatrix} \hat{A}_i & O \\ O & O \end{bmatrix}, \tag{3.19e}$$

$$\tilde{B}_i = \begin{bmatrix} \hat{B}_i \\ O \end{bmatrix}, \tag{3.19f}$$

$$\tilde{C}_i^{(e)} = \begin{bmatrix} \hat{C}_i & I \end{bmatrix}, \tag{3.19g}$$

$$\tilde{Q}_i = \tilde{C}_i^{(e)T} V_i \tilde{C}_i^{(e)}, \tag{3.19h}$$

$$\tilde{R}_i = W_i. \tag{3.19i}$$

Equations (3.18) define the affine quadratic discrete-time optimal control problem as analysed by Başar and Olsder (1995). The procedure to compute the solution to the optimal control problem, yielding the optimal value of $\breve{\tilde{u}}_i = u_i^{k+1}$, is given at the end of appendix B.1. The procedure consists of the solution of a non-stationary Riccati difference equation, the solution of an anti-causal state-convolution and a causal state-convolution. The number of computational operations of the procedure scales linearly with the length of the iteration. The second step of the procedure checks if the optimal value of the input indeed minimises the objective function. This condition is equivalent to condition (3.12) for the lifted solution. The condition is satisfied if matrices $\tilde{R}_i$ and $\tilde{Q}_i$ are positive definite, which is equivalent to the requirement that matrices V_i and W_i should be positive definite.

The number of computational operations to compute the optimal feedforward update using the algorithm described in this section, which is based on

the optimal-control solution from appendix B.1, depends on the state-dimension and scales linearly with the number of time steps N_i. The number of computational operations to compute the optimal feedforward update using the algorithm from subsection 3.3.1 does not depend on the state-dimension but scales at least quadratically with N_i as it involves the computation of the inverse of a lifted matrix. The procedure described in this section is thus more efficient than the algorithm from subsection 3.3.1 for systems with a low state dimension and a large number of time steps. Moreover, the algorithm is suited for LTV systems and uses no measurements of the error from previous iterations. The algorithm thus meets the requirements following from the objective of this thesis (see section 1.2) and therefore it is used to compute the optimal feedforward update for the experiments of which the results are described in chapter 6.

3.4 Convergence Analysis

In this section the convergence properties of the proposed NILC algorithm are analysed. In particular, the effect of the weighting matrices on the convergence properties is investigated. A similar analysis can be found in most literature on NILC. Still, the convergence analysis is included in this work for two reasons. Firstly, the convergence analysis serves as a basis for the formulation of guidelines for tuning the weighting matrices of NILC. Secondly, the analysis is used for comparison of the convergence properties of NILC with the convergence properties of RILC, which are analysed in section 4.4.

In subsection 3.4.1 the selection of the weighting matrices is discussed. Thereafter, in subsection 3.4.2, the conditions for convergence of the feedforward and the error are derived. Moreover, an expression for the final error is given. In subsection 3.4.3 the convergence analysis is elaborated for systems with a particular type of model error for which the convergence of components of the feedforward and the error can be decoupled. In subsection 3.4.4 the results from the convergence analysis are used to formulate guidelines for the selection of the weighting matrices and the robustness filter.

3.4.1 Preliminaries

The convergence analysis is simplified by assuming that the weighting matrices are selected as

$$\boldsymbol{V} = \boldsymbol{I}, \qquad \boldsymbol{W} = w^2 \boldsymbol{I}, \tag{3.20}$$

where w is a non-zero real number that determines the relative weight on the feedforward update. Substitution of these weighting matrices in the learning matrix in equation (3.14) gives

$$\boldsymbol{L} = \left(\hat{\boldsymbol{G}}^T \hat{\boldsymbol{G}} + w^2 \boldsymbol{I}\right)^{-1} \hat{\boldsymbol{G}}^T. \tag{3.21}$$

In subsection 3.4.2 the convergence of $\boldsymbol{e}^k$ and $\boldsymbol{f}^k$ will be analysed for system equation (3.3), update equations (3.5) and (3.13) with the learning matrix from equation (3.21).

Hereafter it is shown that the choice of the weighting matrices does not limit the generality of the convergence analysis in subsection 3.4.2. A similar convergence analysis can be used to prove convergence of a transformed feedforward and error vector for any other set of symmetric positive weighting matrices. The transformation is based on the Cholesky decomposition of the weighting matrices, yielding

$$\boldsymbol{V} = \tilde{\boldsymbol{V}}\tilde{\boldsymbol{V}}^T, \qquad \boldsymbol{W} = w^2\tilde{\boldsymbol{W}}\tilde{\boldsymbol{W}}^T, \tag{3.22}$$

where w is some non-zero real number. Using this decomposition the following transformations are introduced

$$\tilde{\boldsymbol{e}}^k = \tilde{\boldsymbol{V}}^T\boldsymbol{e}^k, \qquad \tilde{\boldsymbol{d}} = \tilde{\boldsymbol{V}}^T\boldsymbol{d}, \qquad \tilde{\boldsymbol{u}}^k = \tilde{\boldsymbol{W}}^T\boldsymbol{u}^k, \qquad \tilde{\boldsymbol{f}}^k = \tilde{\boldsymbol{W}}^T\boldsymbol{f}^k, \tag{3.23}$$

The system equation (3.3), the feedforward update equation (3.5) and the update equation (3.13) are formulated in terms of these transformed vectors as

$$\tilde{\boldsymbol{e}}^{k+1} = \tilde{\boldsymbol{G}}\tilde{\boldsymbol{f}}^{k+1} + \tilde{\boldsymbol{d}}, \tag{3.24}$$

$$\tilde{\boldsymbol{f}}^{k+1} = \tilde{\boldsymbol{f}}^k + \tilde{\boldsymbol{u}}^{k+1} \tag{3.25}$$

$$\tilde{\boldsymbol{u}}^{k+1} = -\tilde{\boldsymbol{L}}\tilde{\boldsymbol{e}}^k, \tag{3.26}$$

where

$$\tilde{\boldsymbol{G}} = \hat{\tilde{\boldsymbol{G}}} + \tilde{\boldsymbol{\Theta}}, \tag{3.27}$$

$$\hat{\tilde{\boldsymbol{G}}} = \tilde{\boldsymbol{V}}^T\hat{\boldsymbol{G}}\tilde{\boldsymbol{W}}^{-T}, \tag{3.28}$$

$$\tilde{\boldsymbol{\Theta}} = \tilde{\boldsymbol{V}}^T\boldsymbol{\Theta}\tilde{\boldsymbol{W}}^{-T}, \tag{3.29}$$

$$\tilde{\boldsymbol{L}} = \left(\hat{\tilde{\boldsymbol{G}}}^T\hat{\tilde{\boldsymbol{G}}} + w^2\boldsymbol{I}\right)^{-1}\hat{\tilde{\boldsymbol{G}}}^T. \tag{3.30}$$

Note that these equations are similar to equations (3.3), (3.5), (3.13) and (3.21) that are used for the analysis of the convergence of $\boldsymbol{e}^k$ and $\boldsymbol{f}^k$ in subsection 3.4.2. The analysis from subsection 3.4.2 can thus be used to prove convergence of $\tilde{\boldsymbol{e}}^k$ and $\tilde{\boldsymbol{f}}^k$ using the transformed set of equations. The norm of $\tilde{\boldsymbol{e}}^k$ and $\tilde{\boldsymbol{f}}^k$ is related to $\boldsymbol{e}^k$ and $\boldsymbol{f}^k$ as

$$\left\|\tilde{\boldsymbol{e}}^k\right\|_2^2 = \tilde{\boldsymbol{e}}^{kT}\tilde{\boldsymbol{e}}^k = \boldsymbol{e}^{kT}\tilde{\boldsymbol{V}}\tilde{\boldsymbol{V}}^T\boldsymbol{e}^k = \boldsymbol{e}^{kT}\boldsymbol{V}\boldsymbol{e}^k, \tag{3.31}$$

$$w^2\left\|\tilde{\boldsymbol{f}}^k\right\|_2^2 = w^2\tilde{\boldsymbol{f}}^{kT}\tilde{\boldsymbol{f}}^k = w^2\boldsymbol{f}^{kT}\tilde{\boldsymbol{W}}\tilde{\boldsymbol{W}}^T\boldsymbol{f}^k = \boldsymbol{f}^{kT}\boldsymbol{W}\boldsymbol{f}^k. \tag{3.32}$$

Thus if monotonic convergence of the transformed vectors $\tilde{\boldsymbol{e}}^k$ and $\tilde{\boldsymbol{f}}^k$ is proved for the transformed set of system equations using the convergence analysis from subsection 3.4.2, then this implies monotonic convergence of the weighted norm of the error and the feedforward update respectively.

3.4.2 Convergence analysis

Convergence of the feedforward

The relation between the feedforwards in iterations k and $k+1$ is obtained by substitution of the optimal feedforward update equation (3.13) and system equation (3.3) in update equation (3.5):

$$\boldsymbol{f}^{k+1} = \boldsymbol{f}^k - \boldsymbol{L}\boldsymbol{e}^k = \boldsymbol{f}^k - \boldsymbol{L}\left(\boldsymbol{G}\boldsymbol{f}^k + \boldsymbol{d}\right) = \left(\boldsymbol{I} - \boldsymbol{L}\boldsymbol{G}\right)\boldsymbol{f}^k - \boldsymbol{L}\boldsymbol{d}. \tag{3.33}$$

From this equation and the definition of the spectral norm it follows that the difference between feedforward and its final value converges monotonically to zero if the following condition is satisfied

$$\left\|\boldsymbol{I} - \boldsymbol{L}\boldsymbol{G}\right\|_{i2} < 1. \tag{3.34}$$

Note that this condition can only be satisfied if $\boldsymbol{L}\boldsymbol{G}$ is non-singular. Using the expression for the learning matrix in equation (3.21) and the definition of the model error matrix in equation (3.8), this inequality can be written as

$$\left\|\left(w^2\boldsymbol{I} + \hat{\boldsymbol{G}}^T\hat{\boldsymbol{G}}\right)^{-1}\left(w^2\boldsymbol{I} - \hat{\boldsymbol{G}}^T\boldsymbol{\Theta}\right)\right\|_{i2} < 1. \tag{3.35}$$

In subsection 3.4.3 it is shown that this condition is satisfied if the system is modelled perfectly, i.e., $\boldsymbol{\Theta} = \boldsymbol{O}$. Otherwise, this inequality gives a bound on the model error for which the feedforward converges monotonically.

Convergence condition (3.35) is not satisfied if the model error is too large. In that case the feedforward does not converge monotonically if update equation (3.5) is applied. A common method to overcome this problem is the application of a so-called robustness filter (see subsection 2.1.4). A robustness filter removes the components of the feedforward to which the response of the system is not known. With a robustness filter, the feedforward update equation becomes

$$\boldsymbol{f}^{k+1} = \boldsymbol{Q}\left(\boldsymbol{f}^k + \boldsymbol{u}^{k+1}\right), \tag{3.36}$$

where $\boldsymbol{Q}$ is the lifted representation of the robustness filter. The effect of the robustness filter on the convergence of the feedforward can be analysed by inserting system equation (3.3) and the feedforward update equation (3.13) in update equation (3.36), yielding

$$\boldsymbol{f}^{k+1} = \boldsymbol{Q}\left(\boldsymbol{I} - \boldsymbol{L}\boldsymbol{G}\right)\boldsymbol{f}^k - \boldsymbol{Q}\boldsymbol{L}\boldsymbol{d}. \tag{3.37}$$

From this equation follows that the feedforward converges monotonically if the following condition is satisfied

$$\left\|\boldsymbol{Q}\left(\boldsymbol{I} - \boldsymbol{L}\boldsymbol{G}\right)\right\|_{i2} < 1. \tag{3.38}$$

Using the expression for the learning matrix in equation (3.14) and the definition of the model error matrix in equation (3.8), this inequality can be written as

$$\left\| \boldsymbol{Q} \left(\hat{\boldsymbol{G}}^T \hat{\boldsymbol{G}} + w^2 \boldsymbol{I} \right)^{-1} \left(w^2 \boldsymbol{I} - \hat{\boldsymbol{G}}^T \boldsymbol{\Theta} \right) \right\|_{i2} < 1. \tag{3.39}$$

This condition can be satisfied by an appropriate selection of the robustness filter $\boldsymbol{Q}$. The convergence condition is satisfied for any $\boldsymbol{\Theta}$ by taking $\boldsymbol{Q} = \boldsymbol{O}$, but in this trivial case the feedforward is not updated (see equation (3.36)).

Convergence of the error

Firstly, the convergence of the error is analysed for the case no robustness filter is applied, then the effect of a robustness filter on the convergence of the error is investigated. The relation between the errors in iterations k and $k+1$ is obtained by combining system equation (3.3), the feedforward update equation (3.5) (no robustness filter) and the optimal feedforward update equation (3.13) as

$$\boldsymbol{e}^{k+1} \;=\; \boldsymbol{e}^k \,+\, \boldsymbol{G} \left(\boldsymbol{f}^{k+1} - \boldsymbol{f}^k \right) \;=\; \boldsymbol{e}^k \,+\, \boldsymbol{G}\boldsymbol{u}^{k+1} \;=\; (\boldsymbol{I} - \boldsymbol{G}\boldsymbol{L})\, \boldsymbol{e}^k \tag{3.40}$$

The fraction between the 2-norm of the error in iteration $k+1$ and k is thus $\|\boldsymbol{I} - \boldsymbol{G}\boldsymbol{L}\|_{i2}$ at maximum, giving an upper bound for the convergence ratio. The error $\boldsymbol{e}^k$ converges monotonically *to zero* if

$$\|\boldsymbol{I} - \boldsymbol{G}\boldsymbol{L}\|_{i2} < 1. \tag{3.41}$$

Note that this condition can only be satisfied if $\boldsymbol{G}\boldsymbol{L}$ is non-singular. The conditions for monotonic convergence of the feedforward (equation (3.34)) and the error (equation (3.41)) can thus only be satisfied both if $\boldsymbol{G}$ is nonsingular. Using the definition of the model error matrix in equation (3.8) the convergence condition can be written as

$$\left\| \boldsymbol{I} - \hat{\boldsymbol{G}}\boldsymbol{L} - \boldsymbol{\Theta}\boldsymbol{L} \right\|_{i2} < 1. \tag{3.42}$$

This inequality gives a bound on the modelling error for which monotonic convergence of the error is guaranteed. If the model error is too large, then convergence condition (3.42) is not satisfied and the error does not converge monotonically. As mentioned previously, monotonic convergence of the feedforward can be realised for any model error by the application of a robustness filter, which should be chosen such that condition (3.39) is satisfied. Monotonic convergence of the feedforward implies convergence of the error, though monotonic convergence of the error is not necessarily guaranteed. Nonetheless, an expression for the final error can be derived. From equation (3.37) the following expression for the feedforward after convergence is derived,

$$\lim_{k \to \infty} \boldsymbol{f}^k = \left(\boldsymbol{Q} - \boldsymbol{I} - \boldsymbol{Q}\boldsymbol{L}\boldsymbol{G} \right)^{-1} \boldsymbol{Q}\boldsymbol{L}\boldsymbol{d}. \tag{3.43}$$

Substitution of this final feedforward in system equation (3.3) yields the following expression for the final error

$$\lim_{k\to\infty} \boldsymbol{e}^k = \left(\boldsymbol{I} + \boldsymbol{G}\left(\boldsymbol{Q} - \boldsymbol{I} - \boldsymbol{QLG}\right)^{-1}\boldsymbol{QL}\right)\boldsymbol{d}. \tag{3.44}$$

This final error can be non-zero. Trivially, $\boldsymbol{Q} = \boldsymbol{O}$ results in $\boldsymbol{e}^k = \boldsymbol{d}$, i.e., the error is not reduced if no learning is applied. Previously it was shown that if the error converges and $\boldsymbol{Q} = \boldsymbol{I}$, then the error converges to zero. Summarising, it can be stated that a robustness filter can be used to increase the robustness to model errors, but it may also result in a non-zero final error.

3.4.3 Decoupled convergence analysis

In this subsection the convergence analysis for NILC is elaborated for square systems with a special type of model error. The only model error is assumed to be the size of the singular values of the estimated lifted system matrix. The convergence of components of the error and the feedforward can be decoupled for this case and the error and the feedforward converge if each of their components converge, which facilitates the convergence analysis. The results from the decoupled analysis are valid if the system is square and modelled perfectly, i.e., if $\boldsymbol{\Theta} = \boldsymbol{O}$. Moreover, the decoupled analysis is illustrative for the effect of other types of uncertainty like errors in the modelled frequency response of an LTI system, because a close relation exists between the singular values of the lifted system matrix of an LTI system and the frequency response of that system (Dijkstra, 2004).

Preliminaries

The decoupled convergence analysis is based on the singular value decomposition of the estimated lifted system matrix, which is denoted as

$$\hat{\boldsymbol{G}} = \boldsymbol{USV}^T, \tag{3.45}$$

where $\boldsymbol{S}$ is a diagonal matrix containing the singular values s_i on its diagonal and $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices containing the singular vectors in their columns. Dijkstra (2004) has shown that the singular value decomposition of the lifted matrix of an LTI system for an infinitely long iterations results in singular vectors containing frequency components for which the gain of the system's frequency response is identical and the singular values are equal to the corresponding gain.

Based on the singular value decomposition the following transformations are introduced

$$\bar{\boldsymbol{e}}^k = \boldsymbol{U}^T\boldsymbol{e}^k, \qquad \bar{\boldsymbol{d}} = \boldsymbol{U}^T\boldsymbol{d}, \qquad \bar{\boldsymbol{u}}^k = \boldsymbol{V}^T\boldsymbol{u}^k, \qquad \bar{\boldsymbol{f}}^k = \boldsymbol{V}^T\boldsymbol{f}^k, \tag{3.46}$$

The system equation (3.3), the feedforward update equation (3.5) and the update equation (3.13) can be reformulated in terms of these transformed vectors as

$$\bar{\boldsymbol{e}}^{k+1} = \bar{\boldsymbol{G}}\bar{\boldsymbol{f}}^{k+1} + \bar{\boldsymbol{d}}, \tag{3.47}$$

$$\bar{\boldsymbol{f}}^{k+1} = \bar{\boldsymbol{f}}^{k} + \bar{\boldsymbol{u}}^{k+1} \tag{3.48}$$

$$\bar{\boldsymbol{u}}^{k+1} = -\bar{\boldsymbol{L}}\bar{\boldsymbol{e}}^{k}, \tag{3.49}$$

where, using the weighting matrices from equation (3.20),

$$\bar{\boldsymbol{G}} = \boldsymbol{S} + \bar{\boldsymbol{\Theta}}, \tag{3.50}$$

$$\bar{\boldsymbol{\Theta}} = \boldsymbol{U}^T\boldsymbol{\Theta}\boldsymbol{T}, \tag{3.51}$$

$$\bar{\boldsymbol{L}} = \boldsymbol{T}^T\boldsymbol{L}\boldsymbol{U} = \left(\boldsymbol{S}^T\boldsymbol{S} + w^2\boldsymbol{I}\right)^{-1}\boldsymbol{S}^T. \tag{3.52}$$

In the rest of this subsection, it is assumed that $\hat{\boldsymbol{G}}$ is square and $\bar{\boldsymbol{\Theta}}$ is a diagonal matrix. A square $\hat{\boldsymbol{G}}$ means that the number of inputs and outputs is equal. A diagonal $\bar{\boldsymbol{\Theta}}$ implies that the only model error is the size of the singular values. The response of the model and the real system to a column of $\boldsymbol{T}$ is proportional to the corresponding column of $\boldsymbol{U}$. The gain of the model is s_i and the gain of the real system is $s_i + \theta_i$, where θ_i are the diagonal components of $\bar{\boldsymbol{\Theta}}$. Note that $\bar{\boldsymbol{\Theta}} = \boldsymbol{O}$ gives $\theta_i = 0$ and thus the following convergence analysis is valid if there is no model error.

The convergence of the components of the transformed vectors of the error $\bar{\boldsymbol{e}}^k$ and the feedforward update $\bar{\boldsymbol{u}}^k$ can be decoupled if $\hat{\boldsymbol{G}}$ is square and $\bar{\boldsymbol{\Theta}}$ is a diagonal matrix. Recalling that the diagonal components of $\boldsymbol{S}$ and $\bar{\boldsymbol{\Theta}}$ are denoted as s_i and θ_i respectively, the system equation (3.3), the feedforward update equation (3.48) and the update equation (3.49) can be decoupled as

$$\bar{e}_i^{k+1} = (s_i + \theta_i)\,\bar{f}_i^{k+1} + \bar{d}_i, \tag{3.53}$$

$$\bar{f}_i^{k+1} = \bar{f}_i^{k} + \bar{u}_i^{k+1} \tag{3.54}$$

$$\bar{u}_i^{k+1} = -l_i\bar{e}_i^{k}, \tag{3.55}$$

where the subscript i denotes the i^{th} component of the lifted vector and

$$l_i = \frac{s_i}{w^2 + s_i^2}. \tag{3.56}$$

Hereafter, the convergence of $\bar{e}_i^{k+1}$ and $\bar{f}_i^{k+1}$ is analysed. The norm of $\boldsymbol{e}^k$ and $\boldsymbol{f}^k$ can be expressed in terms of $\bar{e}_i^{k+1}$ and $\bar{f}_i^{k+1}$ as

$$\sum_i \left(\bar{e}_i^{k+1}\right)^2 = \left\|\bar{\boldsymbol{e}}^k\right\|_2^2 = \boldsymbol{e}^{kT}\boldsymbol{U}\boldsymbol{U}^T\boldsymbol{e}^k = \boldsymbol{e}^{kT}\boldsymbol{e}^k = \left\|\boldsymbol{e}^k\right\|_2^2, \tag{3.57}$$

$$\sum_i \left(\bar{f}_i^{k+1}\right)^2 = \left\|\bar{\boldsymbol{u}}^k\right\|_2^2 = \boldsymbol{u}^{kT}\boldsymbol{T}\boldsymbol{T}^T\boldsymbol{u}^k = \boldsymbol{u}^{kT}\boldsymbol{u}^k = \left\|\boldsymbol{u}^k\right\|_2^2. \tag{3.58}$$

Thus if convergence of $\bar{e}_i^{k+1}$ and $\bar{f}_i^{k+1}$ is proved for all i then this implies monotonic convergence of the weighted norm of the error and feedforward update respectively.

Convergence of the feedforward

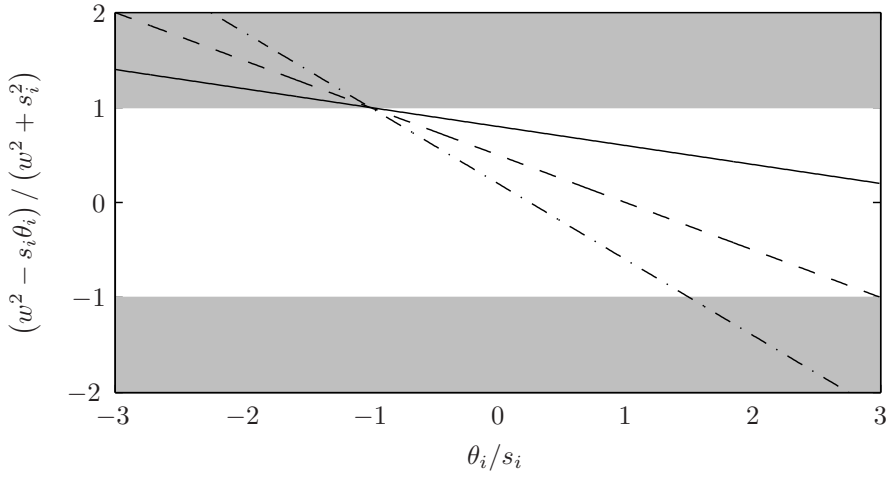
The relation between the components of the transformed feedforwards in iteration k and $k+1$ is obtained by substitution of the optimal feedforward update equation (3.55), system equation (3.53) and the learning matrix from equation (3.56) in update equation (3.54), yielding

$$\bar{f}_i^{k+1} = \bar{f}_i^k - l_i \bar{e}_i^k = \bar{f}_i^k - l_i \left((s_i + \theta_i) \bar{f}_i^k + \bar{d}_i \right) = \frac{w^2 - s_i \theta_i}{w^2 + s_i^2} \bar{f}_i^k - l_i \bar{d}_i. \quad (3.59)$$

The component $\bar{f}_i^k$ thus converges if

$$\left| \frac{w^2 - s_i \theta_i}{w^2 + s_i^2} \right| < 1. \quad (3.60)$$

This convergence condition gives a bound on the allowable model error in the singular value. Figure 3.1 shows the convergence ratio of the feedforward as



— $s_i/w = 0.5$, – – $s_i/w = 1.0$, ·–· $s_i/w = 2.0$,
the feedforward component diverges in the shaded areas

Figure 3.1: The convergence ratio of the feedforward component $\bar{f}_i^k$ as a function of the relative error in the singular value (θ_i/s_i) for NILC without a robustness filter

a function of relative error in the modelled singular value (θ_i/s_i). The convergence condition is satisfied for any real w if $|\theta_i/s_i| < 1$. As a special case, the convergence condition is satisfied if $\theta_i = 0$ and $s_i \neq 0$. This means that the feedforward converges if the controlled system is non-singular and modelled perfectly. The convergence condition is satisfied for $|\theta_i/s_i| > 1$ only if θ_i/s_i is positive and s_i/w is sufficiently small. The weight w can thus be used to increase the allowable model error. The convergence condition is not satisfied for any real w if $\theta_i/s_i < -1$, i.e., the signs of the singular values of the real system and the model are different. In that case a robustness filter could be applied to realise monotonic convergence of the feedforward. The convergence of the components of the feedforward can still be decoupled if the robustness filter is chosen as

$$\boldsymbol{Q} = \boldsymbol{T}^T \bar{\boldsymbol{Q}} \boldsymbol{T}, \tag{3.61}$$

where $\bar{\boldsymbol{Q}}$ is a diagonal matrix with the components q_i on its diagonal. The relation between the components of the transformed feedforwards in iteration k and $k+1$ is obtained by combining the expression for the robustness filter, update equation (3.36), the transformations from equations (3.46), the feedforward update equation (3.55) and the learning matrix from equation (3.56), yielding

$$\begin{aligned} \bar{f}_i^{k+1} &= q_i \left(\bar{f}_i^{k+1} + \bar{u}_i^{k+1} \right) = q_i \left(\bar{f}_i^{k+1} - l_i \bar{e}_i^k \right) = \\ & q_i \left(\bar{f}_i^{k+1} - l_i \left((s_i + \theta_i) \, \bar{f}_i^{k+1} + \bar{d}_i \right) \right) = q_i \frac{w^2 - s_i \theta_i}{w^2 + s_i^2} \bar{f}_i^k - q_i \frac{s_i}{w^2 + s_i^2} \bar{d}_i. \end{aligned} \tag{3.62}$$

The components $\bar{f}_i^k$ thus converge if

$$\left| q_i \frac{w^2 - s_i \theta_i}{w^2 + s_i^2} \right| < 1 \tag{3.63}$$

This condition can be satisfied for any θ_i by choosing a sufficiently small value for q_i. Figure 3.2 shows the left hand side of the condition for different values of q_i. Clearly, a small q_i results in a large range of θ_i/s_i for which the feedforward converges. Thus convergence of the feedforward component corresponding to a large model error can be realised by taking the corresponding gain of the robustness filter small.

Convergence of the error

Firstly, the convergence of the error is analysed for the case no robustness filter is applied, then the effect of a robustness filter on the convergence of the error is investigated. Combining system equation (3.53), the feedforward update equation (3.54), the optimal feedforward update equation (3.55) and the learning

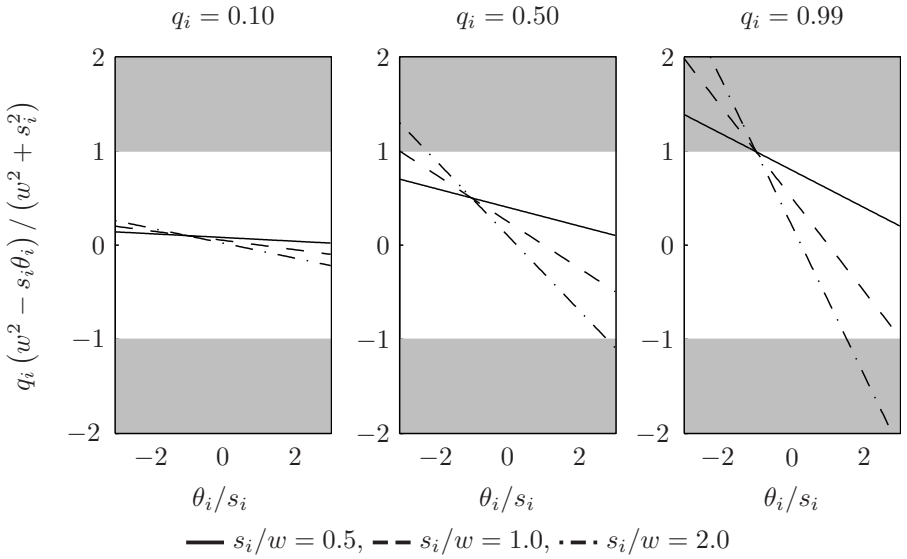


Figure 3.2: The convergence ratio of the feedforward component $\bar{f}_i^k$ as a function of the relative error in the singular value (θ_i/s_i) for NILC with a robustness filter

matrix from equation (3.56) gives

$$\begin{aligned}\bar{e}_i^{k+1} &= \bar{e}_i^k + (s_i + \theta_i)\left(\bar{f}_i^{k+1} - \bar{f}_i^k\right) = \bar{e}_i^k + (s_i + \theta_i)\,\bar{u}_i^{k+1} \\ &= \bar{e}_i^k - (s_i + \theta_i)\,l_i\bar{e}_i^k = \left(1 - (s_i + \theta_i)\,\frac{s_i}{w^2 + s_i^2}\right)\bar{e}_i^k = \frac{w^2 - s_i\theta_i}{w^2 + s_i^2}\bar{e}_i^k \end{aligned} \quad (3.64)$$

The convergence ratio of the error component $\bar{e}_i^k$ is thus $\left(w^2 - s_i\theta_i\right) / \left(w^2 + s_i^2\right)$. and the error component converges to zero if

$$\left|\frac{w^2 - s_i\theta_i}{w^2 + s_i^2}\right| < 1. \quad (3.65)$$

Note that this condition for the convergence of the error component $\bar{e}_i^k$ is equal to the condition for the convergence of the feedforward component $\bar{f}_i^k$ in equation (3.60). Again the condition is satisfied for any real w if $|\theta_i/s_i| < 1$, with $\theta_i = 0$ as a special case. This means that if the system is non-singular and modelled perfectly, then the error converges *to zero*. Figure 3.3 shows the convergence ratio of the error component $\bar{e}_i^k$ as a function of s_i/w for $\theta_i = 0$. The larger the value of s_i/w, the smaller the convergence ratio. The error converges even to zero in one trial if $w = 0$. The error is not reduced if $s_i = 0$, which

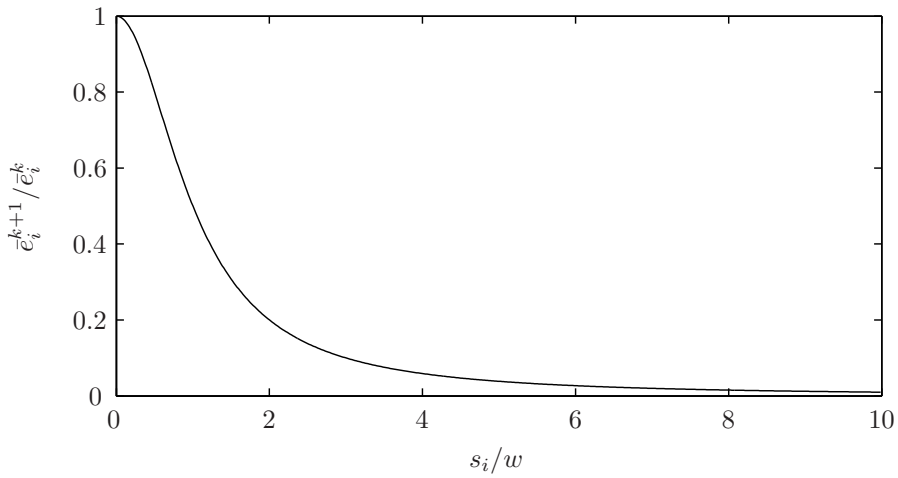


Figure 3.3: The convergence ratio of the error component $\bar{e}_i^k$ as a function of the fraction of the singular value and input-weight (s_i/w) for NILC without a model error ($\theta_i = 0$)

implies that the error components corresponding to the system's zeroes cannot be compensated. The convergence ratio of the error for $\theta_i \neq 0$ is depicted in figure 3.1. Clearly, the error converges if $|\theta_i/s_i| < 1$ and a larger model error is only allowable if θ_i/s_i is positive.

The convergence condition cannot be satisfied for any real w if $\theta_i/s_i < -1$, i.e., the signs of the singular values of the real system and the model are different. As mentioned previously, monotonic convergence of the feedforward can be realised for any model error by the application of a robustness filter. The feedforward converges if convergence condition (3.63) is satisfied for all i. From equation (3.62) the following expression for the feedforward after convergence is derived:

$$\lim_{k\to\infty} \bar{f}_i^{k+1} = -\frac{q_i s_i}{w^2 + s_i^2 - q_i\left(w^2 - s_i\theta_i\right)}\bar{d}_i. \tag{3.66}$$

Substitution of this equation in system equation (3.53) gives the following expression for the final error

$$\lim_{k\to\infty} \bar{e}_i^k = \frac{w^2 + s_i^2 - q_i\left(w^2 + s_i^2\right)}{w^2 + s_i^2 - q_i\left(w^2 - s_i\theta_i\right)}\bar{d}_i \tag{3.67}$$

This expression shows that if the error component $\bar{e}_i^k$ converges, then it indeed converges to zero if $q_i = 1$. Figure 3.4 shows the final error component for several other values of q_i. The final error component is only plotted for the range of θ_i/s_i where the convergence condition (3.63) is satisfied. The figure shows that

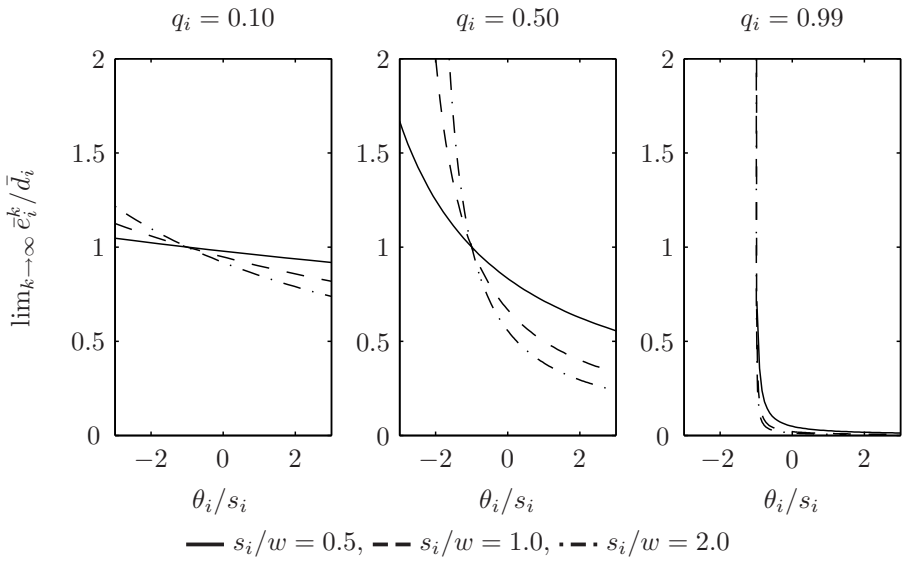


Figure 3.4: The reduction of the error as a function of the relative error in the singular value (θ_i/s_i) for NILC with a robustness filter

for a small value of q_i a large final error remains. Trivially, $q_i = 0$ results in $\bar{e}_i^k = \bar{d}_i$, i.e., the error component is not reduced if no learning is applied. For any value of q_i, the final error component $\bar{e}_i^\infty$ is larger than $\bar{d}_i$ if $\theta_i/s_i < -1$, i.e., the error component increases when the signs of the singular values of the system and the model differ.

3.4.4 Parameter selection

In the preceding subsections the convergence properties of NILC are analysed. In particular, the effects of the weight w and the robustness filter on the convergence of the feedforward and the final error are analysed. In this subsection several guidelines for the selection of these parameters are formulated based on the results of the convergence analysis. The guidelines are partly based on the analysis subsection 3.4.3, in which it is assumed that the system is square and the only model error is the size of the singular values of the system matrix.

Selection of the weight

The learning matrix, expressed by equation (3.14), depends on the selection of the weight w. The convergence ratio of the error, expressed by equation (3.40),

depends on this learning matrix and thus on the selection of w. In subsection 3.4.3 it is shown that a small value of w increases the convergence rate (figure 3.3).

The conditions for the convergence of the feedforward depend on the selected weight according to equations (3.35) and (3.39). Thus, the allowable model error is also affected by the selection of w. In subsection 3.4.3 it is shown by figures 3.1 and 3.2 that a large value of w increases the allowable model error.

The selection of w can thus be used to increase the convergence rate or the allowable model error. From subsection 3.4.3 it is concluded that the value of w value should be taken small to increase the convergence rate, but its value should be taken large to increase the allowable model error.

Selection of the robustness filter

Preferably, no robustness filter should be used, because then the error converges to zero according to equation (3.40) if convergence condition (3.42) is satisfied. In subsection 3.4.3 it is shown that this convergence condition is satisfied if the system is square, nonsingular and modelled perfectly. However, the convergence condition cannot be satisfied for a large model error if no robustness filter is applied. In subsection 3.4.3 it is shown that the condition for convergence of the error cannot be satisfied without a robustness filter if the signs of the singular values of the real system and its estimate differ (equation (3.65)).

The condition for convergence of the feedforward, which is expressed by equation (3.39), can be satisfied for a large model error by application of a (non-unity) robustness filter, though the application of such robustness filter may result in a non-zero final error according to equation (3.44). The convergence condition is even satisfied for any model error if the robustness filter is set to zero, though in this case thee feedforward is not updated and the error is not reduced at all. In subsection 3.4.3 it is shown that the allowable error in the singular value can be increased by taking the corresponding gain of the robustness filter smaller than unity (equation (3.63)), but this also implies that the final value of the corresponding error component is not completely compensated according to equation (3.67).

NILC thus results in zero final error if no robustness filter is used. However, a robustness filter can be needed to increase the robustness to modelling errors, although this could result in a nonzero final error. The robustness filter should thus be zero for the components of the error corresponding to a large model uncertainty to obtain convergence, but close to unity for the other components of the error to reduce these error components to zero. In practice, the high-frequency dynamics of a mechanical system are often not modelled accurately. The robustness filter should then be a low-pass filter which is unity at low-frequencies and close to zero at high frequencies. This behaviour can be realised by implementing the robustness filter as a high-order zero-phase low-pass filter. The high-order results in a small frequency band in which the low-pass filter

rolls off from unity to zero. This behaviour cannot be realised by a stable and causal low-pass filter as the roll-off introduces phase-lag at low-frequencies, which makes the filter unequal to unity at those frequencies.

Discussion

In the previous analysis it is concluded that the final error depends on the selection of the robustness filter and the convergence rate depends on the selection of the weight on the feedforward update. The selection of the weight on the feedforward update and the robustness filter both determine the allowable model error. The best selection of these parameters for a certain application thus depends on the expected model error, the desired convergence rate and the allowable final error. The choice of these filters for the experiments on the Stäubli RX90 robot is discussed in subsection 6.2.1.

A limitation of the proposed NILC algorithm is that the convergence ratio can only be tuned indirectly by the selection of the weights, because the actual convergence rate also depends on the system dynamics. Furthermore, the tuning of the weights and the robustness filter determine the allowable model error, but the actual convergence is verified from convergence conditions like inequality (3.34) that contain the model error, which is mostly unknown. The next chapter describes the design of an RILC algorithm that overcomes the aforementioned disadvantages of NILC. A maximum convergence ratio can be specified explicitly and the learning filter is optimised to realise this convergence ratio for the worst case effect of the specified model uncertainty. Moreover, the actual realisation of convergence with the desired convergence ratio is verified for a specification of the model uncertainty instead of the model error.

Chapter 4

Robust ILC

Robust ILC (RILC) is a model-type ILC method to reduce the tracking error of a system with a specified bounded model uncertainty. The ILC algorithm is designed such that the reduction of the error is optimised for the worst-case effect of the model uncertainty.

Section 4.1 describes the linear time-varying system dynamics and the model uncertainty considered in this chapter. The RILC design objective, which specifies the optimal learning filter, is formulated in section 4.2. Algorithms for the computation of the optimal learning filter and for checking robust convergence of the error are described in section 4.3. A computationally efficient algorithm of RILC for LTV systems is presented in subsection 4.3.2. In section 4.4 the convergence properties of the proposed RILC algorithm are analysed and used to formulate guidelines for the selection of the parameters of RILC.

The formulated design objective for RILC has been proposed in a similar form by Van de Wijdeven and Bosgra (2007a,b). The contributions of this chapter are the proposed computationally efficient RILC algorithm for LTV systems, the integration of a (non-causal) robustness filter in the design of RILC and the developed algorithms for checking convergence of RILC for LTV systems and finite time iterations.

4.1 System description

The dynamics of the controlled system are assumed to be linear, time-varying, iteration-invariant and strictly proper. The effect of the feedforward f_i^k on the error e_i^k can thus be modelled by the state-space equations (3.1) or the corresponding lifted equation (3.3).

It is assumed that the real system dynamics $\boldsymbol{G}$ are not exactly known, though an estimate of the system dynamics and a specification of its uncertainty are available. The estimate of the system dynamics is denoted as $\hat{\boldsymbol{G}}$. The uncertainty in the estimate of the system dynamics is specified as additive uncertainty

by the following system equation

$$\boldsymbol{e}^k = \left(\hat{\boldsymbol{G}} + \boldsymbol{N}\boldsymbol{\Delta}\boldsymbol{M}\right)\boldsymbol{f}^k + \boldsymbol{d}, \tag{4.1}$$

where the term $\boldsymbol{N}\boldsymbol{\Delta}\boldsymbol{M}$ represents the additive uncertainty. The lifted matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ represent weighting filters that are selected such that the normalised uncertainty matrix $\boldsymbol{\Delta}$ is bounded as $\|\boldsymbol{\Delta}\|_{i2} < 1$, i.e., the spectral norm of the normalised uncertainty matrix is less than 1. Note that all lifted matrices, including $\boldsymbol{M}$, $\boldsymbol{N}$ and $\boldsymbol{\Delta}$, are assumed to be real-valued in this thesis. The weighting filters $\boldsymbol{M}$ and $\boldsymbol{N}$ are referred to as the uncertainty weighting filters, where $\boldsymbol{M}$ is the pre-weighting filter and $\boldsymbol{N}$ is the post-weighting filter. The selection of these uncertainty weighting filters is not unique as any combination is appropriate as long as the additive uncertainty of the system dynamics is specified as $\boldsymbol{G} = \hat{\boldsymbol{G}} + \boldsymbol{N}\boldsymbol{\Delta}\boldsymbol{M}$ for some $\|\boldsymbol{\Delta}\|_{i2} < 1$. This is illustrated by the following example. Suppose that filters $\boldsymbol{M}$ and $\boldsymbol{N}$ appropriately specify the uncertainty and consider the following modified set of uncertainty weighting filters

$$\tilde{\boldsymbol{N}} = \beta\boldsymbol{N}, \tag{4.2a}$$

$$\tilde{\boldsymbol{M}} = \tfrac{1}{\beta}\boldsymbol{M}, \tag{4.2b}$$

where the weight ratio β is a real scalar. This modified set of weighting matrices specifies the same uncertainty for any value of the weight ratio. The weight ratio thus modifies the size of the uncertainty weighting filters without affecting the overall size of the specified uncertainty. The selection of the weight ratio is discussed in more detail in subsection 4.4.3.

According to the internal model principle (see section 3.1), an ILC algorithm should contain an integrator over the iterations to compensate for the trial invariant disturbance $\boldsymbol{d}$. For the design of RILC, the integrator is implemented by summing the error $\boldsymbol{e}^k$ over the iterations as

$$\boldsymbol{z}^{k+1} = \boldsymbol{z}^k + \boldsymbol{e}^k, \tag{4.3}$$

and relating the feedforward $\boldsymbol{f}^k$ to the *summed error* $\boldsymbol{z}^k$ as

$$\boldsymbol{f}^k = -\boldsymbol{L}\boldsymbol{z}^k, \tag{4.4}$$

where $\boldsymbol{L}$ represents the learning filter. Note that the sequence of the integrator and the learning filter for RILC is the reverse of the sequence for NILC, which is expressed by equations (3.5) and (3.13). The reversed sequence facilitates the design of RILC in the subsequent sections.

4.2 Objective

The goal of RILC is realising robust convergence of the error, i.e., convergence of the error for all systems that comply with the specified bounded model uncertainty. In this work robust convergence of the error $\boldsymbol{e}^k$ is realised by designing

the learning filter $\boldsymbol{L}$ such that the summed error $\boldsymbol{z}^k$ converges robustly and monotonically. Monotonic convergence of the summed error implies that the error converges *to zero* if update equation (4.3) is used. However, it is not possible to guarantee convergence of the summed error for large model uncertainty for this update equation. This is the reason for including a robustness filter in the design of RILC, which is discussed in more detail in subsection 4.2.1. The convergence of the summed error for RILC with a robustness filter is analysed in subsection 4.2.2. This convergence analysis is the basis for the formulation of the design objective for the learning filter $\boldsymbol{L}$ of RILC in subsection 4.2.3. The section is finished with some remarks on the proposed RILC design in subsection 4.2.4.

4.2.1 Robustness filter

As mentioned above, convergence of the error $\boldsymbol{e}^k$ is realised by designing the learning filter $\boldsymbol{L}$ such that the summed error $\boldsymbol{z}^k$ converges robustly and monotonically, which means that the error converges to zero if update equation (4.3) is used. Moreover, it is shown by the convergence analysis in section 4.4 that if update equation (4.3) is used then monotonic convergence of the summed error implies monotonic convergence of the error. However, the convergence analysis also shows that an RILC algorithm based on update equation (4.3) may not result in robust convergence for a large model uncertainty. This conclusion is similar to the conclusion in section 3.4 that NILC without a robustness filter may not result in convergence for a large model error. Robust convergence for a large model uncertainty can be realised by using a robustness filter. The robustness filter is implemented by changing update equation (4.3) to

$$\boldsymbol{z}^{k+1} = \boldsymbol{R}\boldsymbol{z}^k + \boldsymbol{e}^k, \tag{4.5}$$

where $\boldsymbol{R}$ is the robustness filter. The dimension of the summed error $\boldsymbol{z}^k$ is equal to the dimension of the error $\boldsymbol{e}^k$ and thus the dimensions of the robustness filter $\boldsymbol{R}$ are equal to the dimension of the error. Note that the dimensions of robustness filter $\boldsymbol{R}$ differ from the dimensions of the robustness filter $\boldsymbol{Q}$ for NILC, which are equal to the dimension of the feedforward. Only for $\boldsymbol{R} = \boldsymbol{I}$ the variable $\boldsymbol{z}^k$ corresponds to the sum of the error over the iterations. Nevertheless, the variable $\boldsymbol{z}^k$ is referred to as the *summed error* for any other choice of the robustness filter hereafter. The effect of the choice of $\boldsymbol{R}$ on the final error and the convergence properties of RILC is analysed in detail in section 4.4. Guidelines for the selection of the robustness filter are also formulated in section 4.4. In the following subsections the design of the learning matrix $\boldsymbol{L}$ is considered.

4.2.2 Convergence of the summed error

In this subsection the convergence of the summed error is analysed. This convergence analysis forms the basis for the formulation of the design objective for the learning matrix $\boldsymbol{L}$ in subsection 4.2.3.

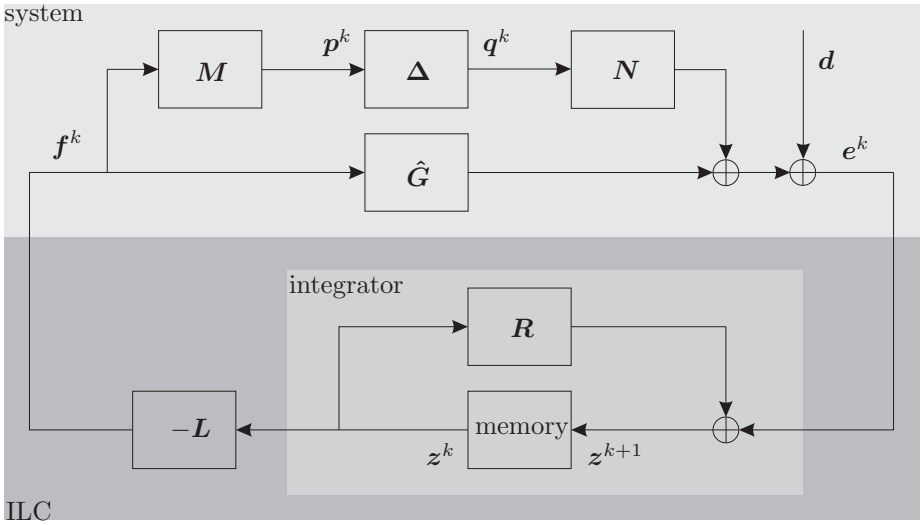


Figure 4.1: Block scheme of a system with additive uncertainty controlled by RILC

The system equation (4.1), the update equation (4.5) and the feedforward equation (4.4) form a closed-loop of which the block diagram is shown in figure 4.1. The relation between the summed error in iterations k and $k+1$ is

$$\boldsymbol{z}^{k+1} = \boldsymbol{R}\boldsymbol{z}^k - \left(\hat{\boldsymbol{G}} + \boldsymbol{N}\boldsymbol{\Delta}\boldsymbol{M}\right)\boldsymbol{L}\boldsymbol{z}^k + \boldsymbol{d} = \left(\boldsymbol{R} - \hat{\boldsymbol{G}}\boldsymbol{L} - \boldsymbol{N}\boldsymbol{\Delta}\boldsymbol{M}\boldsymbol{L}\right)\boldsymbol{z}^k + \boldsymbol{d}. \quad (4.6)$$

The summed error thus converges monotonically if

$$\left\|\boldsymbol{R} - \hat{\boldsymbol{G}}\boldsymbol{L} - \boldsymbol{N}\boldsymbol{\Delta}\boldsymbol{M}\boldsymbol{L}\right\|_{i2} < 1. \quad (4.7)$$

If this condition is satisfied, then the final value of the summed error $\boldsymbol{z}^\infty$ satisfies the following relation

$$\boldsymbol{z}^\infty = \left(\boldsymbol{R} - \hat{\boldsymbol{G}}\boldsymbol{L} - \boldsymbol{N}\boldsymbol{\Delta}\boldsymbol{M}\boldsymbol{L}\right)\boldsymbol{z}^\infty + \boldsymbol{d}. \quad (4.8)$$

Subtracting this equation from equation (4.6) yields

$$\boldsymbol{w}^{k+1} = \left(\boldsymbol{R} - \hat{\boldsymbol{G}}\boldsymbol{L} - \boldsymbol{N}\boldsymbol{\Delta}\boldsymbol{M}\boldsymbol{L}\right)\boldsymbol{w}^k, \quad (4.9)$$

where $\boldsymbol{w}^k$ is defined as

$$\boldsymbol{w}^k = \boldsymbol{z}^k - \boldsymbol{z}^\infty. \quad (4.10)$$

The variable $\boldsymbol{w}^k$ represents the part of the summed error that is reduced to zero over the iterations and therefore this variable is referred to as the *compensable* summed error. Note that the conditions for convergence of the compensable summed error $\boldsymbol{w}^k$ and the summed error $\boldsymbol{z}^k$ coincide.

Further on in this subsection a condition is formulated to satisfy inequality (4.7) for any $\|\boldsymbol{\Delta}\|_{i2} < 1$. The convergence condition is formulated using the standard plant representation of the system in equation (4.9), where the dynamics of the learning matrix $\boldsymbol{L}$, which should be designed, and the normalised uncertainty matrix, which is unknown, are separated from the other dynamics. The standard plant format was previously used for the design of robust ILC by, e.g., Van Dijk et al. (2001); De Roover and Bosgra (2000); Van de Wijdeven and Bosgra (2007a).

For the standard plant representation of the system in equation (4.9) some additional variables are introduced that define the inputs and outputs of the standard plant. The variable $\boldsymbol{u}^k$ is introduced as the output of the learning filter $\boldsymbol{L}$ for input $\boldsymbol{w}^k$, i.e.,

$$\boldsymbol{u}^k = -\boldsymbol{L}\boldsymbol{w}^k. \tag{4.11}$$

Substitution of equation (4.10) and (4.4) in equation (4.11) yields

$$\boldsymbol{u}^k = -\boldsymbol{L}\boldsymbol{z}^k + \boldsymbol{L}\boldsymbol{z}^\infty = \boldsymbol{f}^k - \boldsymbol{f}^\infty. \tag{4.12}$$

Analogous to the term that is used for the variable $\boldsymbol{w}^k$, the variable $\boldsymbol{u}^k$ is referred to as the *compensable input*. Furthermore, the following variables are introduced as the input and output of the normalised uncertainty matrix

$$\boldsymbol{p}^k = \boldsymbol{M}\boldsymbol{u}^k, \tag{4.13a}$$

$$\boldsymbol{q}^k = \boldsymbol{\Delta}\boldsymbol{p}^k. \tag{4.13b}$$

Substitution of the definitions in equations (4.11) and (4.13b) in equation (4.9) gives

$$\boldsymbol{w}^{k+1} = \boldsymbol{R}\boldsymbol{w}^k + \hat{\boldsymbol{G}}\boldsymbol{u}^k + \boldsymbol{N}\boldsymbol{q}^k. \tag{4.14}$$

The standard plant describes the dynamic relation between the compensable summed errors in iteration k and $k+1$, the output and input of the learning filter, which should be designed, and the output and input of the normalised uncertainty matrix, which is unknown. Using equations (4.14) and (4.13a), the standard plant, which is denoted by $\boldsymbol{P}$, can be expressed as

$$\begin{bmatrix} \boldsymbol{w}^{k+1} \\ \boldsymbol{p}^k \\ \boldsymbol{w}^k \end{bmatrix} = \boldsymbol{P} \begin{bmatrix} \boldsymbol{w}^k \\ \boldsymbol{q}^k \\ \boldsymbol{u}^k \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{N} & \hat{\boldsymbol{G}} \\ \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{M} \\ \boldsymbol{I} & \boldsymbol{O} & \boldsymbol{O} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}^k \\ \boldsymbol{q}^k \\ \boldsymbol{u}^k \end{bmatrix}. \tag{4.15}$$

A block-diagram of the dynamics of the system in equation (4.9) in the standard plant format is depicted in figure 4.2, where the shaded block represents the standard plant $\boldsymbol{P}$. The combination of the standard plant $\boldsymbol{P}$ and the learning

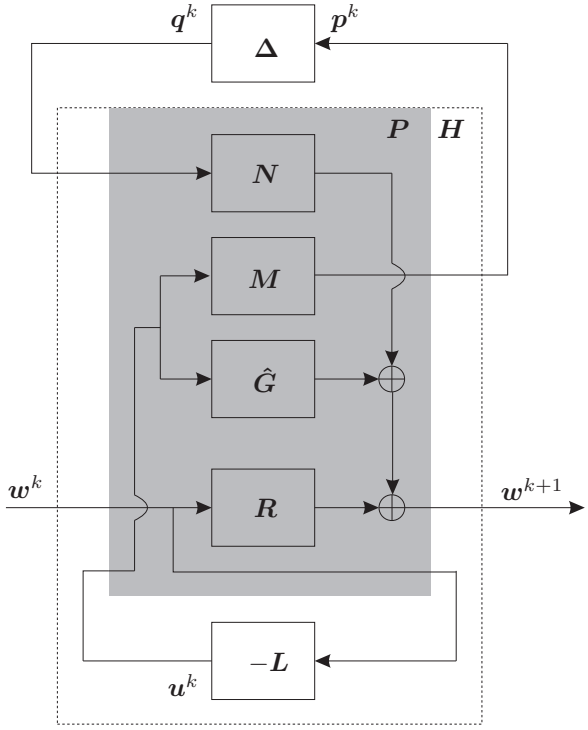


Figure 4.2: Standard plant format of a system with additive uncertainty controlled by RILC

controller $\boldsymbol{L}$, which is outlined by the dashed line and denoted by $\boldsymbol{H}$, can be expressed as

$$\begin{bmatrix} \boldsymbol{w}^{k+1} \\ \boldsymbol{p}^k \end{bmatrix} = \boldsymbol{H} \begin{bmatrix} \boldsymbol{w}^k \\ \boldsymbol{q}^k \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} - \hat{\boldsymbol{G}}\boldsymbol{L} & \boldsymbol{N} \\ -\boldsymbol{M}\boldsymbol{L} & \boldsymbol{O} \end{bmatrix} \begin{bmatrix} \boldsymbol{w}^k \\ \boldsymbol{q}^k \end{bmatrix}. \tag{4.16}$$

Hereafter, it is shown that the convergence of the compensable summed error is related to the spectral norm of matrix $\boldsymbol{H}$, which is denoted as γ_H. This relation is used in subsection 4.2.3 for the formulation of the design objective for $\boldsymbol{L}$. By definition of the spectral norm, the following equality holds

$$\gamma_H = \|\boldsymbol{H}\|_{i2} = \max_{\boldsymbol{w}^k} \max_{\boldsymbol{q}^k} \frac{\left\| \begin{bmatrix} \boldsymbol{w}^{k+1T} & \boldsymbol{p}^{kT} \end{bmatrix}^T \right\|_2}{\left\| \begin{bmatrix} \boldsymbol{w}^{kT} & \boldsymbol{q}^{kT} \end{bmatrix}^T \right\|_2}. \tag{4.17}$$

Squaring both sides of the equation gives

$$\max_{\boldsymbol{w}^k} \max_{\boldsymbol{q}^k} \frac{\left\|\boldsymbol{w}^{k+1}\right\|_2^2 + \left\|\boldsymbol{p}^k\right\|_2^2}{\left\|\boldsymbol{w}^k\right\|_2^2 + \left\|\boldsymbol{q}^k\right\|_2^2} = \gamma_H^2, \tag{4.18}$$

and rewriting this results gives

$$\max_{\boldsymbol{w}^k} \max_{\boldsymbol{q}^k} \left(\left\|\boldsymbol{w}^{k+1}\right\|_2^2 + \left\|\boldsymbol{p}^k\right\|_2^2 - \gamma_H^2 \left\|\boldsymbol{w}^k\right\|_2^2 - \gamma_H^2 \left\|\boldsymbol{q}^k\right\|_2^2 \right) = 0. \tag{4.19}$$

Thus, for any value of $\boldsymbol{w}^k$ and $\boldsymbol{q}^k$ the following inequality holds

$$\left\|\boldsymbol{w}^{k+1}\right\|_2^2 - \gamma_H^2 \left\|\boldsymbol{w}^k\right\|_2^2 + \left\|\boldsymbol{p}^k\right\|_2^2 - \gamma_H^2 \left\|\boldsymbol{q}^k\right\|_2^2 \leq 0. \tag{4.20}$$

Moreover, equation (4.13b) and the assumption $\|\boldsymbol{\Delta}\|_{i2} < 1$ yield the following inequality

$$\left\|\boldsymbol{q}^k\right\|_2 < \left\|\boldsymbol{p}^k\right\|_2. \tag{4.21}$$

Combining inequalities (4.20) and (4.21) for $0 \leq \gamma_H < 1$ gives

$$\left\|\boldsymbol{w}^{k+1}\right\|_2^2 < \gamma_H^2 \left\|\boldsymbol{w}^k\right\|_2^2, \tag{4.22}$$

Thus, $\gamma_H < 1$ is a sufficient condition for monotonic convergence of the compensable summed error $\boldsymbol{w}^k$ with a convergence ratio of at most γ_H for any $\|\boldsymbol{\Delta}\|_{i2} < 1$. Previously it was stated that the conditions for monotonic convergence of the compensable summed error $\boldsymbol{w}^k$ and the summed error $\boldsymbol{z}^k$ coincide. Thus, if the learning matrix $\boldsymbol{L}$ is designed such that $\gamma_H < 1$, then the summed error converges monotonically with a convergence ratio of at most γ_H for any $\|\boldsymbol{\Delta}\|_{i2} < 1$.

4.2.3 The design objective

The convergence analysis from the previous subsection is used to formulate the design objective for RILC that specifies the optimal learning filter. This design objective is formulated using the objective function $J^{(s)^k}$, which is defined as

$$J^{(s)^k} \triangleq \left\|\boldsymbol{w}^{k+1}\right\|_2^2 + \left\|\boldsymbol{p}^k\right\|_2^2 - \gamma^2 \left\|\boldsymbol{w}^k\right\|_2^2 - \gamma^2 \left\|\boldsymbol{q}^k\right\|_2^2, \tag{4.23}$$

where γ is a real scalar satisfying $0 < \gamma < 1$. This objective function is defined such that if

$$\max_{\boldsymbol{w}^k} \max_{\boldsymbol{q}^k} J^{(s)^k} < 0, \tag{4.24}$$

then equality (4.19) holds for some $\gamma_H < \gamma < 1$ and thus $\boldsymbol{w}^k$ converges to zero with a convergence ratio of at most γ according to inequality (4.22). With this in mind, the optimal learning filter is defined as the filter that relates the compensable input $\boldsymbol{u}^k$ to the compensable summed error $\boldsymbol{w}^k$ (equation (4.11))

such that objective function ${J^{(s)}}^k$ is minimised for the worst case effect of the model uncertainty $\boldsymbol{q}^k$. This design objective is formulated in terms of the output of the optimal learning filter as

$$\check{\boldsymbol{u}}^k = \arg\min_{\boldsymbol{u}^k} \max_{\boldsymbol{q}^k} {J^{(s)}}^k, \tag{4.25}$$

where $\boldsymbol{u}^k$ is a function of $\boldsymbol{w}^k$. Robust convergence with a convergence ratio of at most γ is guaranteed if condition (4.24) is satisfied for the optimal compensable input, i.e., if

$$\max_{\boldsymbol{w}^k} \min_{\boldsymbol{u}^k} \max_{\boldsymbol{q}^k} {J^{(s)}}^k < 0, \tag{4.26}$$

The value of γ thus specifies an upper bound for the convergence of the compensable summed error and for this reason the variable is referred to as the maximum convergence ratio hereafter.

The expression for the objective function in equation (4.23) contains the dependent variables $\boldsymbol{w}^{k+1}$ and $\boldsymbol{p}^k$. These variables can be eliminated from the objective function using the expression for the standard-plant in equation (4.15), yielding

$$\begin{aligned} {J^{(s)}}^k = \boldsymbol{w}^{k+1T}\boldsymbol{w}^{k+1} + \boldsymbol{p}^{kT}\boldsymbol{p}^k - \gamma^2\boldsymbol{w}^{kT}\boldsymbol{w}^k - \gamma^2\boldsymbol{q}^{kT}\boldsymbol{q}^k = \\ \left(\boldsymbol{R}\boldsymbol{w}^k + \hat{\boldsymbol{G}}\boldsymbol{u}^k + \boldsymbol{N}\boldsymbol{q}^k\right)^T \left(\boldsymbol{R}\boldsymbol{w}^k + \hat{\boldsymbol{G}}\boldsymbol{u}^k + \boldsymbol{N}\boldsymbol{q}^k\right) \\ + \boldsymbol{u}^{kT}\boldsymbol{M}^T\boldsymbol{M}\boldsymbol{u}^k - \gamma^2\boldsymbol{w}^{kT}\boldsymbol{w}^k - \gamma^2\boldsymbol{q}^{kT}\boldsymbol{q}^k. \end{aligned} \tag{4.27}$$

This expression for the objective function depends only on the unknown effect of the uncertainty $\boldsymbol{q}^k$, the compensable input $\boldsymbol{u}^k$ and the compensable summed error $\boldsymbol{w}^k$. According to the convergence condition this objective should be negative for the worst case effect of $\boldsymbol{q}^k$. This maximising $\boldsymbol{q}^k$ is a function of the variables $\boldsymbol{u}^k$ and $\boldsymbol{w}^k$. According to the design objective in equation (4.25), the optimal compensable input $\boldsymbol{u}^k$ should minimise the objective function for the worst case $\boldsymbol{q}^k$. This optimal $\boldsymbol{u}^k$ is a function of the remaining variable $\boldsymbol{w}^k$ as expressed by equation (4.11). The expression of the objective function ${J^{(s)}}^k$ for the maximising $\boldsymbol{q}^k$ and the minimising $\boldsymbol{u}^k$ is referred to as the optimal objective function. This optimal objective function depends only on (quadratic terms of) $\boldsymbol{w}^k$. If the optimal objective function is negative definite with respect to the compensable summed error $\boldsymbol{w}^k$, then ${J^{(s)}}^k < 0$ for any value of $\boldsymbol{w}^k$ and thus condition (4.26) is satisfied. Previously it was concluded that condition (4.26) is a sufficient condition for realising robust convergence of the summed error with a convergence ratio of at most γ. Therefore, the negative definiteness of the optimal objective function with respect to the compensable summed error $\boldsymbol{w}^k$ is hereafter referred to as the sufficient condition for robust convergence, which is abbreviated as SCRC.

4.2.4 Remarks

In the previous subsections it is shown that the summed error converges with a convergence ratio of at most γ if the SCRC is satisfied for some $\gamma < 1$. The minimally achievable maximum convergence ratio can be found by an iterative search for the smallest γ for which the SCRC is satisfied for the given model uncertainty and robustness filter. If $\boldsymbol{R} = \boldsymbol{I}$, then convergence of the summed error implies convergence of the error to zero according to update equation (4.5). However, it is possible that the SCRC cannot be satisfied for any $\gamma < 1$ if $\boldsymbol{R} = \boldsymbol{I}$, which means that robust convergence of the error to zero cannot be guaranteed for the specified model uncertainty. In section 4.4 it is shown that it is possible to satisfy the SCRC for any value of γ and any size of the model uncertainty using a non-unity robustness filter. However, it is also shown that this could result in a non-zero final error.

The design objective for the optimal learning filter is defined such that the learning filter relates the compensable input $\boldsymbol{u}^k$ to the compensable summed error $\boldsymbol{w}^k$ optimally in the sense of equation (4.25). The same learning filter is used for the implementation of RILC to compute the feedforward $\boldsymbol{f}^k$ from the measured summed error $\boldsymbol{z}^k$ as in equation (4.4).

The next section describes two algorithms for the solution of the optimal learning filter for the specified design objective, yielding two implementations of RILC. Furthermore, in line with the derivation of the two algorithms for the optimal learning filter, two algorithms are proposed to check the SCRC.

4.3 Solutions of the optimal learning filter

The optimal learning filter should relate the compensable input $\boldsymbol{u}^k$ to the compensable summed error $\boldsymbol{w}^k$ optimally in the sense of equation (4.25) for the objective function in equation (4.23) and system equations (4.15). In this section two algorithms for the computation of the optimal learning filter are derived, yielding two algorithms for RILC. In line with the algorithms for the solution of the learning filter, two algorithms for checking the SCRC are derived.

In subsection 4.3.1 it is shown that an explicit expression for the optimal learning filter can be derived using the lifted system description. Besides, the lifted system description is used to formulate the optimal objective function in terms of the compensable summed error. This expression of the optimal objective function can be used to check the SCRC. The lifted expressions for the optimal learning filter and the optimal objective function are used in section 4.4 for the analysis of the convergence properties of RILC. However, the computation of the learning filter and checking the SCRC using the lifted expressions requires computations with lifted matrices, which makes the algorithms computationally demanding for long iterations.

A more efficient algorithm for the computation of the optimal learning filter for RILC can be derived using the state-space representation of the system and

dynamic game theory, which has been demonstrated before by Van de Wijdeven and Bosgra (2007a). However, their algorithm is only suited for LTI systems and does not include a robustness filter, while the application considered in this work requires an algorithm that is suited for LTV system with considerable model uncertainty. In subsection 4.3.2 dynamic game theory is used to derive an algorithm for the computation of the optimal learning filter for LTV systems with a (non-causal) robustness filter. Van de Wijdeven and Bosgra (2007a) use frequency domain analysis for checking the SCRC, which limits the applicability of the convergence analysis to LTI systems and does not account for the finite length of the iteration. In subsection 4.3.2 an algorithm is proposed to check the SCRC from the state-space representation of the system and optimal control theory. This convergence analysis is suited for LTV systems and explicitly accounts for the finite length of the iteration. The algorithms proposed in subsection 4.3.2 are computationally efficient, suited for LTV systems and use no current iteration data. These properties meet requirements on the ILC algorithm imposed by the objectives of this work (see section 1.2). The other requirements on the ILC algorithm following from the objective of this work concern the convergence of the error. The convergence properties of RILC are analysed in section 4.4. The suitability of the proposed RILC algorithm for the objectives of this work is evaluated experimentally by the application to the Stäubli RX90 robot. The experimental results are described in chapter 6.

Summarising, subsection 4.3.1 describes the derivation of the optimal feedforward update from the lifted equations, which yields an explicit lifted expression for the optimal learning filter, and subsection 3.3.2 describes the derivation of the optimal feedforward update using dynamic game theory, yielding an efficient algorithm for the optimal learning filter.

4.3.1 Solution using the lifted description

In this section the optimal feedforward update is solved using the lifted system description. The lifted expression for the objective function in equation (4.27) depends only on $\boldsymbol{q}^k$ and $\boldsymbol{u}^k$, which should maximise and minimise the objective function respectively (equation (4.25)), and $\boldsymbol{w}^k$, which is the input to the learning filter. The optimisation problem from section 4.2 is such that the compensable input $\boldsymbol{u}^k$ is selected first and subsequently $\boldsymbol{q}^k$ reacts to the selected $\boldsymbol{u}^k$, because $\boldsymbol{q}^k$ represents the output of the model uncertainty for the selected compensable input $\boldsymbol{u}^k$. In other words, the maximising input $\boldsymbol{q}^k$ is a function of the minimising input $\boldsymbol{u}^k$. Below, the optimal input $\boldsymbol{u}^k$ is derived taking into account that the maximising disturbance $\boldsymbol{q}^k$ can be a function of $\boldsymbol{u}^k$. The resulting optimal inputs are the so-called Stackelberg solution (Başar and Olsder, 1995) of the optimisation problem. After the derivation of this solution, the optimal input $\boldsymbol{u}^k$ is derived without taking into account that $\boldsymbol{q}^k$ is a function of $\boldsymbol{u}^k$. In that case the resulting solutions of $\boldsymbol{q}^k$ and $\boldsymbol{u}^k$ both optimise the objective function for the worst case effect of the other variable such that a deviation

of either of the inputs from their optimum yields a smaller or a larger objective function respectively. These optimal inputs are the so-called Nash solution (Başar and Olsder, 1995) of the optimisation problem. Both solutions are compared after their derivation and the conclusions from this comparison are used in subsection 4.3.2 for formulating a more efficient algorithm to compute the optimal input $\boldsymbol{u}^k$.

Derivation of the Stackelberg solution

Hereafter, the optimal $\boldsymbol{q}^k$ and $\boldsymbol{u}^k$ are derived taking into account that $\boldsymbol{q}^k$ can be a function of $\boldsymbol{u}^k$. The objective function in equation (4.27) has a maximum with respect to $\boldsymbol{q}^k$ if its second order derivatives with respect $\boldsymbol{q}^k$ is negative definite, i.e.,

$$2\boldsymbol{N}^T\boldsymbol{N} - 2\gamma^2\boldsymbol{I} < 0. \tag{4.28}$$

This inequality poses a constraint on the selection of the uncertainty post-weighting filter and $\boldsymbol{N}$. The condition is satisfied if the spectral norm of the post-weighting filter $\boldsymbol{N}$ is taken sufficiently small. The spectral norm of the post-weighting filter can be reduced by modification of the uncertainty weighting filters as in equation (4.2) with a sufficiently small value for the weight ratio β.

If condition (4.28) is satisfied, then the maximising $\boldsymbol{q}^k$ is obtained by equating the derivative of the objective function ${J^{(s)}}^k$ (equation (4.27)) with respect to this variable to zero, resulting in

$$2\boldsymbol{N}^T\left(\boldsymbol{R}\boldsymbol{w}^k + \hat{\boldsymbol{G}}\boldsymbol{u}^k + \boldsymbol{N}\boldsymbol{q}^k\right) - 2\gamma^2\boldsymbol{q}^k = \boldsymbol{O}. \tag{4.29}$$

Solving $\boldsymbol{q}^k$ from this equation yields

$$\boldsymbol{q}^k = \left(\gamma^2\boldsymbol{I} - \boldsymbol{N}^T\boldsymbol{N}\right)^{-1}\boldsymbol{N}^T\left(\boldsymbol{R}\boldsymbol{w}^k + \hat{\boldsymbol{G}}\boldsymbol{u}^k\right). \tag{4.30}$$

Substitution of this maximising $\boldsymbol{q}^k$ in the objective function ${J^{(s)}}^k$ (equation (4.27)) and rewriting the result using the Woodbury-formula (Golub and Van Loan, 1996) yields

$$\begin{aligned} {J^{(s)}}^k = {}& \boldsymbol{u}^{kT}\boldsymbol{M}^T\boldsymbol{M}\boldsymbol{u}^k - \gamma^2\boldsymbol{w}^{kT}\boldsymbol{w}^k \\ & + \left(\boldsymbol{R}\boldsymbol{w}^k + \hat{\boldsymbol{G}}\boldsymbol{u}^k\right)^T\gamma^2\left(\gamma^2\boldsymbol{I} - \boldsymbol{N}\boldsymbol{N}^T\right)^{-1}\left(\boldsymbol{R}\boldsymbol{w}^k + \hat{\boldsymbol{G}}\boldsymbol{u}^k\right). \end{aligned} \tag{4.31}$$

This objective function is subsequently minimised by $\boldsymbol{u}^k$. The objective has a minimum with respect to $\boldsymbol{u}^k$ if its second order derivatives with respect $\boldsymbol{u}^k$ is negative definite respectively, i.e.,

$$2\boldsymbol{M}^T\boldsymbol{M} + 2\hat{\boldsymbol{G}}^T\gamma^2\left(\gamma^2\boldsymbol{I} - \boldsymbol{N}\boldsymbol{N}^T\right)^{-1}\hat{\boldsymbol{G}} > 0. \tag{4.32}$$

This condition is satisfied if the system matrix $\hat{\boldsymbol{G}}$ is nonsingular and condition (4.28) is satisfied. Otherwise the selection of a non-singular pre-weighting filter $\boldsymbol{M}$ is sufficient to satisfy the inequality.

If condition (4.32) is satisfied, then the minimising $\boldsymbol{u}^k$ is obtained by equating the derivative of the objective function ${J^{(s)}}^k$ from equation (4.31) with respect to this variable to zero, yielding

$$2\hat{\boldsymbol{G}}^T\gamma^2\left(\gamma^2\boldsymbol{I}-\boldsymbol{N}\boldsymbol{N}^T\right)^{-1}\left(\boldsymbol{R}\boldsymbol{w}^k+\hat{\boldsymbol{G}}\boldsymbol{u}^k\right)+2\boldsymbol{M}^T\boldsymbol{M}\boldsymbol{u}^k=\boldsymbol{O}, \tag{4.33}$$

Solving $\boldsymbol{u}^k$ from equation (4.33) gives the following explicit expression for the optimal compensable input

$$\boldsymbol{u}^k=-\boldsymbol{L}\boldsymbol{w}^k, \tag{4.34}$$

where the optimal learning filter is of the form

$$\boldsymbol{L}=\left(\hat{\boldsymbol{G}}^T\gamma^2\left(\gamma^2\boldsymbol{I}-\boldsymbol{N}\boldsymbol{N}^T\right)^{-1}\hat{\boldsymbol{G}}+\boldsymbol{M}^T\boldsymbol{M}\right)^{-1}\hat{\boldsymbol{G}}^T\gamma^2\left(\gamma^2\boldsymbol{I}-\boldsymbol{N}\boldsymbol{N}^T\right)^{-1}\boldsymbol{R}. \tag{4.35}$$

Defining the following matrices

$$\boldsymbol{V'}=\gamma^2\left(\gamma^2\boldsymbol{I}-\boldsymbol{N}\boldsymbol{N}^T\right)^{-1}, \tag{4.36a}$$

$$\boldsymbol{W'}=\boldsymbol{M}^T\boldsymbol{M}, \tag{4.36b}$$

the optimal learning filter can be rewritten as

$$\boldsymbol{L}=\left(\hat{\boldsymbol{G}}^T\boldsymbol{V'}\hat{\boldsymbol{G}}+\boldsymbol{W'}\right)^{-1}\hat{\boldsymbol{G}}^T\boldsymbol{V'}\boldsymbol{R}. \tag{4.37}$$

Note that this expression for the learning filter is similar to the expression for the learning filter of NILC in equation (3.14) if $\boldsymbol{R}=\boldsymbol{I}$. The matrices $\boldsymbol{V}$ and $\boldsymbol{W}$ in the learning filter for NILC are the user-defined static weights on the error and feedforward update in the objective function. The matrices $\boldsymbol{V'}$ and $\boldsymbol{W'}$ in the learning filer for RILC are dynamic filters that depend on the value of the maximum convergence ratio γ and the uncertainty weighting filters $\boldsymbol{M}$ and $\boldsymbol{N}$. The effect of the choice of these variables on the convergence properties of RILC is analysed in more detail in section 4.4.

Substitution $\boldsymbol{u}^k$ from equation (4.34) in the objective function in equation (4.31) yields the following expression for the optimal objective function

$${J^{(s)}}^k=\boldsymbol{w}^{kT}\boldsymbol{X}\boldsymbol{w}^k, \tag{4.38}$$

where matrix $\boldsymbol{X}$ is a function of $\hat{\boldsymbol{G}}$, $\boldsymbol{M}$, $\boldsymbol{N}$, γ and $\boldsymbol{R}$. The SCRC is satisfied if the matrix $\boldsymbol{X}$ is negative definite.

The lifted expressions for the optimal learning filter and the optimal objective function are used for the analysis of the robustness and convergence

properties of RILC in section 4.4. However, the lifted expressions are not suited for practical implementation of RILC for long iterations, because the number of elements of the lifted matrices in these equations is proportional to the square of the length of the iteration and thus the computations are time-consuming for long iterations.

Derivation of the Nash solution

Hereafter, the optimal $\boldsymbol{q}^k$ and $\boldsymbol{u}^k$ are derived without taking into account that $\boldsymbol{q}^k$ can be a function of $\boldsymbol{u}^k$. The objective function is optimised with respect to both inputs for the worst case effect of the other input, such that a deviation of either of the inputs from their optimum yields a smaller or a larger objective function respectively. The objective function has a minimum with respect to $\boldsymbol{u}^k$ for any $\boldsymbol{q}^k$ and a maximum with respect to $\boldsymbol{q}^k$ for any $\boldsymbol{u}^k$ in case its second order derivatives with respect to these variables are positive definite and negative definite respectively, i.e.,

$$2\boldsymbol{N}^T\boldsymbol{N} - 2\gamma^2\boldsymbol{I} < 0, \tag{4.39a}$$

$$2\hat{\boldsymbol{G}}^T\hat{\boldsymbol{G}} + 2\boldsymbol{M}^T\boldsymbol{M} > 0. \tag{4.39b}$$

Again, condition (4.39a) is satisfied if the spectral norm of the post-weighting filter $\boldsymbol{N}$ is taken sufficiently small. Condition (4.39b) is satisfied if the system matrix $\hat{\boldsymbol{G}}$ is nonsingular and otherwise the selection of a non-singular pre-weighting filter $\boldsymbol{M}$ is sufficient to satisfy the condition.

If conditions (4.39) are satisfied, then the minimising $\boldsymbol{u}^k$ and the maximising $\boldsymbol{q}^k$ are obtained by equating the derivatives of the objective function ${J^{(s)}}^k$ (equation (4.27)) with respect to these variables to zero, resulting in

$$2\hat{\boldsymbol{G}}^T\left(\boldsymbol{R}\boldsymbol{w}^k + \hat{\boldsymbol{G}}\boldsymbol{u}^k + \boldsymbol{N}\boldsymbol{q}^k\right) + 2\boldsymbol{M}^T\boldsymbol{M}\boldsymbol{u}^k = \boldsymbol{O}, \tag{4.40}$$

$$2\boldsymbol{N}^T\left(\boldsymbol{R}\boldsymbol{w}^k + \hat{\boldsymbol{G}}\boldsymbol{u}^k + \boldsymbol{N}\boldsymbol{q}^k\right) - 2\gamma^2\boldsymbol{q}^k = \boldsymbol{O}. \tag{4.41}$$

Solving $\boldsymbol{q}^k$ from equation (4.41) yields

$$\boldsymbol{q}^k = \left(\gamma^2\boldsymbol{I} - \boldsymbol{N}^T\boldsymbol{N}\right)^{-1}\boldsymbol{N}^T\left(\boldsymbol{R}\boldsymbol{w}^k + \hat{\boldsymbol{G}}\boldsymbol{u}^k\right), \tag{4.42}$$

Substitution of this expression for $\boldsymbol{q}^k$ in equation (4.40) gives

$$\begin{aligned}\hat{\boldsymbol{G}}^T\left(\boldsymbol{I} + \boldsymbol{N}\left(\gamma^2 - \boldsymbol{N}^T\boldsymbol{N}\right)^{-1}\boldsymbol{N}^T\right)\left(\boldsymbol{R}\boldsymbol{w}^k + \hat{\boldsymbol{G}}\boldsymbol{u}^k\right) + \boldsymbol{M}^T\boldsymbol{M}\boldsymbol{u}^k = \\ \hat{\boldsymbol{G}}^T\gamma^2\left(\gamma^2\boldsymbol{I} - \boldsymbol{N}\boldsymbol{N}^T\right)^{-1}\left(\boldsymbol{R}\boldsymbol{w}^k + \hat{\boldsymbol{G}}\boldsymbol{u}^k\right) + \boldsymbol{M}^T\boldsymbol{M}\boldsymbol{u}^k = \boldsymbol{O},\end{aligned} \tag{4.43}$$

where the Woodbury-formula (Golub and Van Loan, 1996) is used to obtain the second equality. Solving $\boldsymbol{u}^k$ from equation (4.43) gives the following explicit

expression for the optimal compensable input

$$\boldsymbol{u}^k = -\boldsymbol{L}\boldsymbol{w}^k, \tag{4.44}$$

where the optimal learning filter is of the form

$$\boldsymbol{L} = \left(\hat{\boldsymbol{G}}^T\gamma^2\left(\gamma^2\boldsymbol{I} - \boldsymbol{N}\boldsymbol{N}^T\right)^{-1}\hat{\boldsymbol{G}} + \boldsymbol{M}^T\boldsymbol{M}\right)^{-1}\hat{\boldsymbol{G}}^T\gamma^2\left(\gamma^2\boldsymbol{I} - \boldsymbol{N}\boldsymbol{N}^T\right)^{-1}\boldsymbol{R}. \tag{4.45}$$

The optimal input $\boldsymbol{q}^k$ is obtained by substitution of equation (4.44) in equation (4.42). The optimal objective function is obtained by subsequent substitution of $\boldsymbol{q}^k$ from equation (4.44) and $\boldsymbol{u}^k$ from equations (4.44) and (4.45) in the objective function from equation (4.27).

Comparison

The conditions for the existence of the Stackelberg solution for the maximising input $\boldsymbol{q}^k$ (inequality (4.28)) and conditions for the Nash solution (inequality (4.39a)) are the same, but the conditions for the existence of the Stackelberg solution for the minimising input $\boldsymbol{u}^k$ (inequalities (4.32)) differs from the condition for the Nash solution (inequality (4.39b)). Nonetheless, the conditions for the existence of the Nash-solution (inequalities (4.39)) are sufficient to satisfy the condition for the existence of the Stackelberg solution for $\boldsymbol{u}^k$ (inequality (4.32)). This can be explained by the fact that the Stackelberg solution of $\boldsymbol{u}^k$ needs to be minimising only for the worst case $\boldsymbol{q}^k$, while the Nash solution of $\boldsymbol{u}^k$ needs to be minimising for any $\boldsymbol{q}^k$.

The optimal input $\boldsymbol{u}^k$ expressed in equation (4.34) with the learning matrix from equation (4.35) is the same input as expressed by equations (4.44) and (4.45). The optimal inputs $\boldsymbol{q}^k$ expressed by equations (4.30) and (4.42) are the same as well. Furthermore, the resulting values of the optimal objective functions are also the same. The Stackelberg and the Nash solution thus yield the same optimal inputs and the same optimal objective function if the conditions for their existence are satisfied. This result is consistent with known results from differential game theory.

From the previous it can be concluded that the minimising $\boldsymbol{u}^k$ and the optimal objective function for the considered (Stackelberg) problem can be computed without taking into account that $\boldsymbol{q}^k$ can be a function $\boldsymbol{u}^k$, i.e., by deriving the Nash solution. Moreover, the conditions for the existence of the Nash solutions for $\boldsymbol{u}^k$ and $\boldsymbol{q}^k$ are sufficient for the existence of the Stackelberg solution for those variables, where $\boldsymbol{q}^k$ can be a function of $\boldsymbol{u}^k$.

4.3.2 Solution using dynamic game theory

In this section the design objective that defines the optimal learning filter for RILC is rewritten to a finite-horizon dynamic game. This dynamic game can be

solved using an existing algorithm, yielding a computationally efficient algorithm for the optimal learning filter. Besides, the solution of the dynamic game yields an expression for the optimal objective function in terms of the compensable summed error. This expression is used to derive a computationally efficient procedure for checking the SCRC. The procedure is based on optimal control theory.

Assumptions

Before the formulation of the dynamic game, some assumptions are made on the dynamics associated with $\hat{\boldsymbol{G}}$, $\boldsymbol{N}$, $\boldsymbol{M}$ and $\boldsymbol{R}$.

As stated in section 4.1, it is assumed that RILC is applied to a strictly proper system and thus the estimate of the system dynamics $\hat{\boldsymbol{G}}$ is strictly proper as well. Moreover, because there is no uncertainty about the properness of the system, strictly proper dynamics can be taken for either the pre-weighting filter $\boldsymbol{M}$ or the post-weighting filter $\boldsymbol{N}$. In this work strictly proper dynamics are taken for $\boldsymbol{M}$ and proper dynamics is taken for $\boldsymbol{N}$.

In subsection 4.4.3 it is argued that a smaller final error can be realised using a non-causal robustness filter than using a causal robustness filter. Therefore, the robustness filter $\boldsymbol{R}$ is allowed to be non-causal in the derivation of the optimal learning filter. A non-causal robustness filter needs special attention in the solution of the optimal learning filter with dynamic game theory, because conventional dynamic game theory assumes causal dynamics. It is assumed that the non-causal robustness filter is implemented as an anti-causal part and a causal part connected in series, i.e.,

$$\boldsymbol{R} = \overrightarrow{\boldsymbol{R}}\overleftarrow{\boldsymbol{R}}^{T}, \tag{4.46}$$

where the causal part is denoted as $\overrightarrow{\boldsymbol{R}}$ and the anti-causal part as $\overleftarrow{\boldsymbol{R}}^{T}$. Matrices $\overrightarrow{\boldsymbol{R}}$ and $\overleftarrow{\boldsymbol{R}}$ are associated with causal and proper dynamics and the transpose on the second part makes this part anti-causal. A phase-less robustness filter can be implemented by taking the same dynamics for $\overrightarrow{\boldsymbol{R}}$ and $\overleftarrow{\boldsymbol{R}}$. It should be noted that any matrix $\boldsymbol{R}$ can be decomposed in a causal and an anti-causal part as in equation (4.46) using LU-factorisation. The following variables are introduced as the output of the anti-causal part and the causal part of the robustness filter with the compensable summed error as the input

$$\overleftarrow{\boldsymbol{r}}^{k} = \overleftarrow{\boldsymbol{R}}^{T}\boldsymbol{w}^{k}, \tag{4.47a}$$

$$\overrightarrow{\boldsymbol{r}}^{k} = \overrightarrow{\boldsymbol{R}}\overleftarrow{\boldsymbol{r}}^{k}. \tag{4.47b}$$

Table 4.1 lists the state vectors and the state-space matrices that are associated with the lifted matrices $\hat{\boldsymbol{G}}$, $\boldsymbol{M}$, $\boldsymbol{N}$, $\overrightarrow{\boldsymbol{R}}$ and $\overleftarrow{\boldsymbol{R}}$. These variables are used hereafter for the solution of the optimal learning filter for RILC using dynamic game theory.

The optimal learning filter

The design objective for the optimal learning filter is expressed in terms of the compensable input, which is the output of the learning filter, by equation (4.25), and the objective function is given in equation (4.23). These equations are formulated in terms of the time-samples of the various signals as

$$\check{u}_i^k = \arg\min_{u_i^k} \max_{q_i^k} J^{(s)^k}, \tag{4.48}$$

$$J^{(s)^k} \triangleq \sum_{i=1}^{N_i-1} \left(w_{i+1}^{k+1T} w_{i+1}^{k+1} + {p_{i+1}^k}^T p_{i+1}^k - \gamma^2 {w_{i+1}^k}^T w_{i+1}^k - \gamma^2 {q_{i+1}^k}^T q_{i+1}^k \right). \tag{4.49}$$

Since $\hat{\boldsymbol{G}}$ and $\boldsymbol{M}$ are strictly proper, the time samples q_1^k, p_1^k, w_1^k and w_1^{k+1} are not a function of u_i^k and these are not included in the objective function.

The time-samples of the dependent variables w_i^{k+1} and p_i^k are related to u_i^k, q_i^k and w_i^k by the state-space equations associated with the standard-plant equation (4.15). These state-space equations are formulated hereafter using the state variables and state-space matrices from table 4.1. The state-space equations associated with the first row of equation (4.15) are

$$x_{i+1}^k = A_i x_i^k + B_i u_i^k, \tag{4.50a}$$

$$\nu_{i+1}^k = A_i^{(N)} \nu_i^k + B_i^{(N)} q_i^k, \tag{4.50b}$$

$$x_1^k = O, \nu_1^k = O, \tag{4.50c}$$

$$w_i^{k+1} = \overrightarrow{r}_i^k + C_i x_i^k + C_i^{(N)} \nu_i^k + D_i^{(N)} q_i^k, \tag{4.50d}$$

and the state-space equations associated with the second row equation (4.15) are

$$\mu_{i+1}^k = A_i^{(M)} \mu_i^k + B_i^{(M)} u_i^k, \tag{4.51a}$$

$$\mu_1^k = O, \tag{4.51b}$$

$$p_i^k = C_i^{(M)} \mu_i^k. \tag{4.51c}$$

lifted matrix	$\hat{\boldsymbol{G}}$	$\boldsymbol{M}$	$\boldsymbol{N}$	$\overrightarrow{\boldsymbol{R}}$	$\overleftarrow{\boldsymbol{R}}$
state vector	x_i	μ_i	ν_i	$\overrightarrow{\rho}_i$	$\overleftarrow{\rho}_i$
state-transition matrix	A_i	$A_i^{(M)}$	$A_i^{(N)}$	$A_i^{(\overrightarrow{R})}$	$A_i^{(\overleftarrow{R})}$
input matrix	B_i	$B_i^{(M)}$	$B_i^{(N)}$	$B_i^{(\overrightarrow{R})}$	$B_i^{(\overleftarrow{R})}$
output matrix	C_i	$C_i^{(M)}$	$C_i^{(N)}$	$C_i^{(\overrightarrow{R})}$	$C_i^{(\overleftarrow{R})}$
feedthrough matrix	-	-	$D_i^{(N)}$	$D_i^{(\overrightarrow{R})}$	$D_i^{(\overleftarrow{R})}$

Table 4.1: The state variables and matrices corresponding to the lifted matrices

The state-space equations associated with the anti-causal part of the robustness filter (equation (4.47a)) are

$$\overleftarrow{\rho}^{k}_{i-1} = A_i^{(\overleftarrow{R})T}\overleftarrow{\rho}^{k}_{i} + C_i^{(\overleftarrow{R})T} w_i^k, \tag{4.52a}$$

$$\overleftarrow{\rho}^{k}_{N_i} = O, \tag{4.52b}$$

$$\overleftarrow{r}^{k}_{i} = B_i^{(\overleftarrow{R})T}\overleftarrow{\rho}^{k}_{i} + D_i^{(\overleftarrow{R})T} w_i^k, \tag{4.52c}$$

and the state-space equations associated with the causal part of the robustness filter (equation (4.47b)) are

$$\overrightarrow{\rho}^{k}_{i+1} = A_i^{(\overrightarrow{R})}\overrightarrow{\rho}^{k}_{i} + B_i^{(\overrightarrow{R})}\overleftarrow{r}^{k}_{i}, \tag{4.53a}$$

$$\overrightarrow{\rho}^{k}_{1} = O, \tag{4.53b}$$

$$\overrightarrow{r}^{k}_{i} = C_i^{(\overrightarrow{R})}\overrightarrow{\rho}^{k}_{i} + D_i^{(\overrightarrow{R})}\overleftarrow{r}^{k}_{i}. \tag{4.53c}$$

Equations (4.48)-(4.53) define a dynamic game with inputs w_i^k, u_i^k and q_i^k, where inputs u_i^k and q_i^k should minimise and maximise the quadratic objective function in equation (4.49) respectively. Başar and Olsder (1995) describe algorithms to compute the optimal opposing inputs of dynamic games with a quadratic objective function that only depends on the opposing inputs and the state of a causal dynamic system. Such dynamic game is formulated using the state-equations of the system, the uncertainty weighting filters and the causal part of the robustness filter (equations (4.50), (4.51) and (4.53)). The anti-causal part of the robustness filter (equation (4.52)) is not included in the formulated dynamic game, because its state equation is not causal and the output of this filter, which is the input to the causal part of the robustness filter and thus an input of the formulated dynamic game, only depends on the input w_i^k and not on the minimising input u_i^k or the maximising input q_i^k. The effect of the anti-causal part of the robustness filter is added to the solution of the dynamic game later on to obtain the algorithm for the learning filter. The dynamic game is formulated by concatenating state equations (4.50), (4.51) and (4.53), extending the state with the inputs of these state-equations and expressing the objective function from equation (4.49) in terms of the state and the input of the extended state equation, yielding

$$\tilde{x}_{i+1} = \tilde{A}_i\tilde{x}_i + \tilde{B}_i^{(u)}\tilde{u}_i + \tilde{B}_i^{(q)}\tilde{q}_i + \tilde{B}_i^{(v)}\tilde{v}_i, \tag{4.54a}$$

$$\check{\tilde{u}}_i = \arg\min_{\tilde{u}_i}\max_{\tilde{q}_i} {J^{(s)}}^k, \tag{4.54b}$$

$${J^{(s)}}^k = \sum_{i=1}^{N-1}\left(\tilde{x}_{i+1}{}^T\tilde{Q}_{i+1}\tilde{x}_{i+1} + \tilde{u}_i{}^T\tilde{R}_i^{(u)}\tilde{u}_i + \tilde{q}_i{}^T\tilde{R}_i^{(q)}\tilde{q}_i + \tilde{w}_i{}^T\tilde{R}_i^{(w)}\tilde{w}_i\right), \tag{4.54c}$$

where,

$$\tilde{x}_i = \begin{bmatrix} x_i^{kT} & \overrightarrow{\rho}^{kT}_{i} & \mu_i^{kT} & \nu_i^{kT} & u_{i-1}^{k}{}^T & q_i^{kT} & \overleftarrow{r}^{kT}_{i} \end{bmatrix}^T \tag{4.55a}$$

$$\tilde{x}_1 = \begin{bmatrix} O & O & O & O & O & O & O \end{bmatrix}^T, \tag{4.55b}$$

$$\tilde{u}_i = u_i^k, \tag{4.55c}$$

$$\tilde{q}_i = q_{i+1}^k, \tag{4.55d}$$

$$\tilde{v}_i = \overleftarrow{r}^k_{i+1}, \tag{4.55e}$$

$$\tilde{w}_i = w_{i+1}^k \tag{4.55f}$$

$$\tilde{A}_i = \begin{bmatrix} A_i & O & O & O & O & O & O \\ O & A_i^{(\overrightarrow{R})} & O & O & O & O & B_i^{(\overrightarrow{R})} \\ O & O & A_i^{(M)} & O & O & O & O \\ O & O & O & A_i^{(N)} & O & B_i^{(N)} & O \\ O & O & O & O & O & O & O \\ O & O & O & O & O & O & O \\ O & O & O & O & O & O & O \end{bmatrix}, \tag{4.55g}$$

$$\tilde{B}_i^{(u)} = \begin{bmatrix} B_i^T & O & B_i^{(M)T} & O & I & O & O \end{bmatrix}^T, \tag{4.55h}$$

$$\tilde{B}_i^{(q)} = \begin{bmatrix} O & O & O & O & O & I & O \end{bmatrix}^T, \tag{4.55i}$$

$$\tilde{B}_i^{(v)} = \begin{bmatrix} O & O & O & O & O & O & I \end{bmatrix}^T, \tag{4.55j}$$

$$\tilde{C}_i^{(w)} = \begin{bmatrix} C_i & C_i^{(\overrightarrow{R})} & O & C_i^{(N)} & O & D_i^{(N)} & D_i^{(\overrightarrow{R})} \end{bmatrix}, \tag{4.55k}$$

$$\tilde{C}_i^{(p)} = \begin{bmatrix} O & O & C_i^{(M)} & O & O & O & O \end{bmatrix}, \tag{4.55l}$$

$$\tilde{C}_i^{(u)} = \begin{bmatrix} O & O & O & O & I & O & O \end{bmatrix}, \tag{4.55m}$$

$$\tilde{Q}_{i+1} = \tilde{C}_{i+1}^{(w)T} \tilde{C}_{i+1}^{(w)} + \tilde{C}_{i+1}^{(p)\,T} \tilde{C}_{i+1}^{(p)} - \tilde{C}_{i+1}^{(u)\,T} \tilde{R}_i^{(u)} \tilde{C}_{i+1}^{(u)}, \tag{4.55n}$$

$$\tilde{R}_i^{(q)} = -\gamma^2 I, \tag{4.55o}$$

$$\tilde{R}_i^{(w)} = -\gamma^2 I. \tag{4.55p}$$

Matrix $\tilde{R}_i^{(u)}$ may be any positive definite symmetric matrix. This matrix is used for the implementation of the algorithm that solves the dynamic game, but its value does not affect the optimal inputs $\tilde{u}_i$ and $\tilde{q}_i$ and the resulting value of the optimal objective function. The last term of the objective function depends on the input $\tilde{w}_i = w_{i+1}^k$, which is not a function of the minimising input u_i^k or the maximising input q_i^k, but only related to the input $\tilde{v}_i = \overleftarrow{r}^k_{i+1}$ via the inverse of the anti-causal part of the robustness filter (equation (4.52)). This term can thus be ignored in the computation of the optimal inputs u_i^k and q_i^k. Without the last term in the objective function, equations (4.54) define the affine quadratic two-person zero-sum dynamic game as analysed by Başar and Olsder (1995).

Başar and Olsder (1995) give the solution of the inputs of the dynamic game for the case where one optimising input is a function of the other optimising input (the Stackelberg solution, page 278) and the case where both inputs

optimise the objective independently (the Nash solution, page 275). The algorithm to solve the first problem requires the solution of a coupled causal and anti-causal matrix difference equation, while the algorithm to solve the second problem involves only a single anti-causal matrix convolution. The problem defined in section 4.2 is of the Stackelberg type, but it is concluded in subsection 4.3.1 that the Nash solution of the optimisation problem yields the same optimal inputs. In this work the algorithm to compute the Nash solution is used instead of the algorithm to solve the Stackelberg solution. The advantage of this approach is that no algorithm to solve the complex coupled matrix difference equation of the Stackelberg solution is needed. The procedure to solve the dynamic game is given at the end of appendix B.2 and yields the optimal values of the inputs $\tilde{u}_i$ and $\tilde{q}_i$ as a function of $\tilde{v}_i$. The procedure consists of the solution of a non-stationary Riccati difference equation, the solution of a causal state-convolution and an anti-causal state-convolution. The number of computational operations of the procedure scales linearly with the length of the iteration. The first step of the procedure checks if the optimal inputs $\tilde{u}_i$ and $\tilde{q}_i$ indeed minimise and maximise the objective function. These conditions are equivalent to conditions (4.39) for the lifted solution of the optimal learning filter described in subsection 4.3.1. As explained in that section, these conditions are sufficient and can be satisfied by an appropriate selection of γ, $\boldsymbol{N}$ and $\boldsymbol{M}$.

The solution of the dynamic game from appendix B.2 provides an algorithm to compute the optimal compensable input u_i^k from the output of the anti-causal part of the robustness filter $\overleftarrow{r}_i^k$. The anti-causal part of the robustness filter relates its output $\overleftarrow{r}_i^k$ to the compensable summed error w_i^k as in equations (4.52). Together, the solution of the dynamic game and the anti-causal part of the robustness filter relate the optimal compensable input to the compensable summed error and they compose the algorithm for the optimal learning filter.

The optimal objective function

The solution of the dynamic game and the anti-causal part of the robustness filter constitute an algorithm to compute the optimal compensable input $\tilde{u}_i$ and the worst case effect of the uncertainty $\tilde{q}_i$ from the compensable summed error $\tilde{w}_i$. Hereafter it is shown that those optimal inputs result in an optimal objective function that depends only on the compensable summed error. Moreover it is shown that optimal control theory can be used to check if the optimal objective function is negative definite with respect to the compensable summed error, which is the SCRC.

Appendix B.2 shows that for the optimal inputs $\tilde{u}_i$ and $\tilde{q}_i$ the objective

function of the dynamic game can be expressed as

$$J^{(s)^k} = \left(2\tilde{\eta}_1 + \tilde{S}_1\tilde{x}_1\right)^T \tilde{x}_1 + \sum_{i=1}^{N-1} \Big(\left(2\tilde{\eta}_{i+1} + \tilde{S}_{i+1}\tilde{B}_i^{(v)}\tilde{v}_i\right) \tilde{L}_i^{-1}\tilde{B}_i^{(v)}\tilde{v}_i \\ -\tilde{\eta}_{i+1}^T\tilde{Q}_{i+1}^{(\eta)}\tilde{\eta}_{i+1} + {\tilde{w}_i}^T\tilde{R}_i^{(w)}\tilde{w}_i \Big), \quad (4.56)$$

where

$$\tilde{\eta}_i = \tilde{A}_i^T\tilde{S}_{i+1}\tilde{L}_i^{-1}\left(\tilde{P}_i\tilde{\eta}_{i+1} + \tilde{B}_i^{(v)}\tilde{v}_i\right) + \tilde{A}_i^T\tilde{\eta}_{i+1}, \quad (4.57a)$$

$$\tilde{\eta}_N = O. \quad (4.57b)$$

Matrices $\tilde{S}_i$, $\tilde{L}_i$, $\tilde{P}_i$ and $\tilde{Q}_i^{(\eta)}$ are functions of matrices $\tilde{A}_i$, $\tilde{B}_i^{(u)}$, $\tilde{B}_i^{(q)}$, $\tilde{B}_i^{(v)}$, $\tilde{Q}_i$, $\tilde{R}_i^{(q)}$ and $\tilde{R}_i^{(w)}$ as defined in appendix B.2. The expression for the optimal objective function in equation (4.56) is a function of the compensable summed error $\tilde{w}_i = w_{i+1}^k$ and the output of the anti-causal part of the robustness filter $\tilde{v}_i = \overleftarrow{r}^k{}_{i+1}$. According to the expression for the anti-causal part of the robustness filter in equations (4.52), its output is a function of the compensable summed error as well. Thus, the optimal objective function is only a function of the compensable summed error.

The convergence analysis in subsection 4.2.3 states that the SCRC is satisfied if the optimal objective function is negative definite with respect to the compensable summed error. This condition is verified by maximising the optimal objective function with respect to $\tilde{w}_i = w_{i+1}^k$ and checking if this maximum is not positive. The maximisation objective, the expression for the optimal objective function in equation (4.56), the co-state equation (4.57) and the state equation (4.52) of the anti-causal part of the robustness filter constitute the following dynamic game

$$\tilde{\tilde{\eta}}_{i-1} = \tilde{\tilde{A}}_i\tilde{\tilde{\eta}}_i + \tilde{\tilde{B}}_i^{(u)}\tilde{\tilde{u}}_i, \quad (4.58a)$$

$$\tilde{\tilde{u}}_i = \arg\max_{\tilde{\tilde{u}}_i} J^{(s)^k}, \quad (4.58b)$$

$$J^{(s)^k} = \sum_{i=2}^{N}\left({\tilde{\tilde{\eta}}_{i-1}}^T\tilde{\tilde{Q}}_{i-1}\tilde{\tilde{\eta}}_{i-1} + {\tilde{\tilde{u}}_i}^T\tilde{\tilde{R}}_i^{(u)}\tilde{\tilde{u}}_i\right), \quad (4.58c)$$

where, assuming $\tilde{x}_1 = O$,

$$\tilde{\tilde{\eta}}_i = \begin{bmatrix}\tilde{\eta}_{i+1}^T & \overleftarrow{\rho}^{k\,T}_i & \tilde{v}_i^T\end{bmatrix}^T = \begin{bmatrix}\tilde{\eta}_{i+1}^T & \overleftarrow{\rho}^{k\,T}_i & \overleftarrow{r}^{k}_{i+1}{}^T\end{bmatrix}^T, \quad (4.59a)$$

$$\tilde{\tilde{\eta}}_{N_i} = \begin{bmatrix}O & O & O\end{bmatrix}^T, \quad (4.59b)$$

$$\tilde{\tilde{u}}_i = \tilde{w}_{i-1} = w_i^k, \quad (4.59c)$$

$$\tilde{\tilde{A}}_i = \begin{bmatrix} \tilde{A}_i^T \left(I + \tilde{S}_{i+1} \tilde{L}_i^{-1} \tilde{P}_i \right) & O & \tilde{A}i^T \tilde{S}_{i+1} \tilde{L}_i^{-1} \tilde{B}_i^{(v)} \\ O & A_i^{(\overleftarrow{R})T} & O \\ O & B_i^{(\overleftarrow{R})T} & O \end{bmatrix}, \tag{4.59d}$$

$$\tilde{\tilde{B}}_i^{(u)} = \begin{bmatrix} O & C_i^{(\overleftarrow{R})} & D_i^{(\overleftarrow{R})} \end{bmatrix}^T, \tag{4.59e}$$

$$\tilde{\tilde{Q}}_i = \begin{bmatrix} \tilde{Q}_{i+1}^{(\eta)} & O & \tilde{L}_i^{-1} \tilde{B}_i^{(v)} \\ O & O & O \\ \tilde{B}_i^{(v)T} \tilde{L}_i^{-T} & O & \tilde{B}_i^{(v)T} \tilde{S}_{i+1} \tilde{L}_i^{-1} \tilde{B}_i^{(v)} \end{bmatrix}, \tag{4.59f}$$

$$\tilde{\tilde{R}}_i^{(u)} = \tilde{R}_i^{(w)} = -\gamma^2 I. \tag{4.59g}$$

Equations (4.58) define an affine quadratic discrete-time optimal control problem with an anti-causal state convolution. The affine quadratic discrete-time optimal control problem with a causal state-convolution is analysed by Başar and Olsder (1995) and the procedure to solve this problem is given at the end of appendix B.1. Replacing each time-index i by $N_i + 1 - i$ in this procedure yields a procedure to solve the optimal control problem with anti-causal state-convolution defined in equations (4.58). The procedure yields the optimal value of $\tilde{\tilde{u}}_i = \tilde{w}_{i-1} = w_i^k$ and the optimal value of the objective function. The optimal values of the compensable summed error and the optimal objective function are zero, because the compensable summed error w_i^k is the only input to the state-equation and $\tilde{\tilde{\eta}}_{N_i} = 0$. The second step of the procedure described in appendix B.1 verifies if the optimal value of the objective function is a maximum with respect to the compensable summed error. If that condition is satisfied then the maximum of the objective function is zero and thus the optimal objective function is negative definite with respect to the compensable summed error. The negative definiteness of the optimal objective function with respect to the compensable summed error is the SCRC formulated in subsection 4.2.3. The solution of the anti-causal optimal control problem thus gives a procedure to check the SCRC. The procedure consists of the solution of a non-stationary Riccati difference equation and checking if a time-varying matrix related to the solution of this difference equation is negative definite at each time step. The number of computational operations scales linearly with the length of the iteration.

Discussion

The number of computational operations to compute the optimal learning filter and to check the SCRC using the algorithms described in this subsection depends on the state-dimension and scales linearly with the number of time steps N_i. The number of computational operations to compute the optimal learning filter using the algorithm from subsection 4.3.1 does not depend on the state-dimension but scales at least quadratically with N_i as it involves the computation of the inverse of a lifted matrix. The procedure described in this

subsection is thus more efficient than the algorithm from subsection 4.3.1 for systems with a low state dimension and a large number of time steps. Moreover, the algorithm is suited for LTV systems and uses no measurements of the error from previous iterations. The algorithm thus meets the requirements following from the objective of this thesis (see section 1.2) and therefore it is used to compute the optimal learning filter for the experiments of which the results are described in chapter 6.

4.4 Convergence Analysis

In this section the convergence properties of the proposed RILC algorithm are analysed. In particular, the effects of the robustness filter and the uncertainty weighting filters on the convergence properties are investigated. In subsection 4.4.1 the conditions for convergence of the error and the summed error are derived. Moreover, an expression for the final error is given. In subsection 4.4.2 the convergence analysis is elaborated for systems with a particular type of model uncertainty for which the contribution of components of the compensable summed error to the optimal objective function can be decoupled. In subsection 4.4.3 the results from the convergence analysis are used to formulate guidelines for the selection of the maximum convergence ratio, the uncertainty weighting filters and the robustness filter.

4.4.1 Convergence analysis

Convergence of the summed error

In section 4.2 the convergence of the summed error is analysed for the formulation of the design objective for the learning filter of RILC. The objective function is formulated such that the summed error converges robustly and monotonically with a convergence ratio of at most γ if the optimal objective function, i.e., the objective function for the optimal compensable input and the worst-case effect of the uncertainty, is negative definite with respect to the compensable summed error. This condition, referred to as the SCRC, can be checked using the algorithms proposed in section 4.3.

From equation (4.5) it follows that if $\boldsymbol{R} = \boldsymbol{I}$ then convergence of the summed error implies that the error converges *to zero*. Thus if the SCRC is satisfied and no robustness filter is used, then the error converges robustly to zero. However, if the model uncertainty is large, then the SCRC cannot be satisfied for $\boldsymbol{R} = \boldsymbol{I}$ and a non-unity robustness filter is needed to guarantee robust convergence. An example of the relation between the choice of the robustness filter and the allowable size of the model uncertainty for a specific type of uncertainty is given in subsection 4.4.2.

Hereafter, it is shown that the SCRC is satisfied for any size of the model uncertainty by taking $\boldsymbol{R} = \boldsymbol{O}$. The optimal learning filter, which is expressed

by equation (4.35), becomes $\boldsymbol{L} = \boldsymbol{O}$ for this choice of the robustness filter. Substitution of $\boldsymbol{L} = \boldsymbol{O}$ in equation (4.11) gives $\boldsymbol{u}^k = \boldsymbol{O}$ and substitution of $\boldsymbol{u}^k = \boldsymbol{O}$ in the second row of equation (4.15) gives $\boldsymbol{p}^k = \boldsymbol{O}$, which according to equations (4.13b) gives $\boldsymbol{q}^k = \boldsymbol{O}$. Substitution of $\boldsymbol{R} = \boldsymbol{O}$, $\boldsymbol{u}^k = \boldsymbol{O}$ and $\boldsymbol{q}^k = \boldsymbol{O}$ in the first row of equation (4.15) gives $\boldsymbol{w}^{k+1} = \boldsymbol{O}$. Substitution of $\boldsymbol{p}^k = \boldsymbol{O}$, $\boldsymbol{q}^k = \boldsymbol{O}$ and $\boldsymbol{w}^{k+1} = \boldsymbol{O}$ in the objective function from equation (4.23) yields $J^{(s)^k} = -\gamma^2 \left\|\boldsymbol{w}^k\right\|_2^2$. This objective function is negative definite with respect to the compensable summed error and thus the SCRC is satisfied if $\boldsymbol{R} = \boldsymbol{O}$ for any model error. It is thus possible to select a robustness filter such that the SCRC is satisfied for any model uncertainty. However, it should be noted that the case $\boldsymbol{R} = \boldsymbol{O}$ is trivial because no learning is applied.

Convergence of the error

Firstly, the convergence of the error is analysed if no robustness filter is applied, i.e., $\boldsymbol{R} = \boldsymbol{I}$, then the effect of a robustness filter on the convergence of the error is considered. Substitution of equation (4.4) in equation (4.1) and subtracting the result for iteration k from the result for $k+1$ yields

$$\boldsymbol{e}^{k+1} = \boldsymbol{e}^k - \left(\hat{\boldsymbol{G}} + \boldsymbol{N}\boldsymbol{\Delta}\boldsymbol{M}\right)\boldsymbol{L}\left(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k\right). \tag{4.60}$$

Substituting equation (4.3), which is the update equation for $\boldsymbol{R} = \boldsymbol{I}$, gives

$$\boldsymbol{e}^{k+1} = \left(\boldsymbol{I} - \hat{\boldsymbol{G}}\boldsymbol{L} - \boldsymbol{N}\boldsymbol{\Delta}\boldsymbol{M}\boldsymbol{L}\right)\boldsymbol{e}^k. \tag{4.61}$$

Thus, the error thus converges monotonically *to zero* if

$$\left\|\boldsymbol{I} - \hat{\boldsymbol{G}}\boldsymbol{L} - \boldsymbol{N}\boldsymbol{\Delta}\boldsymbol{M}\boldsymbol{L}\right\|_{i2} < 1. \tag{4.62}$$

Note that this condition is identical to condition (4.7) for convergence of the summed error if $\boldsymbol{R} = \boldsymbol{I}$. Thus, if no robustness filter is applied, then the conditions for robust monotonic convergence of the summed error and the condition for robust monotonic convergence of the error *to zero* coincide. Both conditions are satisfied if the SCRC is satisfied, which can be checked using the algorithms proposed in section 4.3.

The SCRC might not be satisfied if the model uncertainty is large and no robustness filter is applied. However, the SCRC can be satisfied for any model uncertainty by choosing a non-unity robustness. Satisfying the SCRC implies monotonic convergence of the summed error and thus convergence of the error, though monotonic convergence of the error is not necessarily guaranteed for a non-unity robustness filter. Nevertheless, an expression for the final error can be derived. If the summed error converges then equations (4.7) and (4.8) hold and thus

$$\boldsymbol{z}^\infty = \left(\boldsymbol{I} - \boldsymbol{R} + \hat{\boldsymbol{G}}\boldsymbol{L} + \boldsymbol{N}\boldsymbol{\Delta}\boldsymbol{M}\boldsymbol{L}\right)^{-1}\boldsymbol{d}. \tag{4.63}$$

Combining this relation with equation (4.5) yields the following expression for the final error

$$\boldsymbol{e}^{\infty} = (\boldsymbol{I} - \boldsymbol{R})\, \boldsymbol{z}^{\infty} = (\boldsymbol{I} - \boldsymbol{R}) \left(\boldsymbol{I} - \boldsymbol{R} + \hat{\boldsymbol{G}}\boldsymbol{L} + \boldsymbol{N}\boldsymbol{\Delta}\boldsymbol{M}\boldsymbol{L} \right)^{-1} \boldsymbol{d}. \tag{4.64}$$

This equation shows that for $\boldsymbol{R} = \boldsymbol{I}$ the error converges to zero, which is consistent with the previous analysis. For any other choice of the robustness filter the final error could be non-zero and depends on the actual model error. The extreme case $\boldsymbol{R} = \boldsymbol{O}$, which implies $\boldsymbol{L} = \boldsymbol{O}$ by equation (4.35), results in $\boldsymbol{e}^{\infty} = \boldsymbol{d}$, which is trivial because no learning is applied in this case.

4.4.2 Decoupled convergence analysis

In this subsection the convergence analysis for RILC is elaborated for square systems with a special type of model uncertainty. The only uncertainty is assumed to be the size of the singular values of the estimated lifted system matrix. The contribution of components of the compensable summed error to the optimal objective function can be decoupled for this case. The SCRC is then satisfied if the objective function is negative definite with respect to each of the components of the compensable summed error, which facilitates the convergence analysis. The results from the analysis are valid if there is no model uncertainty, i.e., $\boldsymbol{M} = \boldsymbol{O}$. Moreover, the decoupled analysis is illustrative other types of uncertainty like uncertainty in the modelled frequency response of an LTI system, because a close relation exists between the singular values of the lifted system matrix of an LTI system and the frequency response of the system (see subsection 3.4.3).

Preliminaries

Consider the singular value decomposition of the estimated system matrix $\hat{\boldsymbol{G}}$ given in equation (3.45). It is assumed that the only uncertainty is the size of the singular values of the estimated system matrix, i.e., it is known that the response of the real system to a column of $\boldsymbol{T}$ is proportional to the corresponding column of $\boldsymbol{U}$, but the gain of the response is uncertain. The additive uncertainty in the size of each singular value s_i is assumed to be bounded as $\delta_i m_i$ with $|\delta_i| < 1$.

An appropriate set of uncertainty weighting filters for the considered type of uncertainty is

$$\boldsymbol{M} = \tfrac{1}{\beta}\boldsymbol{T}\bar{\boldsymbol{M}}\boldsymbol{T}^{T}, \tag{4.65a}$$

$$\boldsymbol{N} = \beta\boldsymbol{I}, \tag{4.65b}$$

where the pre-weighting filter $\bar{\boldsymbol{M}}$ is a diagonal matrix with elements m_i on its diagonal and the weight ratio β is a scalar which can be used to scale the relative size of the uncertainty weighting filters without affecting the overall size of the specified system uncertainty (a similar weight ratio is used for the

transformation in equations (4.2)). For these uncertainty weighting filters, an additive error of $\delta_i m_i$ in each singular value of the system dynamics is obtained by the following normalised uncertainty matrix

$$\boldsymbol{\Delta} = \boldsymbol{U}\bar{\boldsymbol{\Delta}}\boldsymbol{T}^T, \tag{4.66}$$

where $\bar{\boldsymbol{\Delta}}$ is a diagonal matrix with elements δ_i on its diagonal. Since $\boldsymbol{U}$ and $\boldsymbol{T}$ are orthogonal and $|\delta_i| < 1$, the following relation holds

$$\|\boldsymbol{\Delta}\|_{i2} = \left\|\bar{\boldsymbol{\Delta}}\right\|_{i2} = \max_i \delta_i < 1. \tag{4.67}$$

The spectral norm of the normalised uncertainty matrix is thus less than 1 for the chosen set of weighting filters.

The robustness filter $\boldsymbol{R}$ is chosen such that the components of the summed error in the column space of $\boldsymbol{U}$ are filtered independently, i.e.,

$$\boldsymbol{R} = \boldsymbol{U}\bar{\boldsymbol{R}}\boldsymbol{U}^T, \tag{4.68}$$

where $\bar{\boldsymbol{R}}$ is a diagonal matrix with elements r_i on its diagonal. No robustness filter is applied if $\boldsymbol{R} = \boldsymbol{I}$, which implies $r_i = 1$ by the orthogonality of $\boldsymbol{U}$.

With the aforementioned choice of the weighting filters and the robustness filter, the expression for the system equations and the objective function can be decoupled in terms of the components of the following transformed vectors

$$\bar{\boldsymbol{z}}^k = \boldsymbol{U}^T\boldsymbol{z}^k, \qquad \bar{\boldsymbol{w}}^k = \boldsymbol{U}^T\boldsymbol{w}^k, \qquad \bar{\boldsymbol{p}}^k = \boldsymbol{T}^T\boldsymbol{p}^k, \qquad \bar{\boldsymbol{q}}^k = \boldsymbol{U}^T\boldsymbol{q}^k. \tag{4.69}$$

The first two rows of the expression for the standard plant in equation (4.15) can be expressed in terms of the components of these transformed vectors as

$$\bar{w}_i^{k+1} = r_i\bar{w}_i^k + s_i\bar{u}_i^k + \beta\bar{q}_i^k \tag{4.70a}$$

$$\bar{p}_i^k = m_i\bar{u}_i^k, \tag{4.70b}$$

where the subscript i denotes the i^{th} component of the transformed lifted vector. The objective function for RILC, which is defined in equation (4.23), can be expressed in terms of the components of these transformed vectors as

$$J^{(s)^k} = \bar{\boldsymbol{w}}^{k+1T}\bar{\boldsymbol{w}}^{k+1} + \bar{\boldsymbol{p}}^{kT}\bar{\boldsymbol{p}}^k - \gamma^2\bar{\boldsymbol{w}}^{kT}\bar{\boldsymbol{w}}^k - \gamma^2\bar{\boldsymbol{q}}^{kT}\bar{\boldsymbol{q}}^k = \sum_i \left(\left(\bar{w}_i^{k+1}\right)^2 + \left(\bar{p}_i^k\right)^2 - \gamma^2\left(\bar{w}_i^k\right)^2 - \gamma^2\left(\bar{q}_i^k\right)^2\right), \tag{4.71}$$

where the orthogonality of $\boldsymbol{U}$ and $\boldsymbol{T}$ is exploited.

According to the design objective for RILC, formulated in equation (4.25), the optimal compensable input $\boldsymbol{u}^k$ should minimise the objective function for a maximising effect of $\boldsymbol{q}^k$. In subsection (4.3.1) it is concluded that conditions (4.39) are sufficient for the existence of a minimising $\boldsymbol{u}^k$ and a maximising

$\boldsymbol{q}^k$. Substitution of the uncertainty weighting matrices from equation (4.65) in those conditions yields the following set of decoupled inequalities

$$s_i^2 + \frac{m_i^2}{\beta^2} > 0, \tag{4.72a}$$

$$\beta^2 - \gamma^2 < 0. \tag{4.72b}$$

These conditions are satisfied by selecting β such that $\beta^2 < \gamma^2$ and selecting $m_i \neq 0$ if $s_i = 0$. If conditions (4.72) are satisfied, then the minimising $\boldsymbol{u}^k$ and the maximising $\boldsymbol{q}^k$ are expressed by equations (4.34) and (4.30) respectively. Substitution of the uncertainty weighting matrices from equations (4.65) and the robustness filter from equation (4.68) and using the transformations from equation (4.69) yields the following decoupled equations

$$\bar{q}_i^k = \tfrac{\beta}{\gamma^2 - \beta^2} \left(r_i \bar{w}_i^k + s_i \bar{u}_i^k \right), \tag{4.73a}$$

$$\bar{u}_i^k = -l_i \bar{w}_i^k, \tag{4.73b}$$

where

$$l_i = \frac{\gamma^2 \beta^2 s_i r_i}{\gamma^2 \beta^2 s_i^2 + (\gamma^2 - \beta^2)\, m_i^2}. \tag{4.74}$$

Thus, the algorithm for computing the optimal compensable input is reduced to a set of scalar equations in terms of the components of the transformed vectors. Note that the structure of $\boldsymbol{\Delta}$ in equation (4.66) is not used for the derivation of the optimal inputs in equations (4.73). These inputs are thus also optimal for any $\|\boldsymbol{\Delta}\|_{i2} < 1$ that does not comply with the structure in equation (4.66). Using system equations (4.70) and the expressions for the optimal inputs from equation (4.73), the optimal objective function from equation (4.71) can be expressed in terms of $\bar{w}_i^k$ only as

$$J^{(s)^k} = \sum_i J^{(s)^k}_i \left(\bar{w}_i^k \right)^2, \tag{4.75a}$$

where,

$$J^{(s)^k}_i = -\gamma^2 \frac{\gamma^2 \beta^2 s_i^2 + \left(\gamma^2 - \beta^2 - r_i^2\right) m_i^2}{\gamma^2 \beta^2 s_i^2 + (\gamma^2 - \beta^2)\, m_i^2} = \\ - \gamma^2 \left(1 - \frac{r_i^2 m_i^2}{\gamma^2 \beta^2 s_i^2 + (\gamma^2 - \beta^2)\, m_i^2} \right). \tag{4.75b}$$

Each component of the transformed compensable summed error thus contributes independently to the objective function. If $J^{(s)^k}_i < 0$ for all i then the optimal objective function is negative definite with respect to all components of the compensable summed error, which means that the SCRC is satisfied (see section 4.2).

No uncertainty

Hereafter, a non-singular system with no uncertainty in the size of the singular values is considered. Substitution of $s_i \neq 0$ and $m_i = 0$ in equation (4.75b) yields

$$J^{(s)}{}_i^k = -\gamma^2 \tag{4.76}$$

for all i, which implies that the SCRC is satisfied and robust convergence is guaranteed. The proposed RILC algorithm thus results in robust convergence of the summed error with a maximum convergence rate γ if the dynamics of the controlled system are non-singular and there is no model uncertainty. The convergence for $m_i \neq 0$ is analysed hereafter.

Without a robustness filter

According to the analysis in subsection 4.4.1, the error converges monotonically to zero if the SCRC is satisfied and no robustness filter is applied, i.e., $\boldsymbol{R} = \boldsymbol{I}$. Substitution of $r_i = 1$ in equation (4.75b) gives

$$J^{(s)}{}_i^k = -\gamma^2 \left(1 - \frac{m_i^2}{\gamma^2\beta^2 s_i^2 + (\gamma^2 - \beta^2)\, m_i^2}\right) \tag{4.77}$$

The value of $J^{(s)}{}_i^k$ is negative if

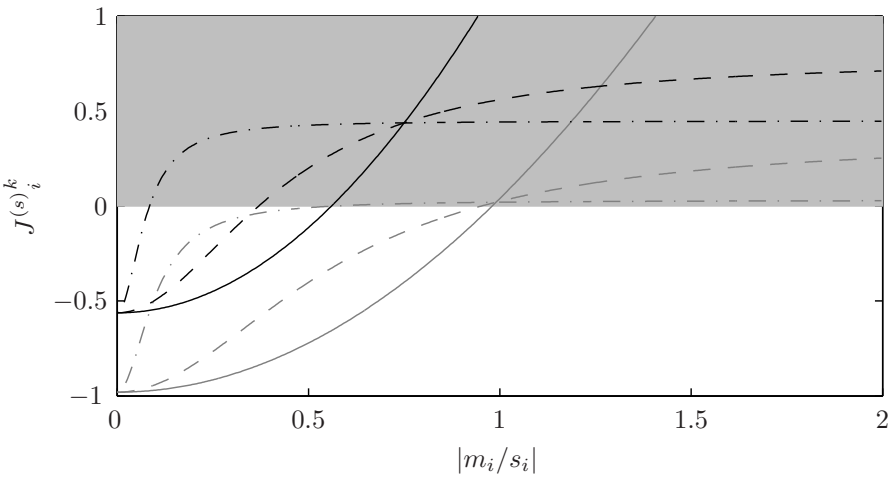
$$m_i^2 < \frac{\beta^2}{1 - \gamma^2 + \beta^2}\gamma^2 s_i^2, \tag{4.78}$$

Since $\gamma^2\beta^2/(1 - \gamma^2 + \beta^2) < 1$ for any $\gamma^2 < 1$ and $\beta^2 < \gamma^2$ (condition (4.72b)), $J^{(s)}{}_i^k < 0$ can only be satisfied if $m_i^2 < s_i^2$. This implies that the SCRC is only satisfied for $r_i = 1$ if the uncertainty in each singular value of the model is smaller than the size of the singular value. Moreover, the smaller the maximum convergence ratio, which is specified by γ, the smaller the ratio $|m_i/s_i|$ for which the SCRC is satisfied if $r_i = 1$. These conclusions are confirmed by figure 4.3, which shows $J^{(s)}{}_i^k$ as a function of $|m_i/s_i|$ for several values of the maximum convergence ratio γ and the weight ratio β. The figure also shows that the smaller β, the smaller the ratio $|m_i/s_i|$ for which $J^{(s)}{}_i^k < 0$.

With a robustness filter

Above it is shown that the SCRC can only be satisfied for $r_i = 1$ if $|m_i/s_i| < 1$. Thus, robust convergence is not guaranteed if $|m_i/s_i| \geq 1$ and $r_i = 1$. However, the SCRC can be satisfied for $|m_i/s_i| \geq 1$ by taking $r_i \neq 1$. The expression for $J^{(s)}{}_i^k$ for $r_i \neq 1$ is given in equation (4.75b). From this equation it can be derived that $J^{(s)}{}_i^k$ is negative definite if

$$r_i^2 m_i^2 < \left(\gamma^2 - \beta^2\right) m_i^2 + \beta^2\gamma^2 s_i^2 \tag{4.79}$$



weight ratio: — $\beta = 0.99\gamma$, – – $\beta = 0.50\gamma$, ·–·– $\beta = 0.10\gamma$,
maximum convergence ratio: — $\gamma = 0.75$, — $\gamma = 0.99$,
the SCRC is not satisfied in the shaded area

Figure 4.3: The objective function component $J^{(s)}{}_i^k$ as a function of the additive uncertainty ($|m_i/s_i|$) for RILC without a robustness filter ($r_i = 1$)

This inequality can be satisfied for any m_i by taking $r_i^2 < \gamma^2 - \beta^2$. Thus the SCRC, which implies monotonic convergence of the summed error with a maximum convergence ratio γ, can be satisfied for any size of the model uncertainty if r_i^2 is taken sufficiently small.

Figure 4.4 shows $J^{(s)}{}_i^k$ as a function of $|m_i/s_i|$ for several values of the maximum convergence ratio γ, the weight ratio β and the robustness filter component r_i. The figure shows that the range of $|m_i/s_i|$ for which $J^{(s)}{}_i^k < 0$ is enlarged by taking a small value of r_i^2 or a large value of γ^2. The effect of the weight ratio β on the range of $|m_i/s_i|$ for which $J^{(s)}{}_i^k < 0$ depends on the value of γ and r_i.

4.4.3 Parameter selection

In the preceding subsections the convergence properties of the proposed RILC algorithm are analysed. In particular, the effects of the maximum convergence ratio, the robustness filter and the uncertainty weighting filters on the convergence of the summed error and the final error are analysed. In this subsection several guidelines for the selection of these parameters are formulated, based on the results of the convergence analysis. The guidelines are partly based on the

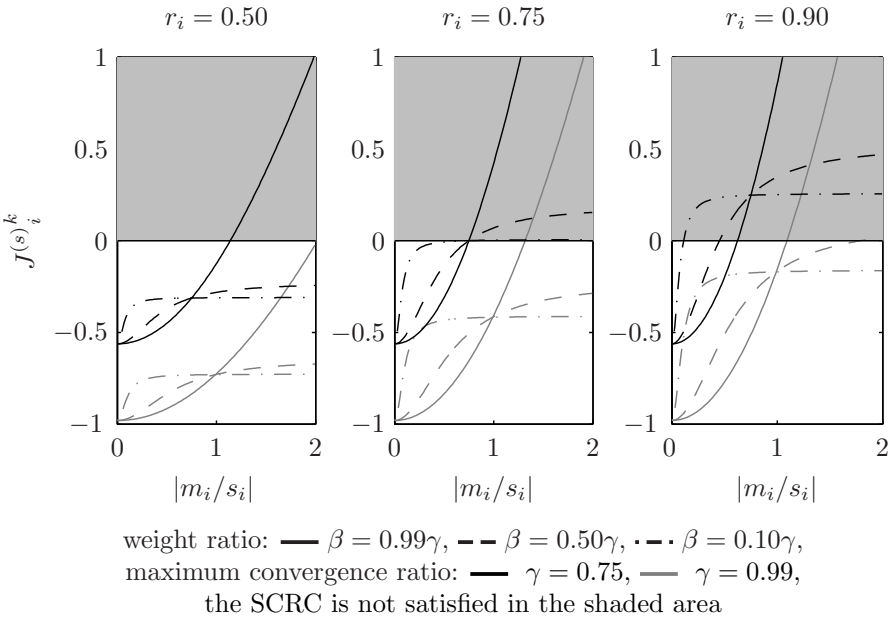


Figure 4.4: The objective function component $J^{(s)}{}_i^k$ as a function of the additive uncertainty (m_i/s_i) for RILC with a robustness filter

analysis in subsection 4.4.2, in which it is assumed that the system is square and the only model uncertainty is the size of the singular values of the system matrix.

Selection of the maximum convergence ratio

The convergence analysis in section 4.2 shows that the convergence ratio of the summed error is γ at most if the SCRC is satisfied for that value of γ. The value of γ can thus be used to specify the desired maximum convergence ratio.

The objective function depends on the selected value of γ according to equation (4.27). The optimal objective function should be negative definite for the specified model uncertainty to satisfy the SCRC and thus the allowable model uncertainty for which the SCRC is satisfied depends on the value of γ. In subsection 4.4.2 it is shown by equation (4.78), figures 4.3 and 4.4 that a small value of γ decreases the model uncertainty for which the SCRC is satisfied.

The selection of γ can thus be used to set the maximum convergence rate, but it affects the allowable model uncertainty. A small value of γ should be taken to decrease the maximum convergence ratio, but from subsection 4.4.2 it

is concluded a large value of γ should be taken to increase the allowable model error.

Selection of the robustness filter

Preferably, no robustness filter should be used, because then the error converges to zero robustly and monotonically according to equation (4.61) if the SCRC is satisfied. In subsection 4.4.2 it is shown that the SCRC is satisfied if the system is square, modelled perfectly and the model uncertainty is zero. However, the SCRC is cannot be satisfied for a large model error if no robustness filter is used. In subsection 4.4.2 it is shown that the SCRC cannot be satisfied without a robustness filter if the uncertainty in the size of a singular value of the system is larger than the size of that singular value. The convergence condition for NILC without a robustness filter (equation (3.60)) can be satisfied if the signs of the singular values of the estimated system model and the real system are equal. Hence, the SCRC for RILC without a robustness filter is more strict than the convergence condition for NILC without a robustness filter.

In subsection 4.4.1 it is shown that the SCRC can only be satisfied for large model uncertainty by using a (non-unity) robustness filter, though the application of a robustness filter can also result in a non-zero final error according to equation (4.64). The SCRC can even be satisfied for any size of the model uncertainty if the robustness filter is set to zero, though in this case the error is not reduced at all. In subsection 4.4.2 it is shown that the SCRC can be satisfied for a large uncertainty in the singular value by taking the corresponding gain of the robustness filter smaller than unity (equation (4.79)).

RILC thus results in zero final error if no robustness filter is used and the SCRC is satisfied. However, the robustness filter can be needed to increase the robustness to model uncertainty, although this could result in a nonzero final error. The robustness filter should thus be zero for the components of the error corresponding to a large model uncertainty to obtain convergence, but close to unity for the other components of the error to reduce these error components to zero. In practice, the dynamics of mechanical systems are accurately known at low frequencies but uncertain at high frequencies. The robustness filter should then be a low-pass filter that is unity at low-frequencies and close to zero at high frequencies. This behaviour can be realised by implementing the robustness filter as a high-order zero-phase low-pass filter. The high-order results in a small frequency band in which the low-pass filter rolls off from unity to zero. This behaviour cannot be realised by a stable and causal low-pass filter as the roll-off introduces phase-lag at low-frequencies, which makes the filter unequal to unity at those frequencies.

Selection of the uncertainty weighting filters

The uncertainty weighting filters $\boldsymbol{M}$ and $\boldsymbol{N}$ should be selected such the uncertainty in the system dynamics is expressed by $\boldsymbol{G} = \hat{\boldsymbol{G}} + \boldsymbol{N}\boldsymbol{\Delta}\boldsymbol{M}$ for some

$\|\boldsymbol{\Delta}\|_{i2} < 1$. The selection of the uncertainty weighting filters is not unique, any combination is appropriate as long as the uncertainty is specified appropriately. Uncertainty weighting filters that appropriately specify the uncertainty can be modified as in equation (4.2) for some real value of the weight ratio β without affecting the specified overall uncertainty.

The uncertainty weighting filters should not only specify the uncertainty appropriately, but they should also be selected such that the sufficient conditions (4.39) for the existence of an optimal learning filter are satisfied. These conditions are satisfied if the pre-weighting filter $\boldsymbol{M}$ is nonsingular and the spectral norm of the post-weighting filter $\boldsymbol{N}$ is sufficiently small. The spectral norm of the post-weighting filter can be decreased by modifying the set of weighting matrices as in equation (4.2) using a sufficiently small value of the weight ratio β.

The uncertainty weighting filters should thus appropriately specify the model uncertainty and the conditions for the existence of an optimal learning filter should be satisfied. Still some freedom exist in the selection of these filters. This freedom can be used to satisfy the SCRC for a small value of the maximum convergence ratio γ.

In subsection 4.4.2 the effect of the selection of the uncertainty weighting filters on the convergence properties of RILC is illustrated for square systems of which the model uncertainty is related to the singular values. The post-weighting filter $\boldsymbol{N}$ is taken equal to the (scalar) weight ratio β and pre-weighting filter $\boldsymbol{M}$ is defined such that an additive uncertainty m_i for each singular value s_i is specified appropriately. The sufficient conditions (4.39) for the existence of the optimal learning filter are satisfied if $\beta^2 < \gamma^2$ and if $m_i \neq 0$ for $s_i = 0$. The convergence analysis shows that the smaller the value of m_i, the smaller the maximum convergence ratio γ for which the SCRC is satisfied. If no robustness filter is applied, then selecting β close γ increases the value of m_i for which the SCRC is satisfied, though the SCRC can only be satisfied if $|m_i/s_i| < 1$.

Guidelines for the selection of the uncertainty weighting filters are derived from the decoupled analysis in subsection 4.4.2. The spectral norm of the post-filter $\boldsymbol{N}$ should be taken smaller than γ to satisfy the condition for the existence of an optimal learning filter, but only slightly smaller than γ to maximise allowable uncertainty if $\boldsymbol{R} = \boldsymbol{I}$. The post-weighting filter $\boldsymbol{M}$ should be taken as small as possible to maximise the convergence rate that can be realised, though it should be taken sufficiently large to specify the dynamic model uncertainty appropriately. A reasonable choice for the weighting filters, derived from the previous considerations, is a static post-weighting filter $\boldsymbol{N}$ with a gain slightly below γ, while the pre-weighting filter $\boldsymbol{M}$ is a dynamic filter that is just large enough to specify the dynamic uncertainty appropriately. The proposed selection of the weighting filters differs from the static pre-weighting filter and the dynamic post-weighting filter used by Van de Wijdeven and Bosgra (2007a). As noted in their work, a dynamic pre-weighting filter and a static post-weighting filter "may result in less conservative solutions", which agrees with the previous

analysis. The RILC algorithms proposed in subsections 4.3.1 and 4.3.2 can be implemented with any combination of static and dynamic weighting filters.

Discussion

In the previous analysis it is concluded that the final error depends on the selection of the robustness filter and the convergence rate is bounded by the selected maximum convergence ratio and also depends on the selection of the uncertainty weighting filters. The selection of the maximum convergence ratio, the robustness filter and the uncertainty weighting filters all determine the allowable model uncertainty. The best selection of these parameters for a certain application thus depends on the model uncertainty, the desired convergence rate and the allowable final error. The choice of these filters for the experiments on the Stäubli RX90 robot is discussed in subsection 6.2.2.

The advantage of the proposed RILC algorithm over the NILC algorithm from chapter 3 is that a maximum convergence rate can be specified explicitly and the learning filter is optimised to realise this convergence ratio for the worst case effect of the specified model uncertainty. Moreover, a condition is derived that can be used to check if the specified convergence rate can be realised for the specified model uncertainty and the selected robustness filter. A disadvantage of the proposed RILC algorithm is that the used convergence condition is only a sufficient condition. This conservativeness could lead to a convergence ratio or a final error that are larger than those realised by, e.g., the NILC algorithm from chapter 3.

Chapter 5

The experimental setup

This chapter describes the experimental setup that is used for testing the ILC algorithms described in chapters 3 and 4. The results from the experiments demonstrate the performance of the developed ILC algorithms in relation to the requirements formulated in section 1.2. The experimental results are presented in chapter 6.

The experimental setup, which consists of the Stäubli RX90 robot and auxiliary equipment, is described in section 5.1. The reduction of the tracking error of this robot by ILC is tested for two trajectories, which are described in section 5.2. The dynamics of the robot system along those trajectories are modelled as described in section 5.3. These models are used for the implementation of the ILC algorithms.

5.1 System description

The experimental setup is depicted in figure 1.2. A schematic overview of the components of the experimental setup and their interconnections is shown in figure 5.1. The Stäubli RX90 robot manipulator is used for the experiments. The motion of the joints of this manipulator is controlled in real-time by the industrial CS8 controller. This controller supplies the currents for the motors in the manipulator's joints such that the actual joint angles approximately trace the specified setpoints. The tracking error of the robot tip is measured by an optical seam-tracking sensor, which is integrated with the welding head attached to the robot. The measurements of this sensor are triggered by the CS8 controller and the sensor returns the measured tip tracking errors to the CS8 controller. The controller synchronises these measurements with the robot motion. ILC is used to reduce the measured tip tracking errors by iteratively updating the setpoints for the position of the robot tip. The components of the experimental setup are described in more detail in the following subsections.

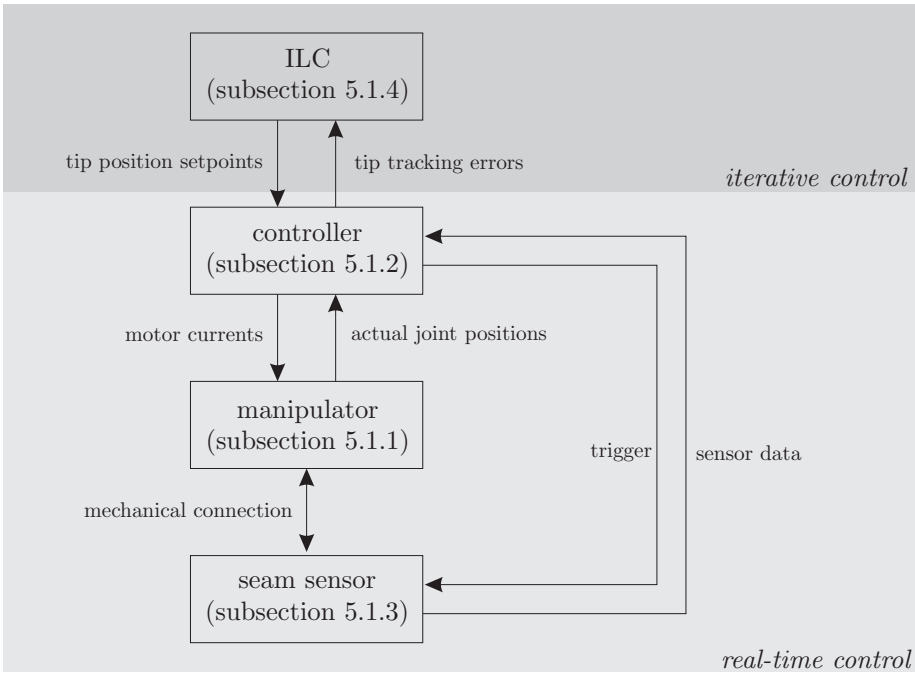
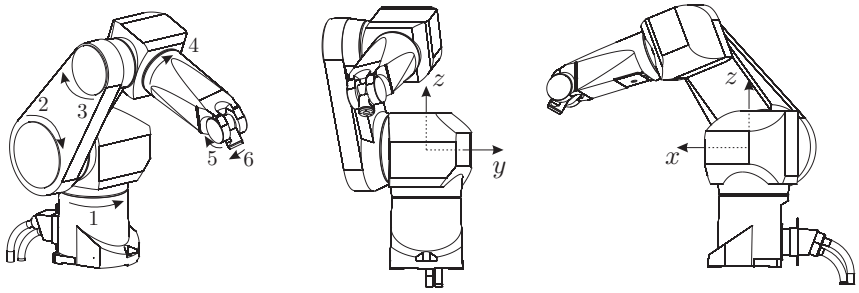


Figure 5.1: The components of the experimental setup and their interconnections

5.1.1 Manipulator

The Stäubli RX90 robot manipulator (Stäubli, 2003a) is depicted in figures 1.2 and 5.2. The manipulator consists of stiff and lightweight aluminium links that are interconnected by six revolute joints. The joints are numbered as shown in figure 5.2(a). Joints 1 to 4 are so-called Stäubli Combined Joints (JCS), which is an assembly containing the joint's transmission and bearing. Joints 5 and 6 are driven by two motors located in link 4 via a worm and wheel gear for each joint and a bevel gear for joint 6. The construction is such that the motion of joint 5 is coupled to the motion of one of these motors and the motion of joint 6 is coupled to the motion of both motors. All joints are actuated by three-phase servo motors, which are powered by the CS8 controller. Resolvers on the motor axes measure the angle and angular velocity. These measurements are returned to the CS8 controller (see figure 5.1).

The six joints allow manipulation of the end-flange of the robot to any position and orientation within the robot's working range. The location of a tool mounted to this end-flange is expressed by its position and orientation in the global $\mathcal{O}_{xyz}$ coordinate system. The origin of this coordinate system is located



(a) The six axes (b) Front view (c) Side view

source: Stäubli Robotics

Figure 5.2: The Stäubli RX90 robot

property	value	unit
reach between joints 2 and 5	900	mm
repeatability at end-effector	± 0.02	mm
maximum velocity at end-effector	11	m/s
nominal load capacity	6	kg
maximum load capacity	12	kg

Table 5.1: Characteristic properties of the Stäubli RX90 robot (Stäubli, 2003a)

at the crossing of the axis through joints 1 and 2 and its orientation is as illustrated in figure 5.2.

Table 5.1 list some characteristic properties of the robot. The load capacity and the maximum tip velocity suffice for the application of the robot to laser welding. The small repeatability of the Stäubli RX90 robot is the result of the stiff design of the robot's joints and links and is unsurpassed by any industrial six-axis robots with comparable reach. The repeatability is the lower limit for the tracking error that can be realised by ILC and thus the small repeatability is beneficial for realising a small tracking error with ILC.

5.1.2 Controller

The motion of the Stäubli RX90 robot is controlled by the industrial CS8 controller (Stäubli, 2003b). The controller consists of four layers; the path generation module, the motion controllers, the servo amplifiers and the safety module. The safety module takes care of the communication with components for robot safety such as emergency stops and motor breaks. The other layers and their interconnections are schematically shown in figure 5.3. The servo amplifiers

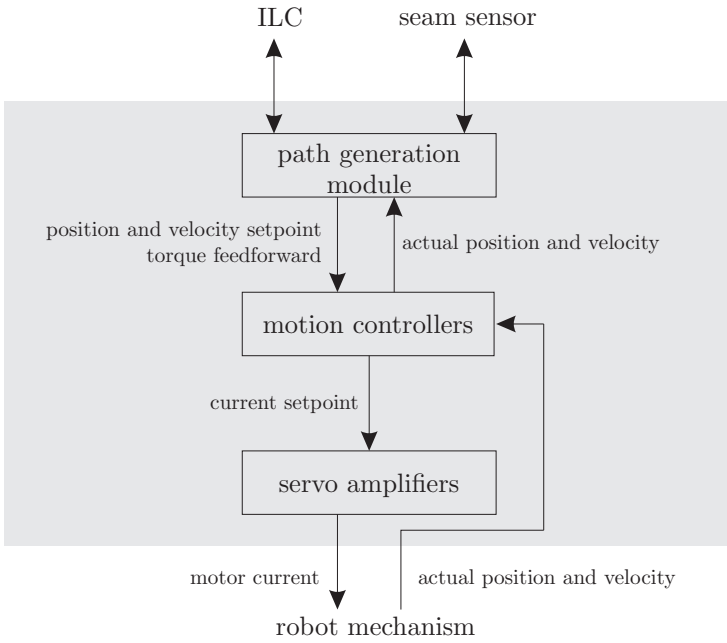


Figure 5.3: The layout of the CS8 controller

power the servo motors that actuate the joints of the robot. The torques exerted by these servo motors are linearly related to the motor currents supplied by the servo amplifiers. The servo amplifiers control the motor currents to the current setpoints specified by the motion controllers. The motion controllers compute these current setpoints to make the actual joint angle and the angular velocity trace the setpoints specified by the path generation module. Besides the setpoints for the joint angle and the angular velocity, the path-generation module can specify a torque feedforward. Furthermore, the path generation module takes care of the communication with the seam-tracking sensor and the ILC algorithm (see also figure 5.1).

The CS8 controller contains an independent motion controller for each of the six motors. Each motion controller consists of an interpolator, a feedback controller and a feedforward controller. A schematic overview of the motion controllers is given in figure 5.4. The interpolator computes position, velocity and acceleration setpoints at a rate of 2 kHz from the position and velocity setpoints that are specified by the path generation module at a rate of 250 Hz. The position setpoints from the interpolator and the actual position and velocity measured by the resolvers on the motor axes are the inputs for the feedback controller. The feedback controller consists of a cascaded position and velocity

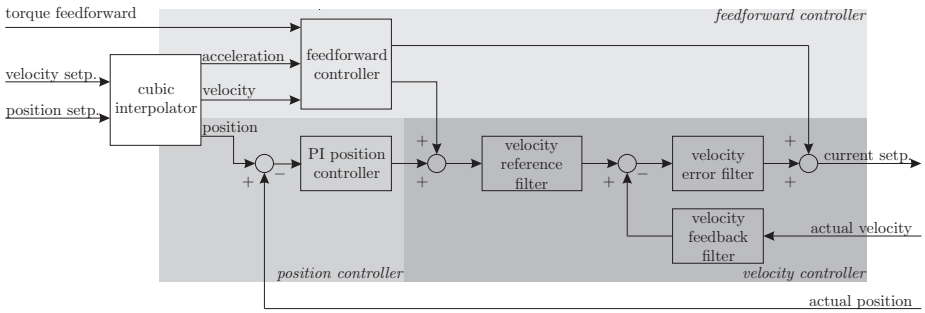


Figure 5.4: Block diagram of the motion controllers of the CS8

loop, running at 2 kHz and 4 kHz respectively. The feedforward controller computes a current feedforward and a velocity feedforward using the velocity and acceleration setpoints from the interpolator and the torque feedforward that is specified by the path generation module. The sum of the output of the feedback controller and the current feedforward is the current setpoint for the servo amplifier. The motion controllers are implemented on digital signal processors (Kollmorgan, 2001). These digital signal processors, the hardware of the safety module and the hardware of the servo amplifiers are integrated in a single hardware module for each drive.

The path generation module specifies the position setpoints, the velocity setpoints and the torque feedforwards for the motion controllers (see figure 5.3). These setpoints are transferred to the motion controllers via a dedicated optical bus at a rate of 250 Hz. Each motion controller returns the measured position and velocity to the path generation module via the same bus. The software of the path generation module runs on an industrial PC with the real-time VxWorks operating system. Conventionally, the VAL3 software provided by Stäubli takes care of the path generation. This software sets the torque feedforward to zero and computes the position and velocity setpoints from the desired velocity and points along the desired trajectory specified by the user. In this work the path generation software developed by De Graaf (2007) is used. This software employs the Real Time Robot Controller Abstraction Layer (RTRCAL, Pertin and Bonnet-des-Tuves (2004)), a C++ Application Programming Interface for communication with the robot hardware. The software of De Graaf (2007) is used instead of the VAL3 software, because it gives direct access to the setpoints for the robot motion and it synchronises the measurements of the seam-tracking sensor (see subsection 5.1.3) with the measured joint motion. Both features are employed for the implementation of ILC in this work. The ILC algorithm uses the synchronised measurements of the tip tracking error to update the reference position for the tip at setpoint level (see figure 5.1).

In the first iteration of ILC the setpoints for the position and velocity of the

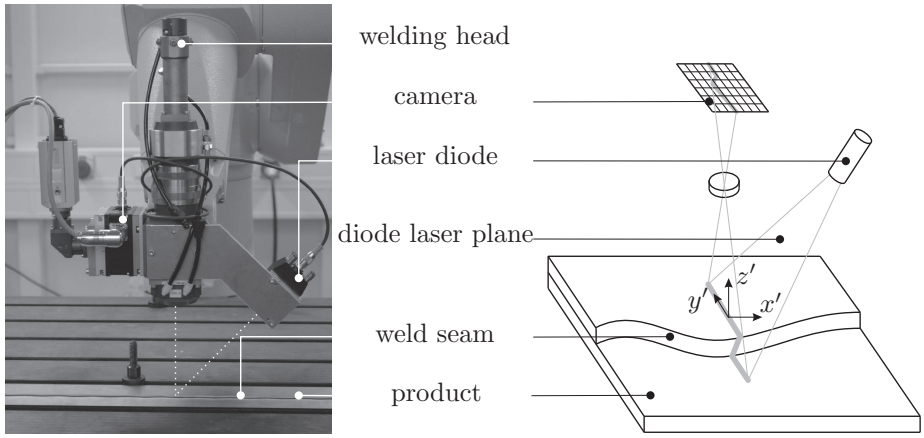
robot joints, which are the input for the joint controllers, are computed from the geometry of the seam and the desired velocity profile (see section 5.2) using a kinematic model of the Stäubli RX90 robot (Corke, 1996). These setpoints are transferred to the motion controller by the path-generation module. The setpoints for the first iteration are referred to as the *nominal* setpoints hereafter. After the first iteration, the ILC algorithm specifies an update for the tip position at setpoint level. The path generation module converts this update of the setpoints for the tip position to an update of the setpoints for the joint angle using the inverse of the local kinematic Jacobian (Corke, 1996). This Jacobian relates infinitesimal motions of the joints to infinitesimal motions of the robot tip. The update of the setpoints for the angular joint velocity are computed from the update of the setpoints for the joint angle using a central difference scheme.

The communication of the setpoints from the path-generator module to the motion controllers and the measured joint motion from the motion controllers to the path generator introduces two delays of 4 ms. In addition, the interpolator in the motion controllers introduces a delay of 4 ms. Altogether, the delay between the setpoints for the joint motion and the measured joint motion is three samples. The path-generation module synchronises the sensor measurements with the measurements of the joint motion, which means that the delay between the setpoints for the tip position and the motion measured by the sensor is also three samples.

5.1.3 Welding head with integrated seam-tracking sensor

The Stäubli RX90 robot carries a welding head with integrated seam-tracking sensor. The welding head is used to focus a high-power laser beam to an intensity that is sufficient for laser welding. Conventionally, the laser beam is transported to the welding head via a glass-fiber, though for the experiments in this work the delicate fiber is not connected. A Trumph welding head with a focal length of 150 mm is used. The welding head is equipped with an integrated seam-tracking sensor (Falldorf, 2002) that measures the location of the weld seam with respect to the welding head. The Close-To-Focus (CTF) sensor is used, which is able to measure the location of the seam close to the focal point of the welding head. A picture and a schematic drawing of the CTF sensor are shown in figure 5.5. The sensor system consists of a laser diode, a CMOS camera and an industrial PC. The laser diode projects a line on the product surface, perpendicular to the weld seam. The camera records the light that is reflected by the product at a known angle with respect to the laser diode. The recorded data are sent to the industrial PC that computes the location of the seam from features in the camera image.

The sensor measurements are triggered by the robot controller and the sensor's PC sends the measured seam location back to the robot controller (see figure 5.1). The robot controller triggers the measurements at the same rate



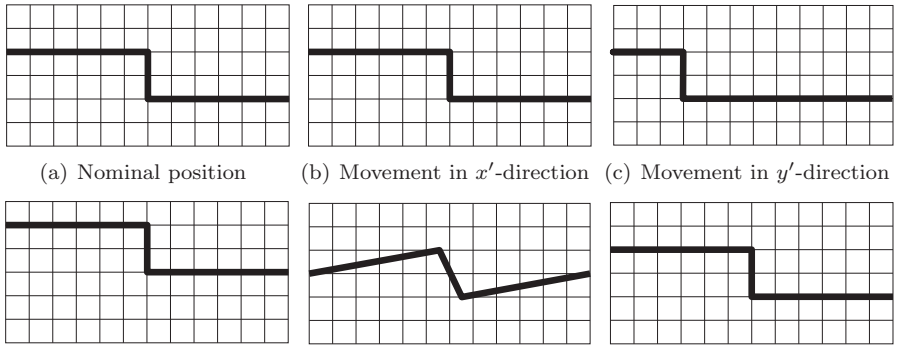
source: Falldorf Sensor

Figure 5.5: The Falldorf close-to-focus sensor

as the setpoints are supplied to the motion controller and compensates for the delay between triggering and reception of the measured seam location. This way, the sensor measurements are synchronised with the measured motion of the robot's joints. This synchronisation procedure is developed and described in more detail by De Graaf (2007).

The location of the weld seam is measured with respect to the local $\mathcal{O}_{x'y'z'}$ coordinate system of the sensor. This orthogonal coordinate system is oriented with respect to the sensor as illustrated in figure 5.5; the z'-direction coincides with focal line of the camera, the y'-direction is the perpendicular direction in the diode laser plane and the x'-direction is perpendicular to the other two axes. The focal line of the sensor's camera coincides with the focal line of the high-power laser. In this work the location of the weld seam is not measured during welding and therefore the location of the seam can be measured at the (virtual) focal point of the high-power laser beam. The origin of the sensor's coordinate system is put at this focal point such that the measured location of the seam equals the tracking error. The location of the focal point is calibrated with respect to the end-flange of the robot according to the procedure described by De Graaf (2007).

The x'-direction of the sensor should be approximately parallel to the local direction of the weld seam to obtain useful measurements. The camera detects a projection of the intersection of the weld seam with the diode laser plan. Figure 5.6 schematically shows the camera image of the CTF sensor and its change as a result of a movement of the welding head in the different directions above a straight overlap weld seam. Clearly, the y'- and z'-position of the point of intersection of the seam and the diode laser plane can be extracted from the



(a) Nominal position (b) Movement in x'-direction (c) Movement in y'-direction

(d) Movement in z'-direction (e) Rotation around x'-axis (f) Rotation around y' or z'-axis

Figure 5.6: The camera image of the CTF sensor above a straight overlap weld seam and the change of the image as a result of a movement of the welding head

camera image. The x'-position of this intersection is coupled to the z'-position via the angle of the laser plane around y'-axis. A movement of the welding head in the x'-direction does not result in a change of the camera image and thus the x'-position along the weld-seam cannot be measured with the sensor. A rotation of the welding head around the x'-axis changes the camera image and thus the angle around the x'-axis can be extracted from the image. The angle around the y'- and z'-axes cannot be detected. The resolution of the sensor in the linear directions is determined by the optical components in the sensor, the angle of the laser diode and the pixel-size of the camera. The resolution is 15 μm in the y'-direction and 26 μm in the z'-direction.

As a consequence of the small focal spot size and the small focal depth of the high-power laser beam, the tracking error of the laser focus with respect to the weld seam should be smaller than 0.1 mm in the y'- and z'-direction to obtain defect free welds (Duley, 1998; Olde Benneker and Gales, 2007; Römer, 2002). Small errors in the welding speed and the orientation of the welding head do not significantly affect the weld quality. The focus of this work is thus on reducing the tracking error in the y'- and z'-directions. It is assumed that the focal point of the high-power laser beam should be placed on the seam. Since the origin of the sensor's coordinate system coincides with the location of the focal point, the measured location of the weld seam in the y'- and z'-directions equals the tracking error that should be reduced by ILC.

5.1.4 The implementation of the ILC algorithms

The ILC algorithms proposed in chapters 3 and 4 are used to reduce the tip tracking errors measured by the sensor by updating the setpoints for the position

of the robot's tip (see figure 5.1). The ILC algorithms are implemented as described in subsections 3.3.2 and 4.3.2 using the algorithms from appendix B. These algorithms are implemented in MATLAB on a contemporary PC with 1.25 GB RAM and a processor running at 1.6 GHz.

In most literature on the application of ILC to robots (see section 2.3) the ILC algorithm is used to update either the torque feedforward setpoints or the position setpoints. In this work the position setpoints are updated, because this has several advantages over updating the torque feedforward setpoints. Firstly, although a torque feedforward can be specified for the motion controllers of the CS8, this is not the case for all industrial controllers. Updating the position setpoints makes the proposed procedure more generally applicable. Secondly, the relation between an update of the position setpoints and the resulting change of the tracking error is approximately unity at low frequencies. This facilitates the modelling of the robot dynamics (see section 5.3) and the tuning of the weights in the objective function for NILC (see subsection 6.2.1). Thirdly, the feedback motion controller contains an integrating action. This integrating action counteracts a constant torque feedforward and thus a constant error cannot be compensated by a constant torque feedforward. Moreover, a large torque feedforward is needed to compensate for the low-frequency components of the tracking error due to the integrating action. These disadvantages could be overcome by switching off the integrator of the feedback controller, though this is not in line with the requirements formulated in section 1.2.

Laser welding requires the compensation of the tracking errors of the welding head in the y' and z' direction (see subsection 5.1.3). The tracking errors in those directions are compensated with ILC by updating the setpoints for the position of the welding head in the same directions. This update is converted to an update of the setpoints for the joint angles by the path generation module (see subsection 5.1.2). Expressing the tracking errors and the setpoint update in the same coordinate system facilitates the modelling of the robot dynamics (see section 5.3) and the tuning of weights in the objective function for NILC (see subsection 6.2.1).

The delay between the setpoints for the joint motion and the motion measured by the sensor is three samples of 4 ms (see subsection 5.1.2). As a result of the delay, the updated setpoints computed by the ILC algorithm cannot compensate for the tracking error in the first three samples of the iteration. This is resolved by commanding the robot to move to the updated initial position setpoint and wait for 0.5 s prior to each iteration. As a result, the robot is at rest in the first four samples of each iteration and the (constant) tracking error in those samples is compensated by the update of the first position setpoint. Another effect of the delay is that the last three position setpoints do not affect the measured tracking error in the iteration. The optimal update of the setpoints for these time-samples computed by the ILC algorithms is thus zero. This could result in a discontinuity in the position setpoints and an undesired movement of

the robot after the end of the iteration. This is resolved by holding the position setpoints at the same value after the third last sample of each iteration.

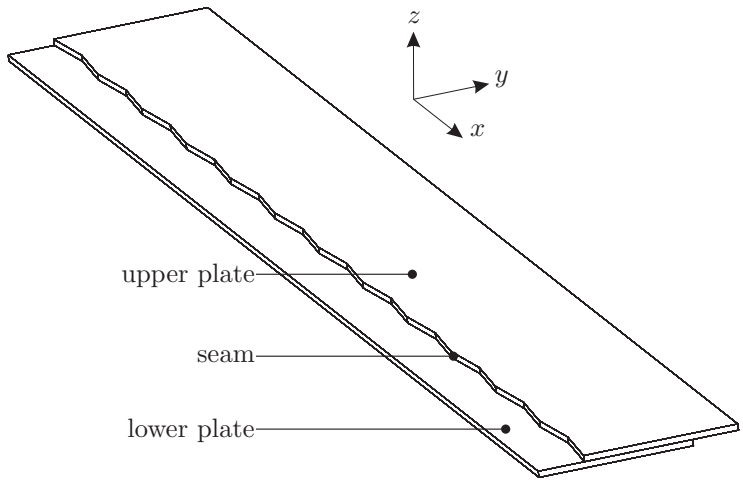
5.2 Trajectory definition

The performance of the proposed ILC algorithms is tested for two weld-seam geometries, which define trajectory A and B for the robot motion. Trajectory A is specially designed to show the performance that can be realised with the proposed ILC algorithms. Trajectory B is typical for laser welding tasks in industry.

5.2.1 Trajectory A

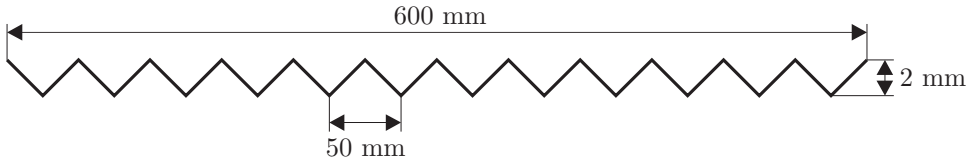
The weld seam that defines trajectory A is formed by the serrated edge of a metal strip that overlaps a second strip. These strips are depicted in figure 5.7 and the dimensions of the seam geometry are illustrated in figure 5.8.

The profile is placed in the xy-plane of the $\mathcal{O}_{xyz}$ coordinate system (see section 5.1.1) at $z = -368\ mm$ with its base-line oriented in the x-direction. The nominal setpoints for the robot motion are defined such that the focus point of the welding head moves along the base-line of the profile, starting at $(x, y, z) = (350, 0, -368)$ mm and ending at $(x, y, z) = (850, 0, -368)$ mm. The xy-components of these setpoints are shown in figure 5.9. The nominal setpoints for the orientation of the welding head are defined such that the sensor's $\mathcal{O}_{x'y'z'}$



(the arrows indicate the directions of $\mathcal{O}_{xyz}$, not the origin)

Figure 5.7: The weld seam defining trajectory A



(the horizontal and vertical dimensions are scaled differently)

Figure 5.8: The dimensions of the weld seam defining trajectory A

coordinate system is kept aligned with the global $\mathcal{O}_{xyz}$ coordinate system. The velocity profile, depicted in figure 5.10, is trapezoidal with a maximum velocity of 400 mm/s and an acceleration of 1600 mm/s^2. The nominal setpoints for the joint angles are computed from the nominal setpoints for the position and orientation of the welding head and the velocity profile using a kinematic model of the Stäubli RX90 robot (Corke, 1996). The nominal setpoints for the joint angles are depicted in figure 5.11. The robot configuration changes from almost fully retracted at the start to almost fully stretched at the end of the trajectory, which can be seen from the significant change of the angles of joints 2, 3 and 6 along the trajectory. Figure 5.12 shows pictures of the change of configuration of the Stäubli RX90 robot along trajectory A.

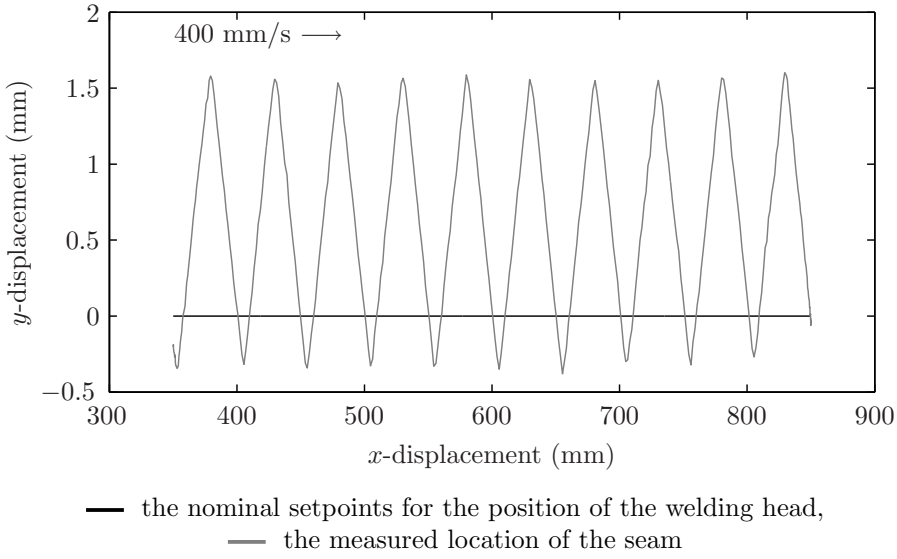


Figure 5.9: The location of trajectory A

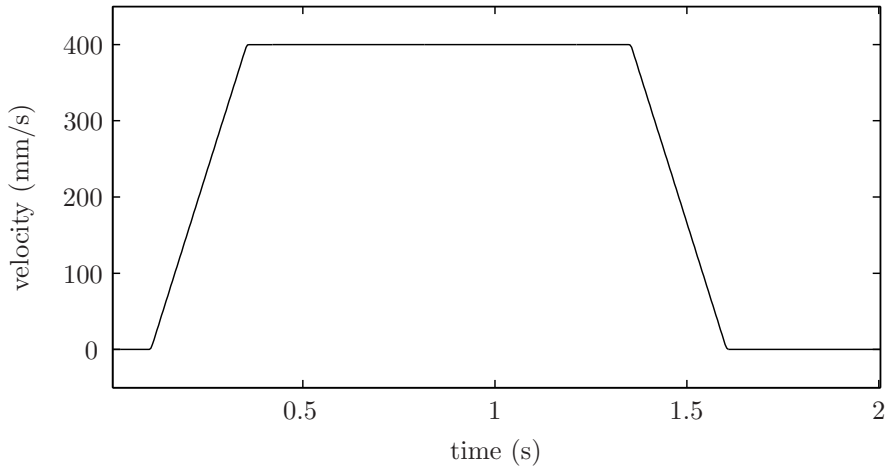


Figure 5.10: The velocity profile along trajectory A

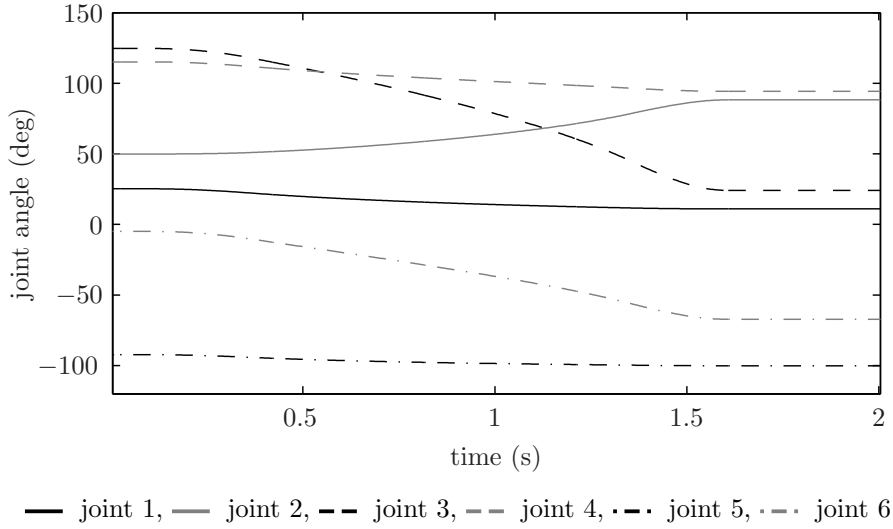
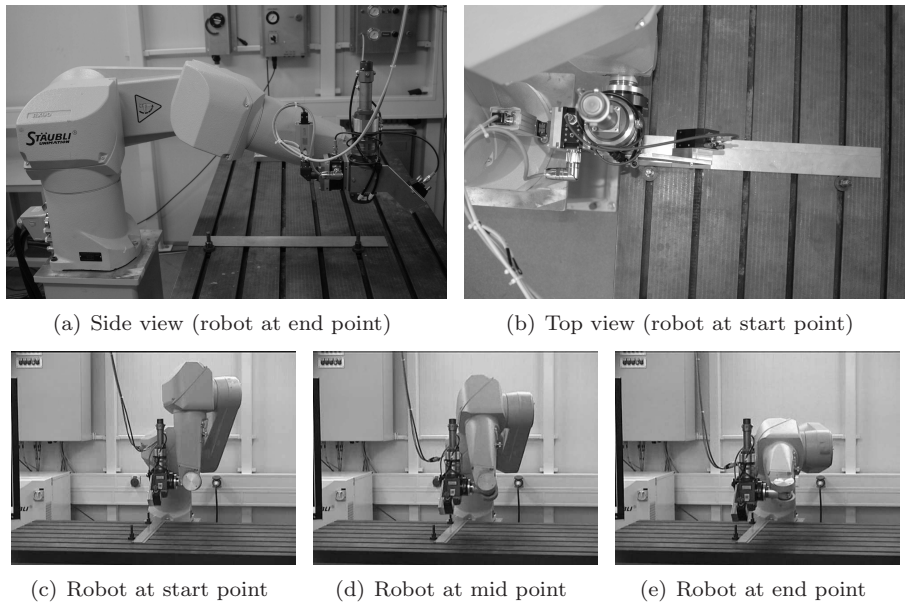


Figure 5.11: The nominal setpoints for the joint angles along trajectory A



(a) Side view (robot at end point) (b) Top view (robot at start point)

(c) Robot at start point (d) Robot at mid point (e) Robot at end point

Figure 5.12: The robot movement along trajectory A

The location of the seam with respect to the nominal trajectory of the welding head is measured with the sensor by commanding the robot to move along the trajectory at 50 mm/s. At this low speed geometrical effects dominate the tracking error. Figure 5.9 shows the measured location of the seam, which is obtained by adding the measured tracking error in the y'-direction to the nominal position setpoints for the welding head. The serrated profile of the seam is clearly visible in the figure. Next, the location of the seam is measured while the robot is commanded to move along the trajectory at nominal velocity. The measured tracking error at nominal speed, which is referred to as the nominal tracking error, should be reduced by the application of ILC. Figure 5.13 shows the nominal tracking error in the y' and z'-direction for the motions at 50 mm/s and 400 mm/s. In both cases the serrated profile of the seam is visible in the y'-direction of the tracking error and a trend is visible in the z'-direction of the tracking error. The trend is the result of either a misalignment of the metal strips or a deflection of the robot mechanism due to gravity forces while it stretches. The difference between the tracking error at 50 mm/s and 400 mm/s is the result of dynamic effects. The most notable difference is the error in the z'-direction at the end of the motion. Apparently the robot cannot keep the welding head at the commanded height during the (high) deceleration. Another difference is that the y'-component of the error for 400 mm/s lags the error for

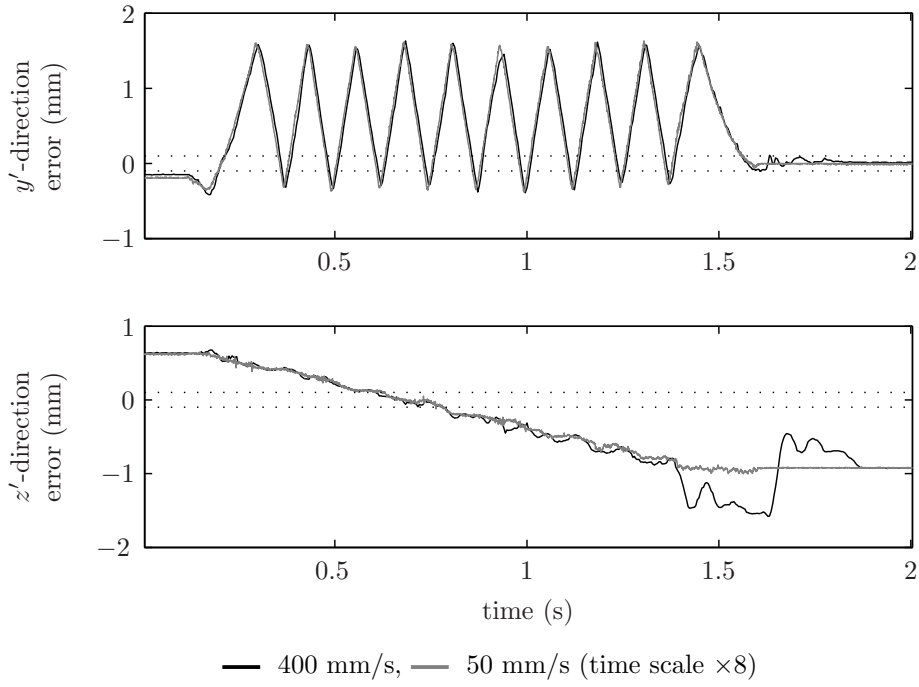


Figure 5.13: The nominal tracking error along trajectory A

50 mm/s. Probably the robot lags the commanded position in the x'-direction for a velocity of 400 mm/s. This tracking error is not compensated by the application of ILC as the tracking error in the x'-direction cannot be measured with the seam-tracking sensor.

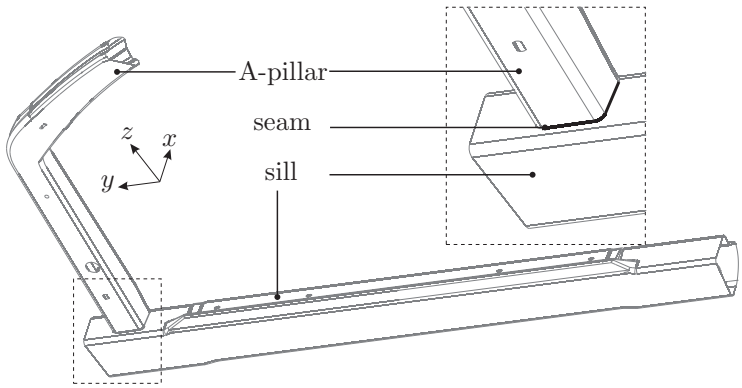
Realising high-accuracy motion along trajectory A is challenging for two reasons. Firstly, during the motion along this trajectory the configuration of the robot changes from almost fully retracted to almost fully stretched. In subsection 5.3.5 it is shown that this results in a considerable change of the dynamics of the robot. The ILC algorithm should be able to cope with these varying dynamics. Secondly, the combination of the high velocity and the serrated profile results in a nominal tracking error with considerable high-frequency components. In subsection 6.2.3 it is shown that the required tracking accuracy can only be realised if the frequency components of the tracking error beyond the first resonance frequency of the robot mechanism are compensated by the ILC algorithm.

5.2.2 Trajectory B

The weld seam that defines trajectory B is formed by the edge along the junction of two hydroformed tubes as depicted in figure 5.14. The hydroformed tubes constitute a conceptual design for the A-pillar and sill of a Land Rover Freelander car. The design is the result of a study of Corus on the application of hydroformed tubes in automotive (see also Van Tienhoven, 2008). Two of the four sides of the junction are welded. The dimensions of the seam geometry are illustrated in figure 5.15.

The seam is placed in the xy-plane of the $\mathcal{O}_{xyz}$ coordinate system (see section 5.1.1) at $z = -242\ mm$. The nominal setpoints for the robot motion are defined such that the focus point of the welding head moves along the seam. The xy-components of these setpoints are shown in figure 5.16. The nominal setpoints for the orientation of the welding head are defined such that the sensor's x'-direction is aligned with the local direction of the seam and the z'-direction is kept at an angle of 29 degrees with respect to the global z-direction, which is needed to be able to access the seam. The velocity profile, depicted in figure 5.17, is trapezoidal with a maximum velocity of 50 mm/s and an acceleration of 100 mm/s^2. The nominal setpoints for the joint angles are computed from the nominal setpoints for the position and orientation of the welding head and the velocity profile using a kinematic model of the Stäubli RX90 robot (Corke, 1996). The nominal setpoints for the joint angles are depicted in figure 5.18. Note that the tracking of the small radius of the circular part of the trajectory requires high joint velocities and accelerations. The configuration of the Stäubli RX90 robot along trajectory B is shown in figure 5.19.

The location of the seam with respect to the nominal setpoints for the position of the welding head is measured with the sensor by commanding the



(the arrows indicate the directions of $\mathcal{O}_{xyz}$, not the origin)

Figure 5.14: The weld seam defining trajectory B

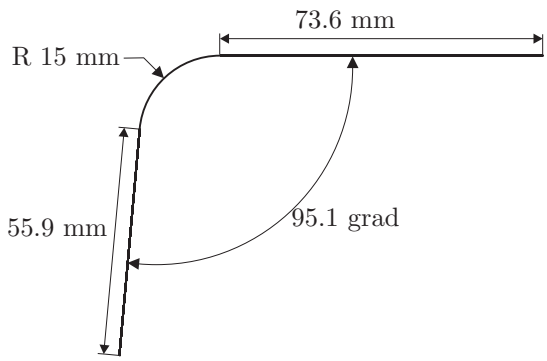


Figure 5.15: The dimensions of the weld seam defining trajectory B

robot to move along the trajectory at 10 mm/s. At this low speed geometrical effects dominate the tracking error. Figure 5.16 shows the measured location of the seam, which is obtained by adding the measured tracking error in the y'-direction to the nominal position setpoints for the welding head. The small constant deviation is the result of a misalignment of the seam with respect to the nominal trajectory. Next, the location of the seam with respect to the welding head is measured while the robot is commanded to move along the trajectory at nominal velocity. The measured tracking error at nominal speed, which is referred to as the nominal tracking error, should be reduced by the application of ILC. Figure 5.20 shows the measured tracking error in the y' and z'-direction for 10 mm/s and 50 mm/s. The constant error due to the misalignment in the y'-direction is also visible for 50 mm/s. The difference between the tracking error at 10 mm/s and 50 mm/s is caused by dynamic effects. The most notable effect of the dynamics is the large tracking error in the circular part of the trajectory where the velocities and accelerations are high. Moreover, the acceleration at the start of the trajectory results in a small error.

The geometry of trajectory B is typical for weld seam trajectories in industry, which often consist of straight sections interconnected with circular parts. The tracking errors along such trajectories are mostly similar to the tracking errors along trajectory B, i.e., constant errors due to misalignment, errors during acceleration an deceleration and errors along circular sections due to the centripetal acceleration. The ability to compensate the tracking errors along trajectory B is thus illustrative for the potential benefit of the application of ILC for many industrial applications.

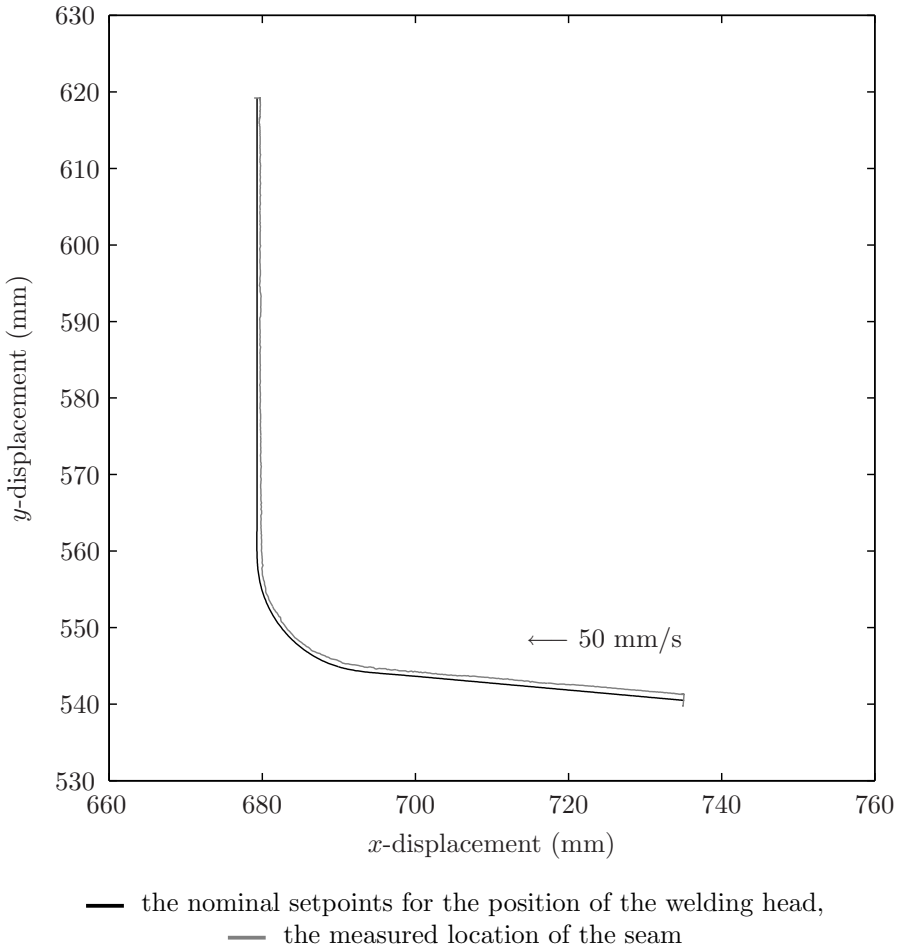


Figure 5.16: The location of trajectory B

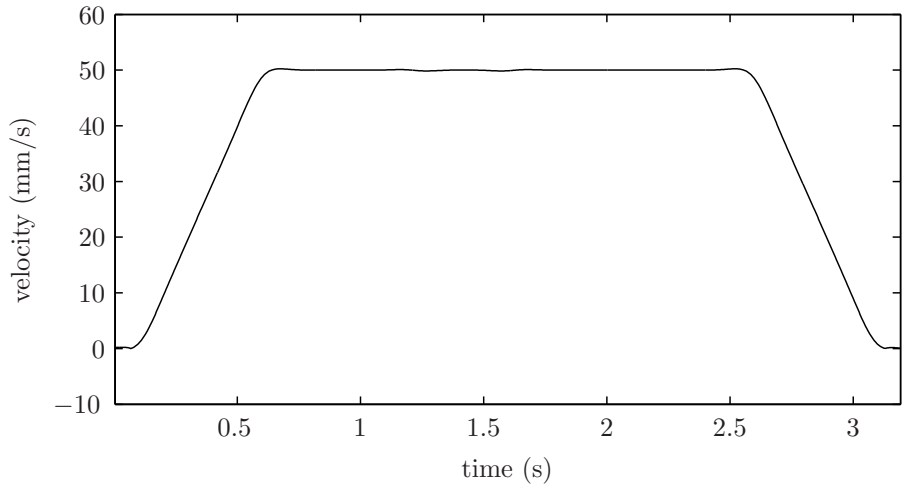


Figure 5.17: The velocity profile along trajectory B

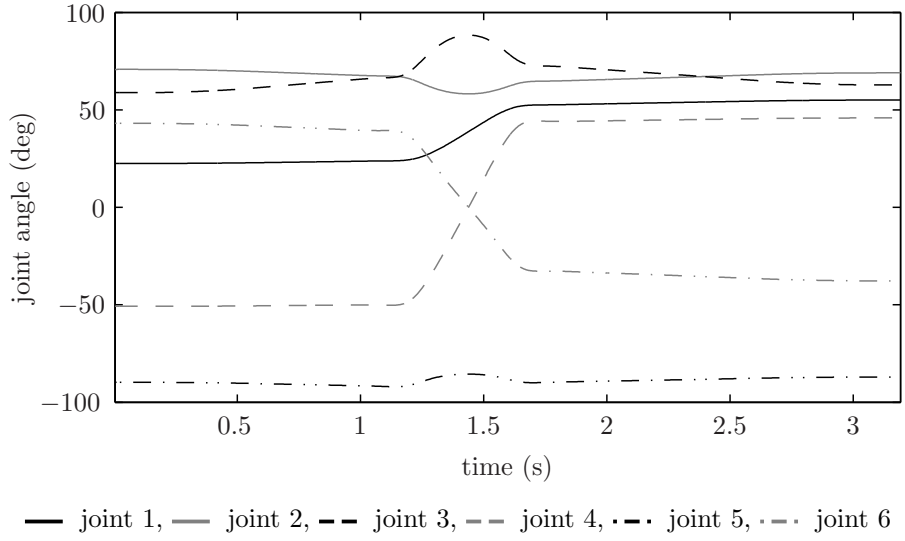
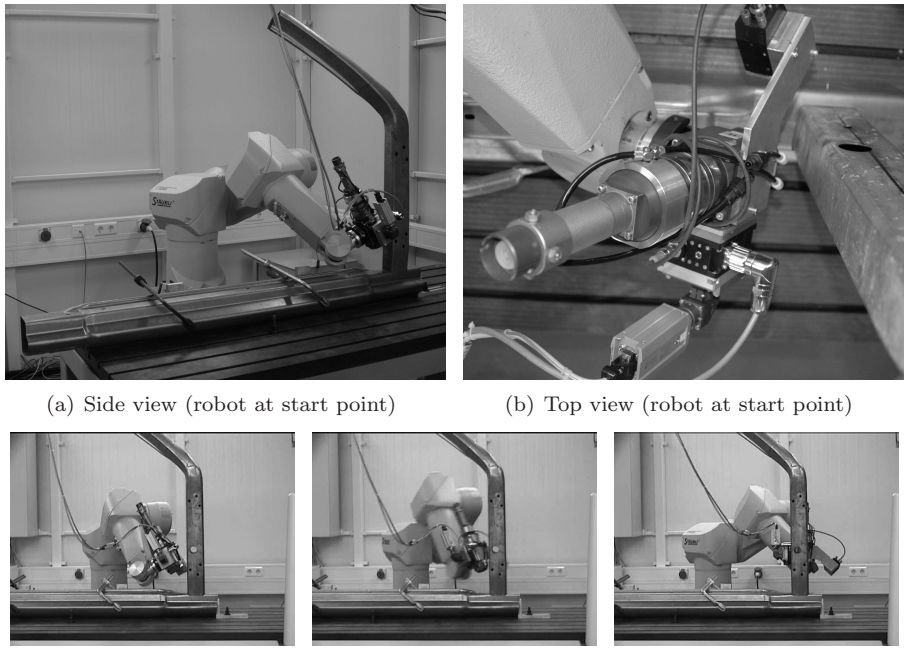


Figure 5.18: The nominal setpoints for the joint position along trajectory B



(a) Side view (robot at start point) (b) Top view (robot at start point)

(c) Robot at start point (d) Robot at mid point (e) Robot at end point

Figure 5.19: Robot movement along trajectory B

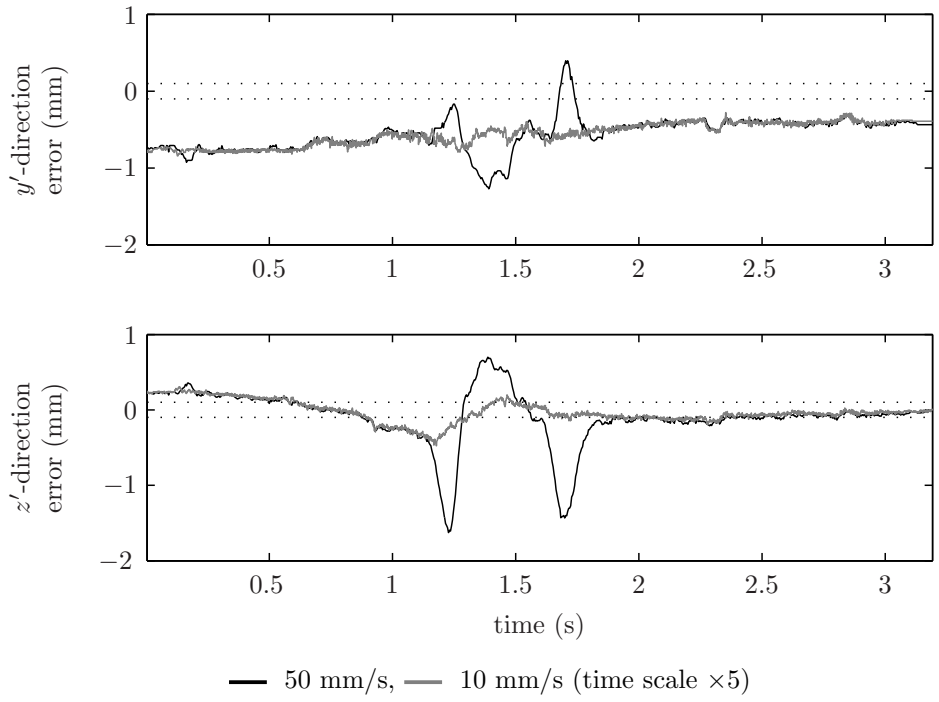


Figure 5.20: The nominal tracking error along trajectory B

5.3 Dynamic modelling

5.3.1 Introduction

A model of the dynamics of the robot system is required for the implementation of the NILC and RILC algorithms proposed in chapters 3 and 4 respectively. The model should describe the dynamic response of the tracking error that should be reduced to the feedforward input signal that is updated by ILC. As discussed in subsection 5.1.4, the tracking error measured by the seam-tracking sensor in the y' and z'-direction should be reduced by updating the setpoints for the position of the welding head in those directions. The model should thus describe the dynamic response of the measured tracking error to the setpoints for the position of the welding head. These dynamics are the result of the dynamics of the robot mechanism and the controller.

Several methods are proposed in literature to model the dynamics of an industrial robot based on physical considerations. These models are able to describe the robot dynamics along any feasible trajectory. Waiboer (2007) developed such model for the Stäubli RX90 robot system. The dynamics of the robot mechanism have been modelled using a non-linear finite element formulation. This formulation describes the effect of the inertia of the links, the rotational inertia of the driving system and the gravitation compensation spring of the robot. Moreover, the friction in the driving system is modelled with a detailed friction model, which is based on insights from tribological models. Parameters of the model are identified from experimental data. A detailed dynamic model of the controller dynamics is obtained from data of the manufacturer. The resulting model accurately predicts the low-frequency components of the joint torques and the tracking error of the Stäubli RX90 robot. However, the model is not able to predict the tracking error beyond the bandwidth of the robot system. At these frequencies the dynamics of the robot are significantly affected by flexibilities in the mechanism, which are not included in the model of Waiboer (2007). Hardeman (2008) has shown that the dominant flexibilities in the mechanism of the Stäubli RX90 robot are the flexibilities in the driving and bending direction of the joints and proposes an extension of the finite element model to include these flexibilities. The identification of all stiffness and inertia parameters of such physically parameterised robot model including flexibilities is addressed by Hardeman (2008); Tjepkema (2008); Wernholt and Gunnarsson (2006), but the research has not resulted in a method for the identification of all model parameters so far. Thus, currently no model of the robot dynamics is available that is based on physical considerations and that is able to describe the robot dynamics along any feasible trajectory at frequencies beyond the bandwidth of the robot system. The development of such model is recommended for future research, but it is not the main focus of this work.

The robot model that is used in this thesis is not based on physical considerations, but suffices to demonstrate the effectiveness of ILC for realising

high-accuracy motion. A model structure is adopted that is able to describe the multidimensional, configuration dependent robot dynamics, including the high-frequency dynamics resulting from flexibilities in the robot manipulator. The parameters of this so-called black box model are estimated from measurements of the dynamic response of the robot along a specific trajectory. The resulting model is only suited to describe the dynamic response of the robot near this trajectory. The model structure is described in subsection 5.3.2 and the procedure for identification of the model parameters is proposed in subsection 5.3.3. This identification procedure is used to model the dynamics of the Stäubli RX90 robot along the trajectories from section 5.2. The data acquisition is described in subsection 5.3.4 and the resulting models are presented in subsection 5.3.5. Finally, in subsection 5.3.6, the uncertainty in the modelled dynamics is specified. This specification of the model uncertainty is needed for the implementation of RILC.

5.3.2 Model Structure

The black-box model structure should be able to describe the multidimensional configuration dependent dynamics of the Stäubli RX90 robot system. An extensive study on black-box models and the identification of their parameters is published by Ljung (1999). The Auto-Regressive model with eXogenous inputs (ARX-model) is suited for the description of time-invariant multidimensional dynamics and the model structure has the advantageous property that its parameters can be estimated efficiently using linear regression. In this work, the ARX-model structure is extended with time-varying parameters to be able to describe the configuration dependent robot dynamics. The model is denoted as the Time-Varying Auto-Regressive model with eXogenous inputs (TVARX-model). Previously, a similar model structure was used by Fujimori et al. (2004) to model the dynamics of a robot manipulator for the design of a gain-scheduled feedback controller and by Petsounis and Fassois (2000) to analyse the non-stationary vibrations at the end-effector of a flexible planar manipulator as a result of stationary torque vibrations at the joints.

The input-output relation of the TVARX-model is described by the following equations

$$\bar{\bar{x}}_i^k = \sum_{j=1}^{N_a} \bar{\bar{A}}_{j,i} \bar{\bar{x}}_{i-j}^k + \sum_{j=N_c}^{N_b+N_c-1} \bar{\bar{B}}_{j,i} f_{i-j}^k + v_i^k, \tag{5.1}$$

$$e_i^k = \bar{\bar{x}}_i^k + d_i, \tag{5.2}$$

where e_i^k is the error that should be compensated with ILC, f_i^k the feedforward that is updated by ILC, $\bar{\bar{x}}_i$ is a state-variable, d_i represents the effect of iteration-invariant disturbances and v_i the effect of a stochastic (iteration-varying) disturbances. Matrices $\{\bar{\bar{A}}_{j,i}, \bar{\bar{B}}_{j,i}\}$ are time-varying matrices and describe the dynamics of the system. Parameters N_a and N_b describe the order

of the system and parameter N_c denotes the number of delays of the model. The effect of iteration-varying disturbances v_i is assumed to be white for the estimation of the model parameters. The consequence of this assumption for the parameter estimation is discussed at the end of subsection 5.3.3.

The proposed model structure is used to construct three models of the robot dynamics:

- Model 1 describes the dynamics of the robot system below its bandwidth, where the output perfectly tracks the input with a fixed delay, but ignores the high-frequency dynamics. This model is realised by taking $N_a = 0$, $N_b = 1$, $\bar{\bar{B}}_{N_c,i} = I$.
- Model 2 describes the dynamics of the robot system including its high-frequency dynamics, but ignores the variation along the trajectory. This model is realised by taking the same matrices $\{\bar{\bar{A}}_{j,i}, \bar{\bar{B}}_{j,i}\}$ for all i and estimating these matrices from measurement data.
- Model 3 describes the dynamics of the robot system including the high-frequency dynamics and the (slow) variation along the trajectory. This model is realised by piece-wise linear variation of the components of matrices $\{\bar{\bar{A}}_{j,i}, \bar{\bar{B}}_{j,i}\}$ and estimating these matrices from measurement data.

The reduction of the tracking error with ILC is tested for all three models. The experimental results are described in chapter 6. Those results show the effect of the model complexity on the achievable reduction of the error. In particular, the results for model 3 show the benefit of an LTV dynamic model and an ILC algorithm that is able to cope with LTV dynamics for the reduction of the tracking error.

The components of matrices $\{\bar{\bar{A}}_{j,i}, \bar{\bar{B}}_{j,i}\}$ for models 2 and 3 are estimated by means of identification. The number of parameters for the time-invariant model 2 equals $N_e \times (N_e \times N_a + N_f \times N_b)$, where N_e is the dimension of error e_i^k and N_f is the dimension of feedforward f_i^k. The number of parameters for the time-varying model 3 would be $N_e \times (N_e \times N_a + N_f \times N_b) \times N_i$ if the parameter values at different time-instances were taken independent. The number of parameters that needs to be estimated is reduced by approximating the slow variation of the parameters along the trajectory as piece-wise linear with respect to some predefined time-dependent interpolation parameter ξ_i. Using this predefined interpolation parameter, the parameters of the model can still be estimated from linear regression. The piece-wise linear interpolation of the matrices of the model is expressed by

$$\bar{\bar{A}}_{j,i} = \sum_{n=1}^{N_n} \psi_{n,i} \bar{\bar{A}}_{j,n}, \tag{5.3}$$

$$\bar{\bar{B}}_{j,i} = \sum_{n=1}^{N_n} \psi_{n,i} \bar{\bar{B}}_{j,n}, \tag{5.4}$$

where $\psi_{n,i}$ is defined as

$$\psi_{n,i} = \begin{cases} 2 - n + \xi_i & \text{if} \quad n-2 < \xi_i \leq n-1, \\ n - \xi_i & \text{if} \quad n-1 < \xi_i < n, \\ 0 & \text{else.} \end{cases} \tag{5.5}$$

The interpolation parameter ξ_i is taken proportional to the distance along the trajectory and scaled such that it is equal to zero at the start of the trajectory and equal to $N_n - 1$ at the end. The use of the piece-wise linear interpolation reduces the number of unknown parameters to $N_e \times (N_e \times N_a + N_f \times N_b) \times N_n$. The estimation of these unknown parameters is discussed in subsection 5.3.3.

The efficient implementations of NILC and RILC, proposed in subsections 3.3.2 and 4.3.2 respectively, are based on the state-space equations (3.1). Such state-space equations are obtained from the TVARX model by the following transformation

$$x_{i+1}^k = \begin{bmatrix} \bar{\bar{x}}_{i+1}^k \\ \sum_{j=2}^{N_o} \left(\bar{\bar{A}}_{j,i+2} \bar{\bar{x}}_{i+2-j}^k + \bar{\bar{B}}_{j,i+2} f_{i+2-j}^k \right) \\ \vdots \\ \bar{\bar{A}}_{N_o,i+N_o} \bar{\bar{x}}_i^k + \bar{\bar{B}}_{N_o,i+N_o} f_i^k \end{bmatrix}, \tag{5.6a}$$

$$A_i = \begin{bmatrix} \bar{\bar{A}}_{1,i+1} & I & O & \dots & O \\ \bar{\bar{A}}_{2,i+2} & O & I & \ddots & O \\ \bar{\bar{A}}_{3,i+3} & O & O & \ddots & O \\ \vdots & \vdots & \ddots & \ddots & \ddots \\ \bar{\bar{A}}_{N_o,i+N_o} & O & O & \dots & O \end{bmatrix}, \tag{5.6b}$$

$$B_i = \begin{bmatrix} \bar{\bar{B}}_{1,i+1} \\ \bar{\bar{B}}_{2,i+2} \\ \bar{\bar{B}}_{3,i+3} \\ \vdots \\ \bar{\bar{B}}_{N_o,i+N_o} \end{bmatrix}, \tag{5.6c}$$

$$C_i = \begin{bmatrix} I & O & O & \dots & O \end{bmatrix} \tag{5.6d}$$

where $N_o = \max(N_a, N_b + N_c)$ and $\bar{\bar{A}}_{j,i} = O$ for $j > N_a$ and $\bar{\bar{B}}_{j,i} = O$ for $j < N_c$ and $j > N_b + N_c - 1$. Note that the state-space matrices at time-instance i depend on matrices $\{\bar{\bar{A}}_{j,i}, \bar{\bar{B}}_{j,i}\}$ of the TVARX model at multiple future time instances.

As described in subsection 5.1.4, the Stäubli RX90 robot is commanded to move to the initial position setpoint of the iteration and to wait for 0.5 s prior to each movement along the trajectory. As a result, the Stäubli RX90 robot is in its steady-state at the updated initial position setpoint at the start of each movement. This is modelled by using the following equation for the initial state

$$x_1^{k+1} = (I - A_1)^{-1} B_1 f_1^{k+1}. \tag{5.7}$$

5.3.3 Parameter identification procedure

In the previous subsection a model structure is proposed to describe the configuration dependent dynamics of the Stäubli RX90 robot. The proposed model structures is used to construct three models, where the parameters of models 2 and 3 are estimated from experimental data. The estimation procedure is described in this subsection.

The components of $\{\bar{\bar{A}}_{j,n}, \bar{\bar{B}}_{j,n}\}$ are estimated from measurement data by means of parameter identification. Measurements of the response of the tracking error to a set of broadband excitations of the feedforward input are used for the identification. Moreover, the tracking error is measured for zero feedforward to compensate for the trial-invariant disturbance d_i. A superscript is used to denote the measurement series, where 0 refers to the measurement in which the feedforward input is set to zero. The acquisition of the measurement data is described in subsection 5.3.4. The parameter estimates are obtained from the minimisation of the prediction error, which is the difference between the measured error and the model-based prediction of the error. The model-based prediction of the error is defined as

$$\begin{aligned}\hat{e}_i^k = \bar{\bar{x}}_i^k - \bar{\bar{x}}_i^0 + e_i^0 = e_i^0 + \sum_{j=1}^{N_a} \sum_{n=1}^{N_n} \psi_{n,i} \bar{\bar{A}}_{j,n} \left(e_{i-j}^k - e_{i-j}^0\right) \\ + \sum_{j=N_c}^{N_b+N_c-1} \sum_{n=1}^{N_n} \psi_{n,i} \bar{\bar{B}}_{j,n} \left(f_{i-j}^k - f_{i-j}^0\right),\end{aligned} \tag{5.8}$$

where the contribution of the stochastic disturbances v_i^k and v_i^0, which are assumed to be white, is set to zero. Note that the model based prediction is based on measurements of the error and the feedforward in previous time-steps and in iteration 0.

Stacking the model-based prediction of the error for N_k measurement series of length N_i gives

$$\begin{bmatrix} \hat{e}^1_{N_o+1} \\ \vdots \\ \hat{e}^1_{N_i} \\ \vdots \\ \hat{e}^{N_k}_{N_o+1} \\ \vdots \\ \hat{e}^{N_k}_{N_i} \end{bmatrix} = \begin{bmatrix} e^0_{N_o+1} \\ \vdots \\ e^0_{N_i} \\ \vdots \\ e^0_{N_o+1} \\ \vdots \\ e^0_{N_i} \end{bmatrix} + \sum_{j=1}^{N_a} \sum_{n=1}^{N_n} \psi_{n,i} \bar{\bar{A}}_{j,n} \begin{bmatrix} e^1_{N_o+1-j} - e^0_{N_o+1-j} \\ \vdots \\ e^1_{N_i-j} - e^0_{N_i-j} \\ \vdots \\ e^{N_k}_{N_o+1-j} - e^0_{N_o+1-j} \\ \vdots \\ e^{N_k}_{N_i-j} - e^0_{N_i-j} \end{bmatrix}$$

$$+ \sum_{j=N_c}^{N_b+N_c-1} \sum_{n=1}^{N_n} \psi_{n,i} \bar{\bar{B}}_{j,n} \begin{bmatrix} f^1_{N_o+1-j} - f^0_{N_o+1-j} \\ \vdots \\ f^1_{N_i-j} - f^0_{N_i-j} \\ \vdots \\ f^{N_k}_{N_o+1-j} - f^0_{N_o+1-j} \\ \vdots \\ f^{N_k}_{N_i-j} - f^0_{N_i-j} \end{bmatrix}. \quad (5.9)$$

The parameter estimate is obtained from minimising the difference between this vector of the model-based predictions of the error and the vector of the measurements of the error. The right-hand side of equation (5.9) is affine in the unknown $\{\bar{\bar{A}}_{n,i}, \bar{\bar{B}}_{n,i}\}$ and thus the minimising parameters can be computed efficiently from linear regression. This efficient estimation procedure of the parameters is the main advantage of using the proposed TVARX model structure.

The effect of the stochastic disturbance v_i^k, which is assumed to be white, on the output of the TVARX model is coloured by the dynamics of the model. It is observed that the effect of stochastic disturbances on the tracking error of the Stäubli RX90 robot is also not white, but coloured by the robot dynamics. However, it is not verified if the effect of the stochastic disturbances can be correctly represented by a white noise source affecting the dynamics as v_i^k in equation (5.2). If this assumption is not correct, then the estimation procedure could result in a biased estimate of the parameters of the TVARX model. A conventional time-invariant ARX model has the same disadvantage, which is analysed in detail by Ljung (1999). Despite the fact that the proposed identification procedure could result in a biased estimate of the parameters, the TVARX model structure is used in this work because of the computational efficiency of the parameter estimation.

5.3.4 Data acquisition

Measurement of the response of the error to a set of broadband excitations of the feedforward input are required for the estimation of the parameters of model structure from subsection 5.3.2 using the identification procedure from subsection 5.3.3. As described in subsection 5.1.4 the tracking error of the welding head in the y' and z'-direction is considered as the output of the system and the update of the position setpoints in those directions is the input manipulated by ILC.

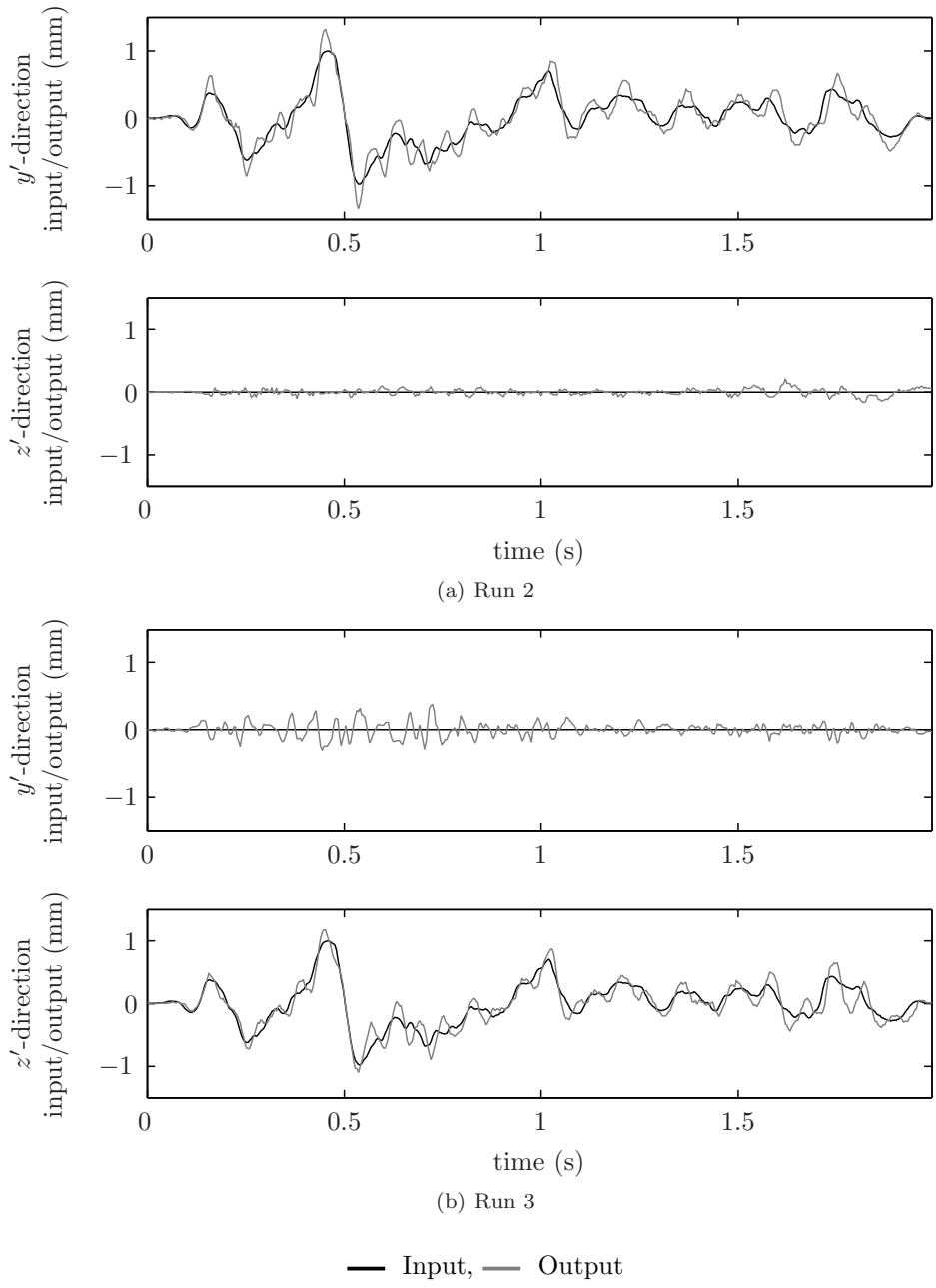
Ten different multisine realisations are used for the excitation of the feedforward input. Each multisine realisation consists of all frequency components between 1 Hz and 125 Hz of which an integer number of periods fit within the length of the iteration. The amplitudes of the frequency components up to 5 Hz are taken equal and the amplitudes of frequency components beyond 5 Hz decrease quadratically with the frequency. The phase of the frequency components are selected randomly and a different set of randomly selected phases is used for each of the multisine realisations. The first 50 samples and the last 50 samples of each multisine realisation are windowed to prevent discontinuities in the setpoints for the robot at the start and end of the iteration. The maximum absolute value of each multisine realisation is scaled to 1 mm to keep the weld seam within the range of the sensor.

The tracking error is measured in 10 series of 3 runs. For each series a different multisine realisation is used. In run 1 the feedforward input is set to zero and thus the tracking error for the nominal trajectory is measured. In runs 2 and 3 the y' and z' component of the feedforward input are excited respectively. The difference between the tracking error that is measured with and without the excitation is referred to as the *output* and the multisine excitation is referred to as the *input* hereafter.

Figures 5.21 and 5.22 show the input and the output in one of the measurement series for trajectories A and B respectively. Clearly, the low-frequency components of the input are traced by the output and the response to the low-frequency components of the input is decoupled. The high-frequency components of the input are amplified in the output and the response to the high-frequency components is coupled.

5.3.5 Estimated dynamic models

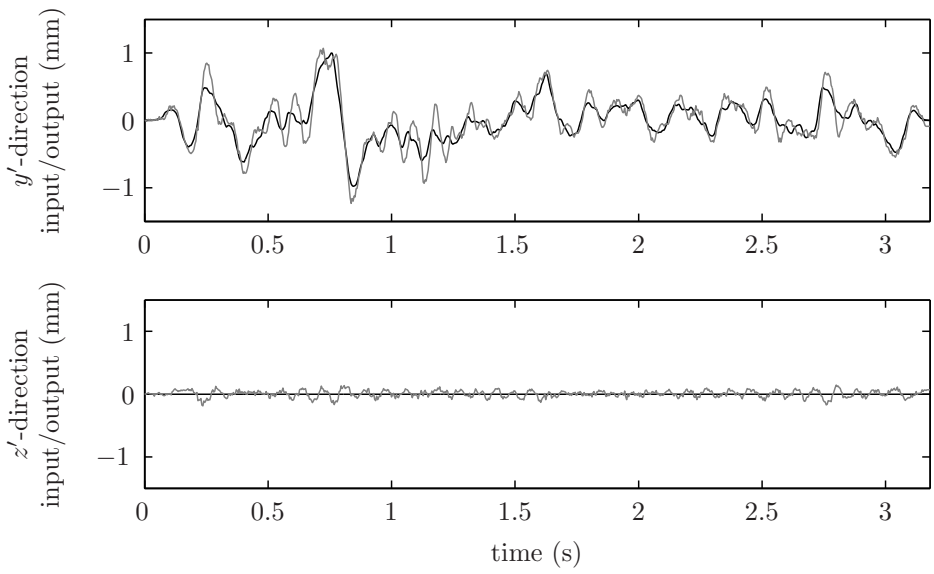
The dynamics of the robot in the vicinity of the trajectories described in section 5.2 are modelled using the TVARX model structure from subsection 5.3.2. As described in that subsection three different models are constructed using the TVARX model structure; The output of model 1 exactly traces the setpoints with a fixed delay, model 2 is a time-invariant model estimated from measurement data and model 3 is a time-varying model estimated from measurement data. The delay between the setpoints for the position of the welding head and the motion measured by the sensor is three samples (see subsection 5.1.2)



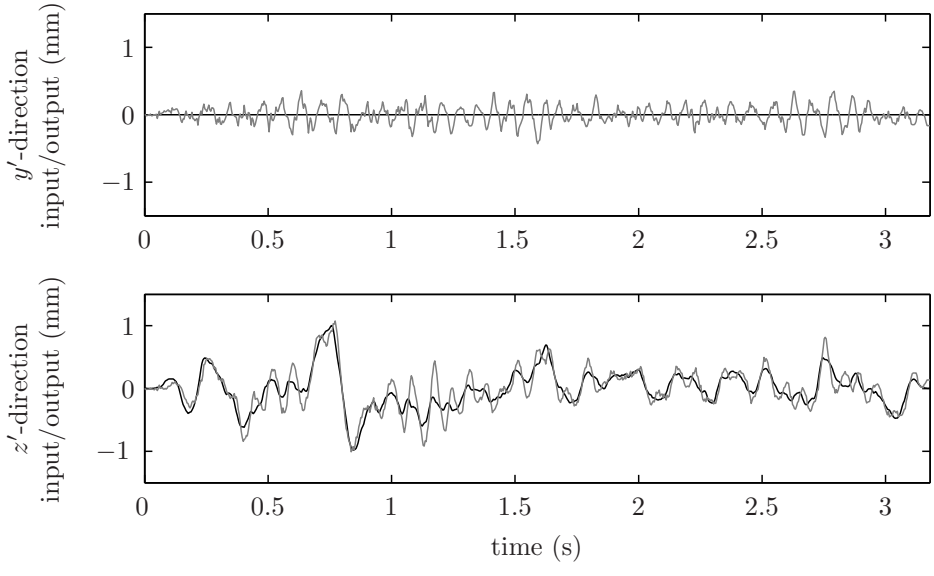
(a) Run 2

(b) Run 3

Figure 5.21: Response of the tracking error (output) to a multisine excitation added to the setpoints (input) for trajectory A



(a) Run 2



(b) Run 3

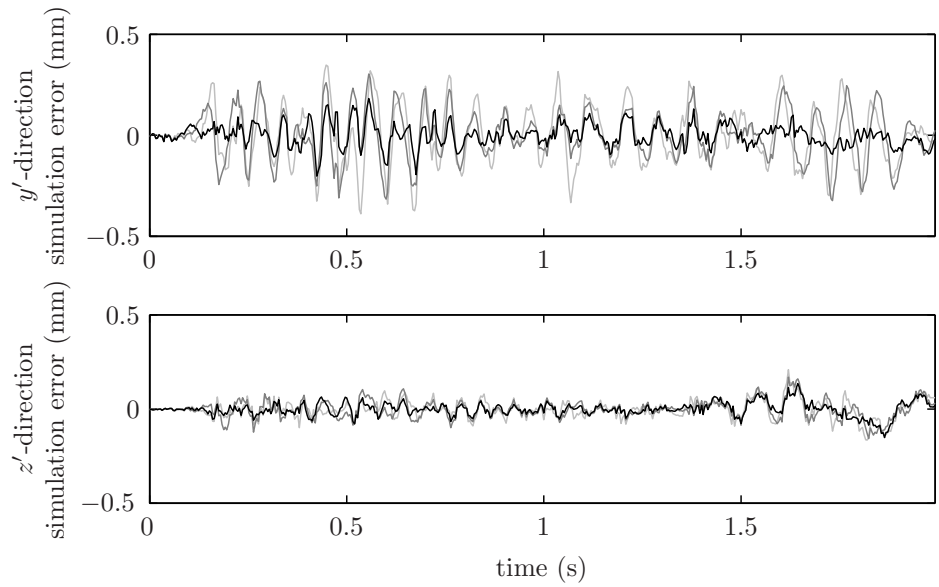
Input, Output

Figure 5.22: Response of the tracking error (output) to a multisine excitation added to the setpoints (input) for trajectory B

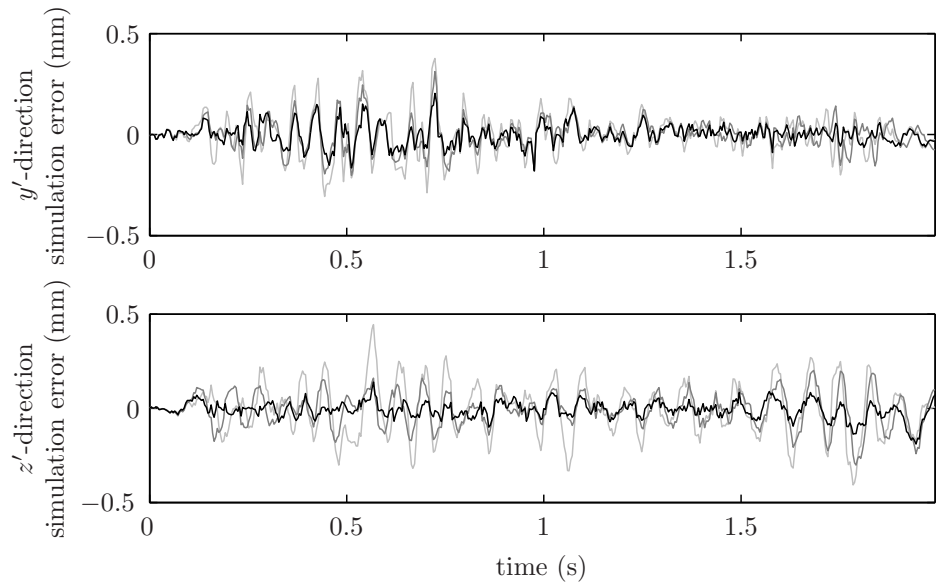
and thus $N_c = 3$ is taken for all three model. The parameters of models 2 and 3 are estimated using the identification procedure from subsection 5.3.3 with the measurement data described in subsection 5.3.4. The measurement data is split in two sets; the data from measurement series 1-5 are used for the estimation of the models and the data from measurement series 6-10 are used for the validation of the models. The parameters of 512 models are estimated using $N_a = 1 \ldots 8$, $N_b = 1 \ldots 8$ and $N_n = 1 \ldots 8$. The selection of the parameters N_a, N_b and N_n is based on the difference between the output that is measured in series 6-10 and the data that is simulated for the 512 models. The difference between the measured and simulated data is referred to as the simulation error. For the tested ranges of N_a, N_b and N_n, it is found that the larger the number of parameters, the smaller the norm of the simulation error in measurement series 6-10 for both trajectories. The best proper model with $N_n = 1$ is selected for model 2, i.e., $\{N_a, N_b, N_n\} = \{8, 7, 1\}$, and the best proper model is selected for model 3, i.e., $\{N_a, N_b, N_n\} = \{8, 7, 8\}$. This selection of $\{N_a, N_b, N_n\}$ is used for trajectories A and B.

Figures 5.23 and 5.24 show the simulation error of the three models for the measurement series of which the input and output data are shown in figures 5.21 and 5.22. Clearly, the simulation error of model 1 is larger than the simulation error of model 2, which is in turn larger than the simulation error of model 3. Models 2 and 3, which are based on identification, thus describe the robot dynamics better than model 1, which assumes perfect tracking of the setpoints. Moreover, the dynamics of the robot can be described better using the time-varying model 3 than using the time-invariant model 2. The difference between the simulation error of the time-invariant model 2 and time-varying model 3 is largest for trajectory A, because the large change of configuration of the robot along this trajectory results in a considerable variation of the dynamics. The simulation errors for measurement series 1-10 are used in subsection 5.3.6 for the specification of the model uncertainty.

The dynamics of models 1 and 2 are characterised by their frequency response. Model 3 does not have a unique frequency response as its dynamics are time-varying. For comparison of model 3 with the others, the local frequency response is used, which is the frequency response of the state-space matrices of model 3 at a single time-step. Figures 5.25 and 5.26 show the frequency response of models 1 and 2 and the local frequency response of model 3 at the mid-point of both trajectories. Figures 5.27 and 5.28 show the local frequency responses of model 3 at the start, mid and end point of both trajectories. Hereafter, these frequency responses are discussed in relation to the dynamic behaviour of the robot system. The output of the robot system traces the low-frequency components of the feedforward input due to the high gain of the feedback controller below its bandwidth. This is modelled by the frequency responses of all three models, which are all approximately unity up to 5 Hz. The gain of the feedback controller rolls off at the bandwidth. This roll-off and the related phase behaviour of the feedback controller cause a peak in the frequency response.



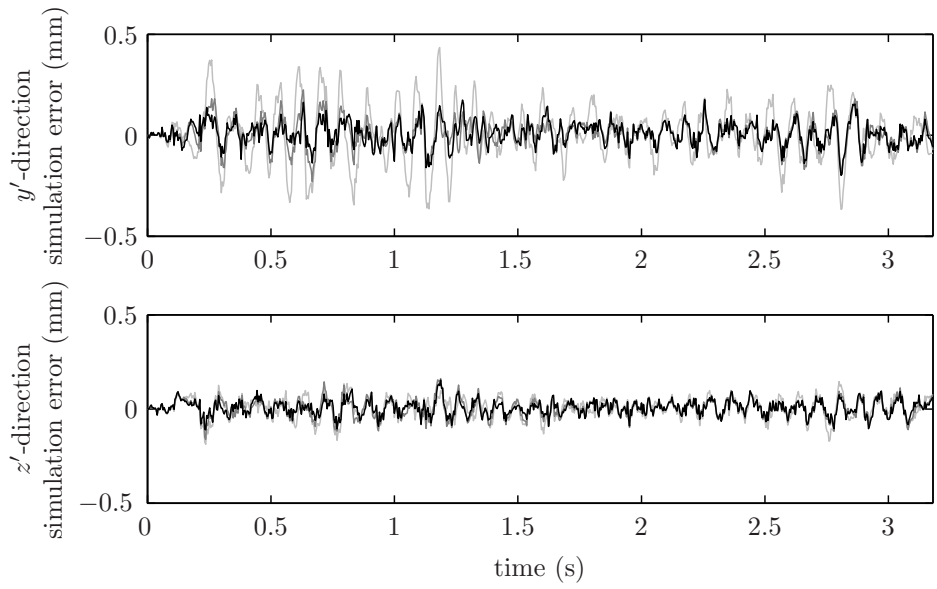
(a) Run 2



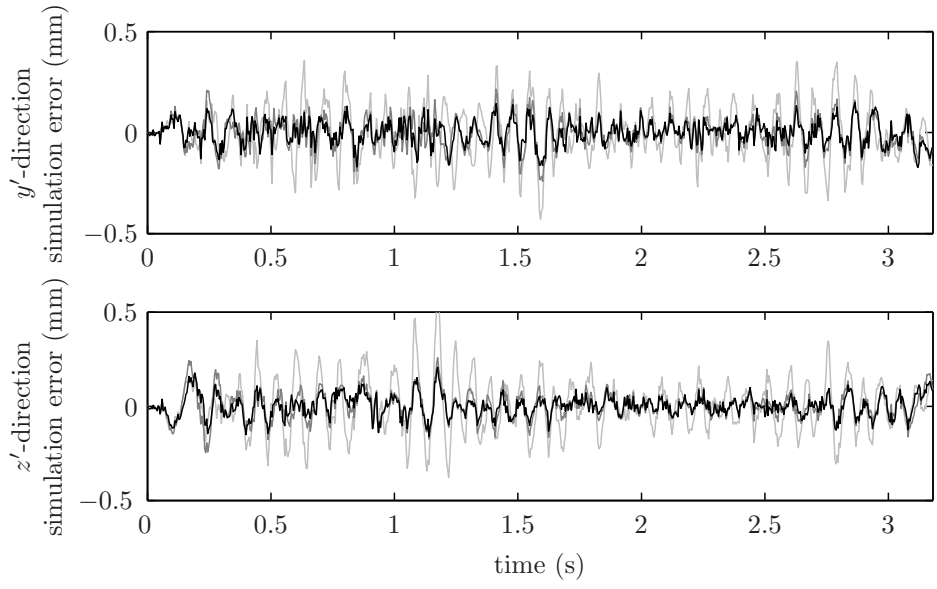
(b) Run 3

Model 1, Model 2, Model 3

Figure 5.23: Difference between the measured and the simulated response of the error to a multisine excitation added to the setpoints for trajectory A



(a) Run 2



(b) Run 3

Model 1, Model 2, Model 3

Figure 5.24: Difference between the measured and the simulated response of the error to a multisine excitation added to the setpoints for trajectory B

This is modelled by the frequency responses of models 2 and 3, which peak around 10 Hz. Beyond the closed-loop bandwidth the dynamics of the robot system depend on the high-frequency dynamics of the controller and the mechanism. The high-frequency dynamics of the controller are dominated by the velocity and acceleration feedforward of the motion controllers in the CS8. As a result of these feedforwards, the frequency response of the robot system does not roll-off at high-frequencies. This is modelled by the frequency responses of models 2 and 3, which do not roll off beyond 10 Hz. The flexibilities in the robot mechanism induce high-frequency resonance vibrations. These resonance vibrations and the lack of roll-off of the controller result in peaks in the closed-loop frequency response at high-frequencies. Those peaks are the cause of the amplification of the high-frequency components of the multisine excitation observed in figures 5.21 and 5.22. The resonance peaks are modelled by the frequency response of models 2 and 3. The closed-loop bandwidth and the resonance frequencies of the mechanism depend on the configuration of the robot. The local frequency response of the robot thus changes along a trajectory. The configuration dependency of the frequency response is only described by model 3. As a result of the large change of configuration of the robot along trajectory A, the dynamics of model 3 change the most along this trajectory. Note that the variation of the bandwidth even introduces a phase-difference of more than 90 degrees in the local frequency response for 13.5 Hz at the start and end of trajectory A.

From the previous it is concluded that model 1 only describes the dynamics of the robot system up to the resonance frequency, while models 2 and 3 also describe the dynamics beyond the resonance frequency. Furthermore, the variation of the dynamics of the robot, which is the largest along trajectory A, is only described by model 3. As mentioned before, the reduction of the tracking error of the Stäubli RX90 robot with ILC is tested for all three models. The experimental results, which are presented in chapter 6, thus show the effect of the difference in model quality on the achievable reduction of the tracking error.

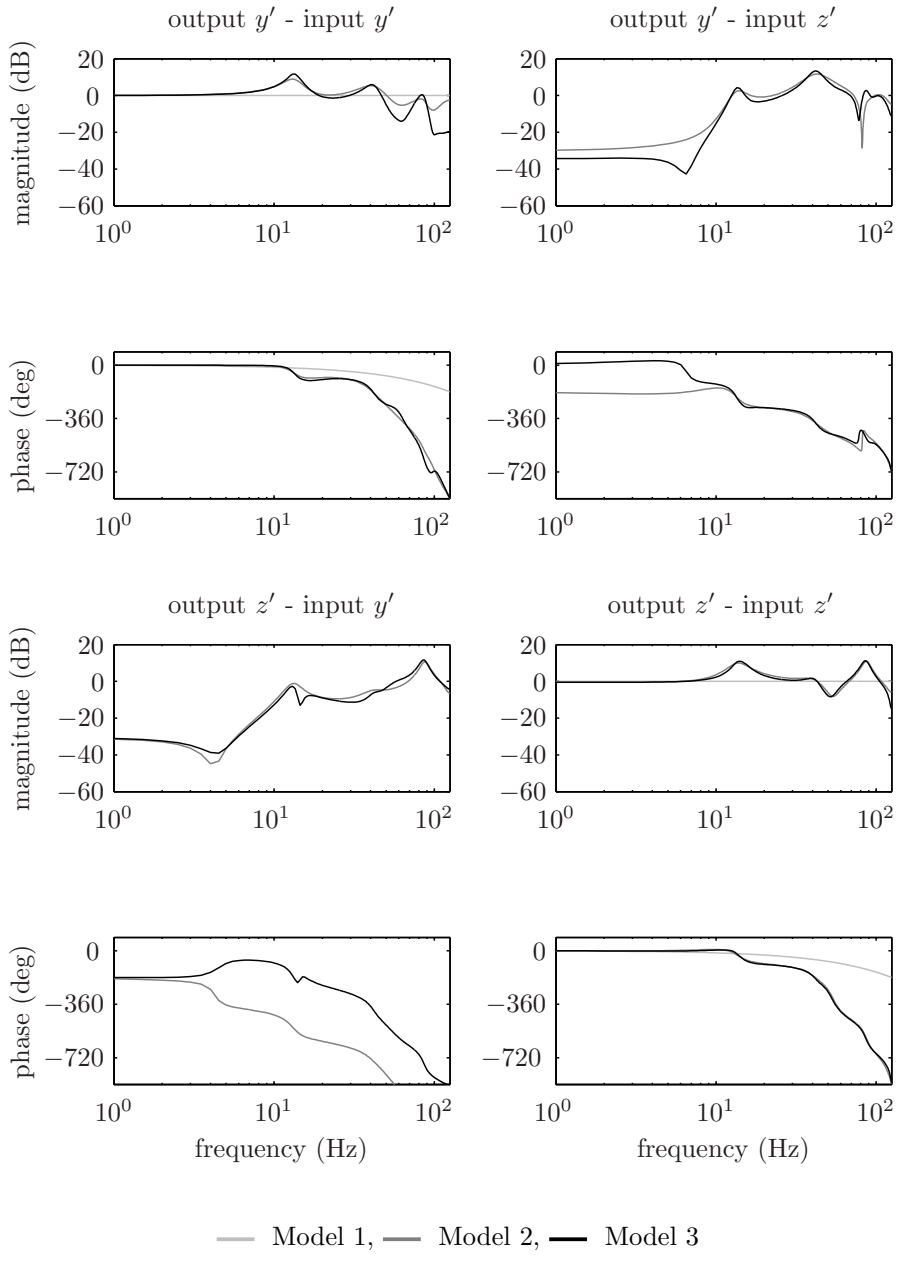


Figure 5.25: Frequency responses of models 1-2 and the local frequency response of model 3 at the mid point of trajectory A

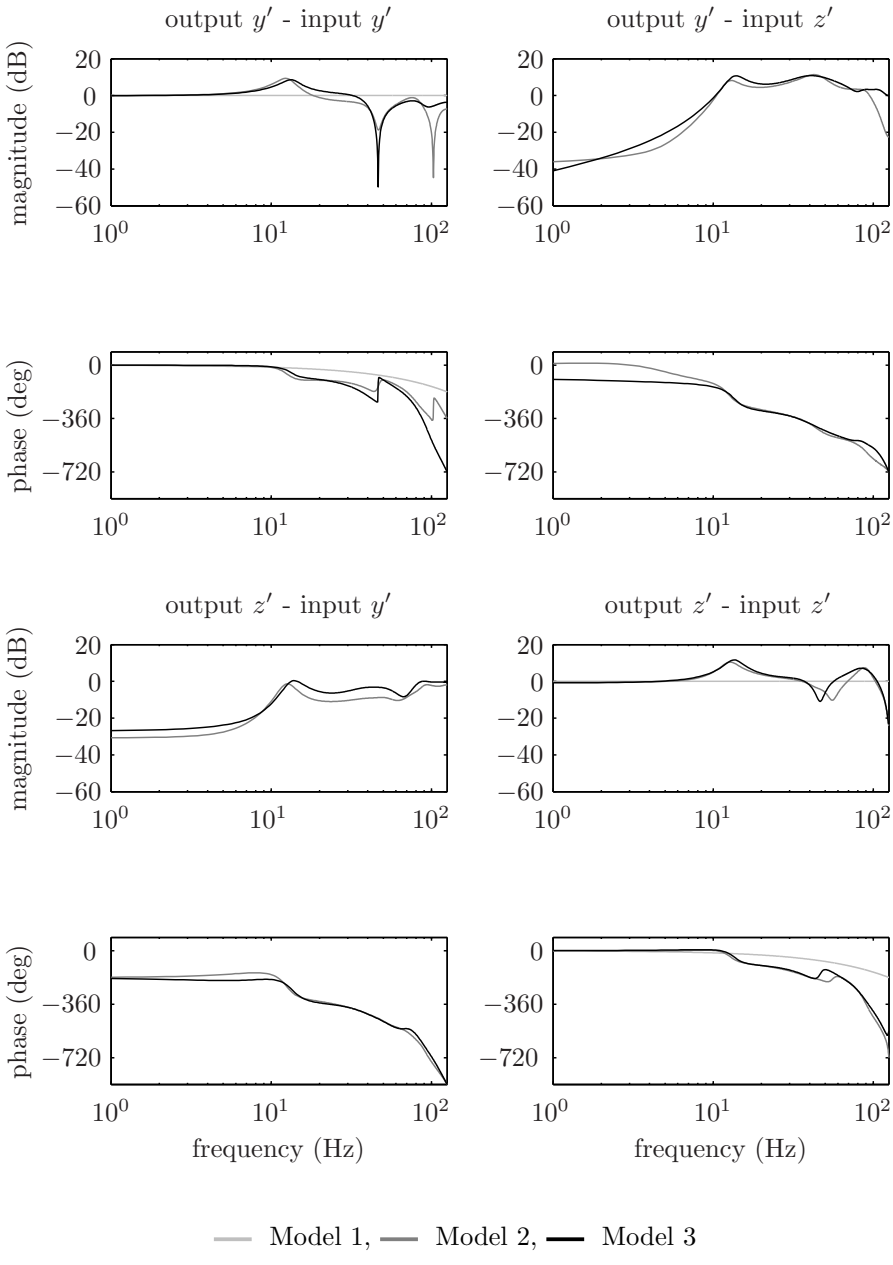


Figure 5.26: Frequency responses of models 1-2 and the local frequency response of model 3 at the mid point of trajectory B

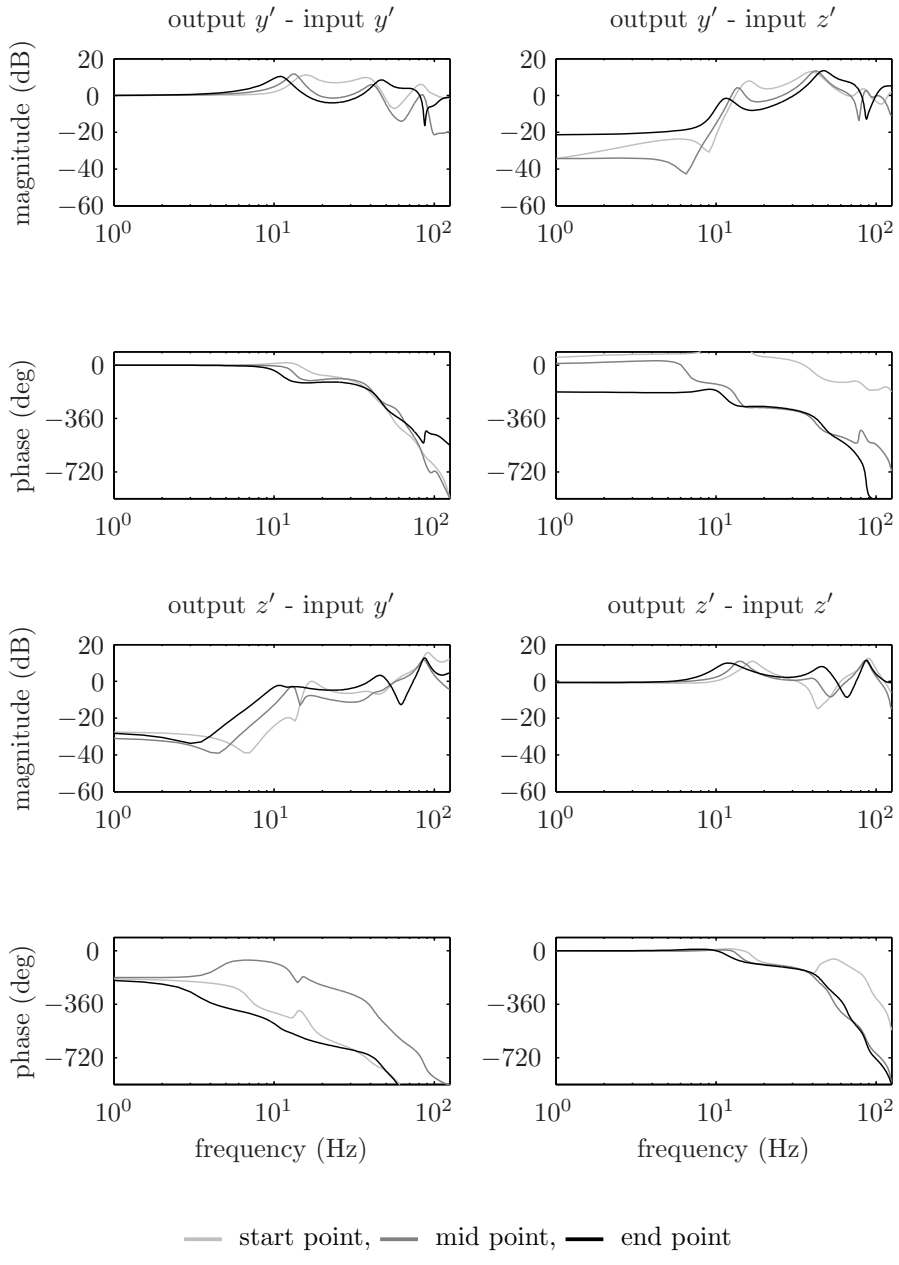


Figure 5.27: Local frequency responses of model 3 at three points along trajectory A

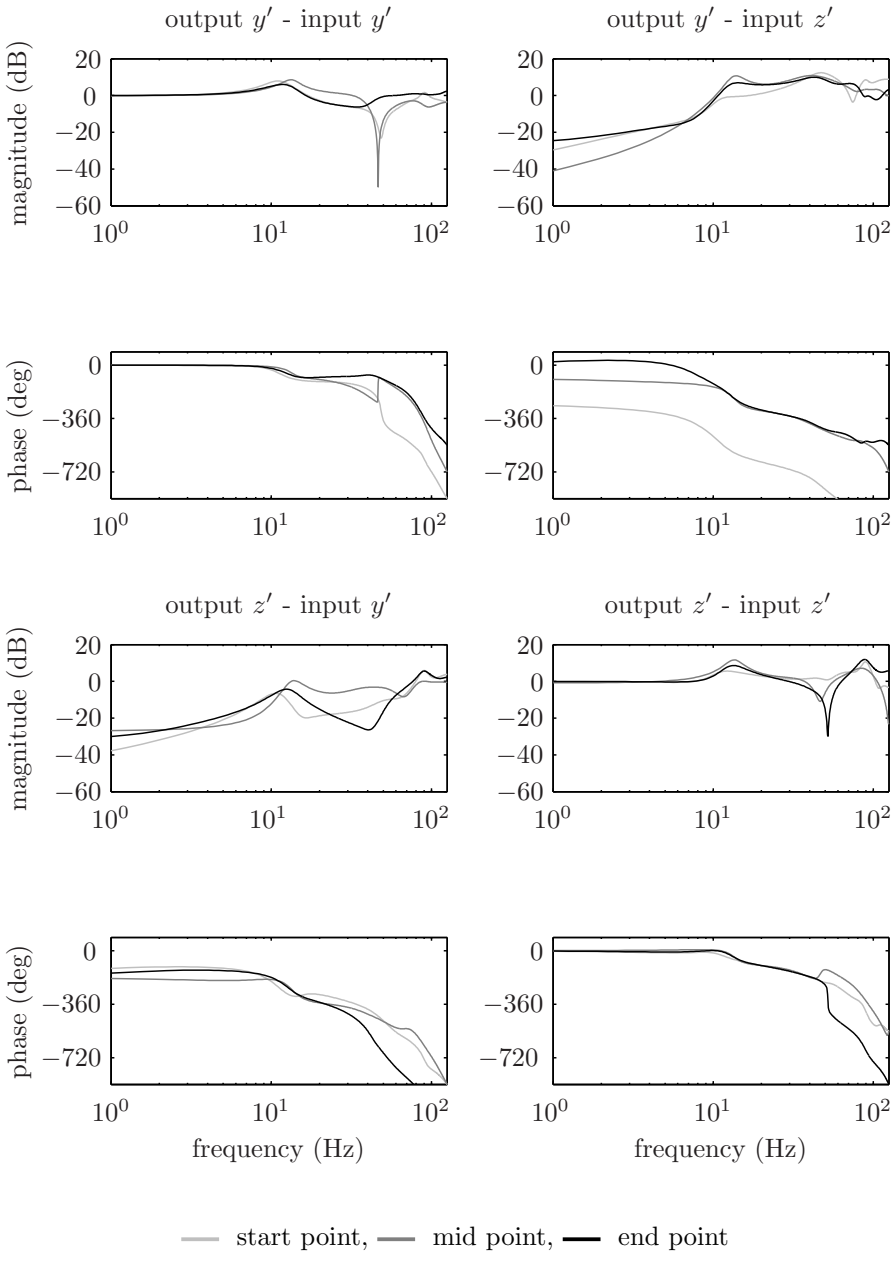


Figure 5.28: Local frequency responses of model 3 at three points along trajectory B

5.3.6 Model uncertainty

In the previous subsections models of the dynamics of the Stäubli RX90 robot are constructed. Besides these models, a specification of their uncertainty is required for the implementation of the RILC algorithm from chapter 4.

Like the modelling of the robot dynamics, the specification of the uncertainty is not the main focus of this work. A heuristic procedure is adopted to specify the model uncertainty, which suffices to show the potential effectiveness of RILC as a method to improve the tracking of an industrial robot. The development of a better founded approach for the specification of the model uncertainty is recommended for future research.

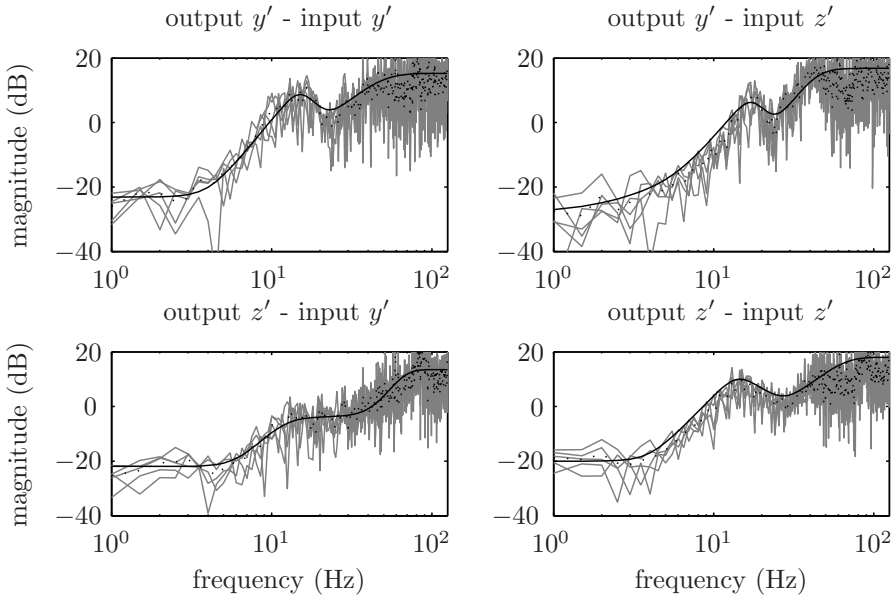
Method

The specification of the model uncertainty for each of the three models is derived from an estimation of the additive model error. The additive model error is estimated from the relation between the feedforward and the simulation error. This simulation error is the difference between output that is measured and the output that is simulated with the models and thus specifies the part of the measured output that is not predicted by the models. The simulation error is computed for all three models using the measurement data from the 10 measurement series described in subsection 5.3.4. These measurement data are also used for the identification of the model parameters in subsection 5.3.5.

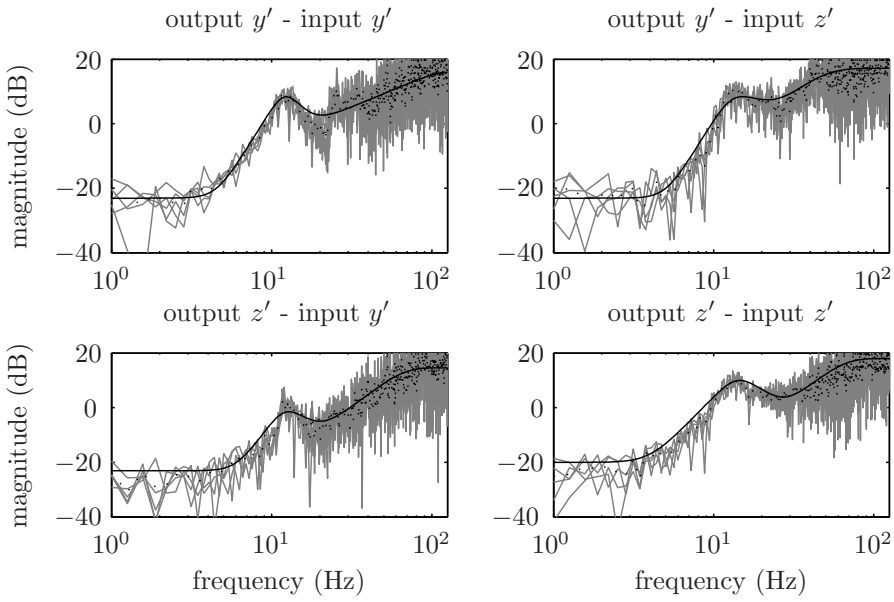
For simplicity, it is assumed that the additive model error is linear time-invariant, such that its estimation can be based on frequency domain analysis. The Fourier transform of the simulation error is divided by the Fourier transform of the feedforward to obtain an estimate of the transfer function of the additive model error for each of the measurement series. The gain of the transfer functions for the 10 measurement series is averaged to reduce the variance of the estimate. Subsequently, an estimate of the gain of the additive model error is obtained by manually fitting a transfer function through the average gain of the transfer functions. The pole locations of the transfer function are set by multiplication of Butterworth filters with manually selected order and cutoff frequency. The zero locations are set by multiplication with the inverse of Butterworth filters with a manually selected order and cut-off frequency.

Results

The measured, the averaged and the fitted transfer functions of the additive model error are depicted in figures 5.29-5.31. The fitted transfer function of the additive model error is used to specify the additive model uncertainty. Below the bandwidth of the robot system, where the system traces the setpoints with three delays, the estimated uncertainty is small for all models. The uncertainty of model 1 peaks at the bandwidth for both trajectories. These peaks are caused by the fact that model 1 does not describe the peak in the frequency



(a) Trajectory A



(b) Trajectory B

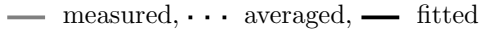
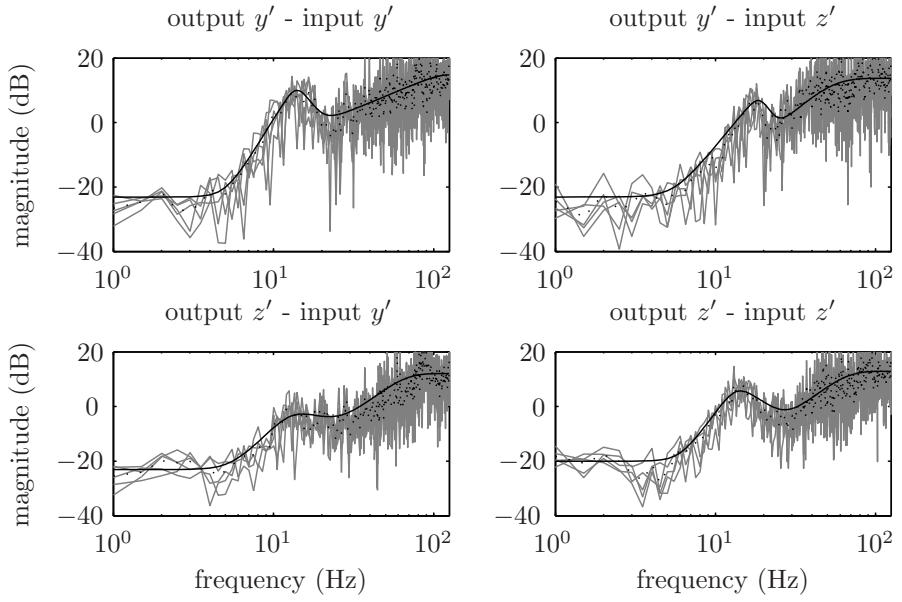
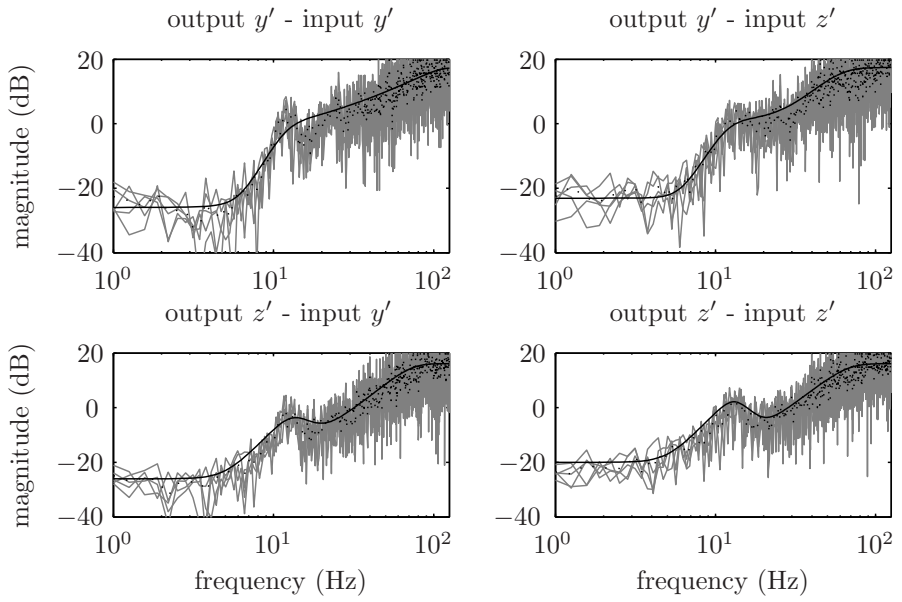


Figure 5.29: Transfer function of the additive model error of model 1



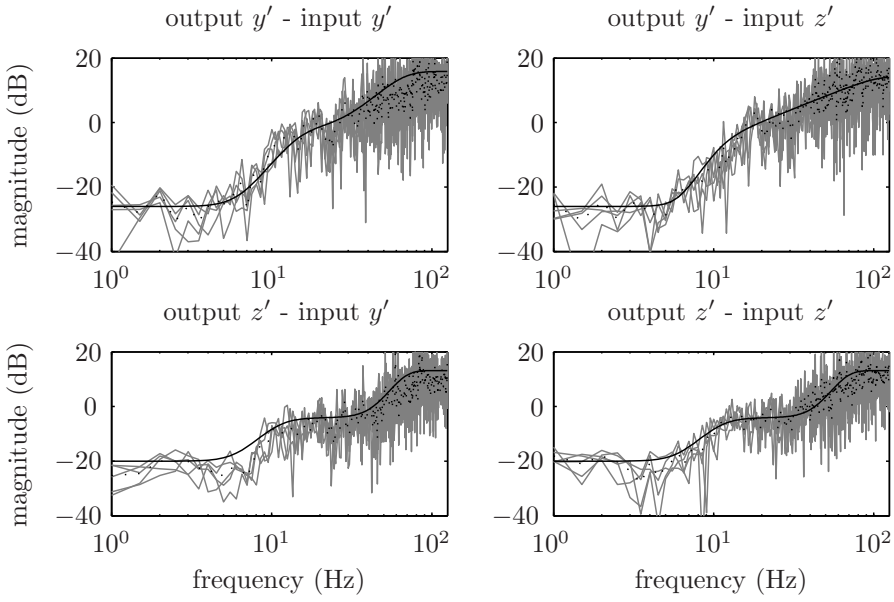
(a) Trajectory A



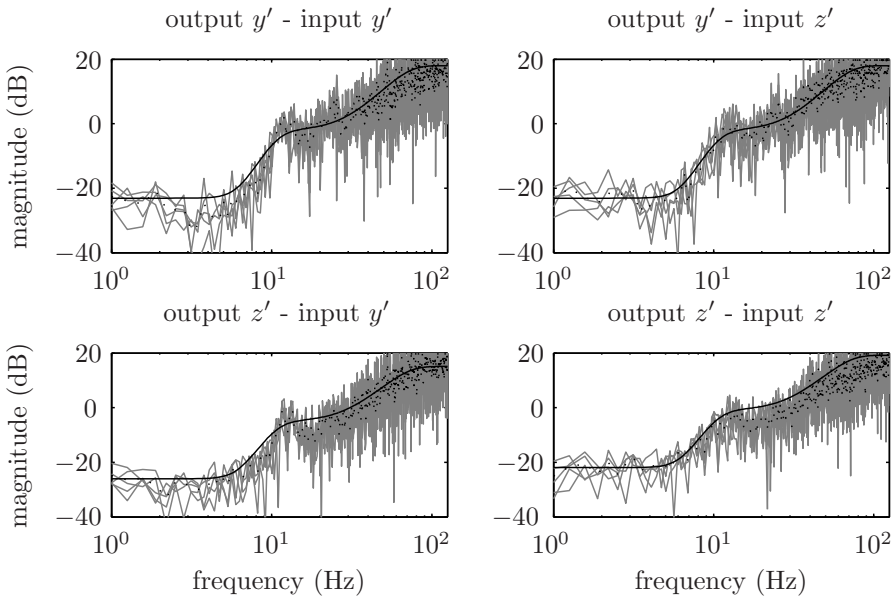
(b) Trajectory B

measured, averaged, fitted

Figure 5.30: Transfer function of the additive model error of model 2



(a) Trajectory A



(b) Trajectory B

measured, averaged, fitted

Figure 5.31: Transfer function of the additive model error of model 3

response of the robot system at the bandwidth (see subsection 5.3.5). The model uncertainty of model 2 peaks only for trajectory A, because the time-invariant model does not describe the large variation of the system's bandwidth along this trajectory. The model uncertainty of model 3 increases at the bandwidth, but it does not peak. The uncertainty of all models is large at high frequencies. This is partly the result of the limited order of the models, by which they cannot describe the high-order dynamic behaviour of the robot system at high-frequencies resulting from the mechanical resonance frequencies. On the other hand, the large uncertainty at high frequencies is the result of the effect of stochastic (iteration-varying) noise on the measurement of the tracking error. The amplitudes of the high-frequency components of the multisine excitation are small and the resulting high-frequency components of the output are small as well. The small effect of the excitation on the output is outmeasured by the effect of stochastic noise, resulting in a relatively large simulation error, which is used for the estimation of the uncertainty. The stochastic noise is not part of the system dynamics and thus the estimation procedure results in a conservative specification of the additive model error and thus the model uncertainty at high-frequencies.

Selection of the uncertainty weighting filters

The selection of the uncertainty weighting filters is based on the previously described estimate of the model uncertainty and the guidelines given in subsection 4.4.3.

In line with those guidelines a dynamic pre-weighting filter $\boldsymbol{M}$ and a static post-weighting filter $\boldsymbol{N}$ are taken. The dynamics of the pre-weighting filter are obtained from the estimate of the model uncertainty.

The uncertainty in the modelled relations between the two directions of the feedforward and the two directions of the error are assumed to be independent. This assumption is accounted for by taking a filter pre-weighting filter $\boldsymbol{M}$ with $N_f = 2$ inputs and $N_f \times N_e = 4$ outputs, where the effect of each of the feedforward components on each of the error components is an output of $\boldsymbol{M}$, and taking a post-weighting filter $\boldsymbol{N}$ with $N_f \times N_e = 4$ inputs and $N_e = 2$ outputs, which adds the effect of all feedforward components on each of the error components. The dimension of the input and the output of the normalised uncertainty matrix is thus $N_f \times N_e = 4$.

The uncertainty weighting filters are scaled as in equation (4.2) such that the spectral norm of the post-weighting filter $\boldsymbol{N}$ is β. The value of the weight ratio β can be used to tune the performance of RILC.

Discussion

The selected uncertainty weighting matrices result in a conservative specification of the uncertainty. Firstly, this is the result of using the difference between the measured and the simulated error for the estimation of the model uncertainty,

while this difference is not only the result of the model error, but also the effect of stochastic noise. Secondly, the modelled relations between the two directions of the feedforward and the two directions of the error are assumed to be independent, while probably some dependency is present in the components of the additive model error. Besides its conservativeness, the specified model uncertainty is time-invariant, while the additive error in the model of the non-linear robot dynamics probably varies along the trajectory, which could lead to an incorrect specification of the model uncertainty. It is recommended to address the aforementioned issues in future research to obtain a better specification of the uncertainty in the model of an industrial robot.

Chapter 6

Experimental results

This chapter presents the experimental results from the application of the ILC algorithms from chapters 3 and 4 to the Stäubli RX90 robot, which is described in chapter 5. These experimental results demonstrate the performance of the developed ILC algorithms in relation to the requirements formulated in section 1.2.

Section 6.1 describes the experimental procedure. The selection of the parameters of the ILC algorithms for the experiments is discussed in section 6.2. The results from the experiments are presented in section 6.3 and discussed in relation to the objectives of the thesis in section 6.4. Finally, in section 6.5, the effect of the improved tracking accuracy on the weld quality is demonstrated.

6.1 Experimental procedure

The main objective formulated in section 1.2 is the development of ILC algorithms for realising high-accuracy motion at the tip of an industrial robot. The ability of the ILC algorithms developed in chapters 3 and 4 to meet this requirement is tested experimentally by reducing the tracking error at the tip of the Stäubli RX90 robot, which is measured by the sensor described in subsection 5.1.3. This tracking error should be reduced to 0.1 mm to enable the use of this robot for laser welding. The reduction of the tracking error is tested for the two the trajectories described in section 5.2. Trajectory A is specially designed to show the performance that can be realised with the proposed ILC algorithms. Trajectory B is typical for laser welding tasks in industry. In section 1.2 it is argued that an ILC algorithm should be able to cope with configuration dependent dynamics to reduce the high-frequency components of the tracking error at the robot tip. In this work the non-linear configuration dependent dynamics are approximated as LTV for the small deviations from the repetitively traced trajectory and ILC algorithms for LTV dynamics are developed. The necessity for being able to cope with the configuration dependent dynamics is demonstrated

by comparing the reduction of the tracking error by ILC for the LTI models 1 and 2 and the LTV model 3 (see section 5.3).

The reduction of the tracking error should be realised under several constraints resulting from practical considerations. Those constraints result in several additional requirements on the ILC algorithm as discussed in section 1.2. Firstly, the tracking error should converge monotonically in a limited number of iterations. From the convergence analyses in sections 3.4 and 4.4 it is concluded that the convergence rate depends on the selection of the parameters for the ILC algorithms. This dependence is demonstrated by repeating the experiments for several different parameter settings. The selection of the parameters is discussed in more detail in section 6.2. The actual convergence rate is demonstrated by the experimental results presented in section 6.3. Secondly, the ILC algorithm should be computationally efficient. The computational efficiency is demonstrated by a record of the computation time and memory that is required by the algorithms for the experiments. Thirdly, the ILC algorithms should not introduce any feedback action or require a modification of the standard industrial feedback controller. This condition is satisfied as the proposed ILC algorithms do not use measurements of the tracking error from the current iteration for the computation of the feedforward and the ILC algorithms update the position setpoints for the controller such that the standard motion controllers of the industrial CS8 controller can be used (see subsection 5.1.4).

The ILC algorithms are applied according to the following procedure:

1. The nominal setpoints for the position and velocity of the robot joints are computed as described in section 5.2.

2. The model of the dynamics of the robot system is constructed as described in section 5.3. In addition, the uncertainty in the dynamic model is specified according to the procedure described in subsection 5.3.6 for the implementation of RILC.

3. The robustness filter and the parameters of the ILC algorithms are selected as described in section 6.2.

4. The non-stationary Riccati difference equation is solved. The Riccati difference equation is the first step of the computation scheme for NILC described in section B.1 and the second step of the computation scheme for RILC described in section B.2.

5. The robot controller is commanded to move the robot along the trajectory setpoints and to record the tracking error measured by the seam-tracking sensor.

6. The trajectory setpoints are updated by the ILC algorithm as described in subsection 5.1.4. The NILC algorithm from chapter 3 is implemented as described in subsection 3.3.2 using the algorithm from section B.1. The

RILC algorithm from chapter 4 is implemented as described in subsection 4.3.2 using the algorithm from section B.2.

7. Steps 5 and 6 are repeated 10 times.

For the evaluation of the computational efficiency of the algorithms, the time to compute the time-varying Riccati matrix and the memory to store this matrix are recorded in step 4 and the time to compute the feedforward update is recorded in step 6. The procedure is repeated for trajectories A and B, models 1, 2 and 3 and for the parameter sets defined in section 6.2. In practice, the iterative procedure could be stopped if the tracking error drops below a certain threshold and it is desirable that the tracking error reaches this threshold in less than 10 iterations. Nevertheless, the iterative procedure is repeated for 10 iterations in this chapter to show that the tracking error does not diverge when continuing iterations.

6.2 Parameter selection

This section describes the selection of the robustness filter and the parameters for the NILC and RILC algorithms as used for the experiments.

6.2.1 Norm-optimal ILC

The NILC algorithm from chapter 3 computes the feedforward update that minimises an objective function that is a weighted sum of the norm of the error estimate and the norm of the feedforward update. The selection of the weights in the objective function is discussed in this subsection. Moreover, the selection of the robustness filter is discussed.

Selection of the weights

The errors in the y' and the z'-direction are considered equally important along the whole trajectory and thus $V_i = I$ is taken. Similarly, there is no reason to vary the weight on the components of the feedforward update along the trajectory and thus $W_i = w^2 I$ is taken, where parameter w determines the relative contribution of the feedforward update to the objective function.

Guidelines for the selection of w are given in subsection 3.4.4. The convergence rate can be increased by decreasing the value of w, though this also decreases the allowable model error. The effect of w on the convergence is demonstrated by testing the performance of NILC for three values of w^2, viz., $w^2 = \{\frac{1}{3}, 1, 2\}$. At low frequencies the dynamics of the robot system are square, modelled accurately and the gain is unity (see section 5.3). Under these conditions the convergence of the low-frequency components of the error can be decoupled according to the decoupled convergence analysis in subsection 3.4.3.

The convergence ratio of the low-frequency components of the error can be derived from equation (3.64) and is $\{\frac{1}{4}, \frac{1}{2}, \frac{2}{3}\}$ for $w^2 = \{\frac{1}{3}, 1, 2\}$ respectively.

Selection of the robustness filter

Guidelines for the selection of the robustness filter for NILC are also given in subsection 3.4.4. The robustness filter should be zero for the components of the error corresponding to a large model error to obtain convergence. On the other hand, the robustness filter should be unity for the other components of the error to reduce these error components to zero. In subsection 5.3.6 it is shown that the model error is largest at high frequencies. The robustness filter should thus be close to zero at high frequencies to obtain convergence, but the filter should be close to unity at low frequencies to reduce these components of the error to zero.

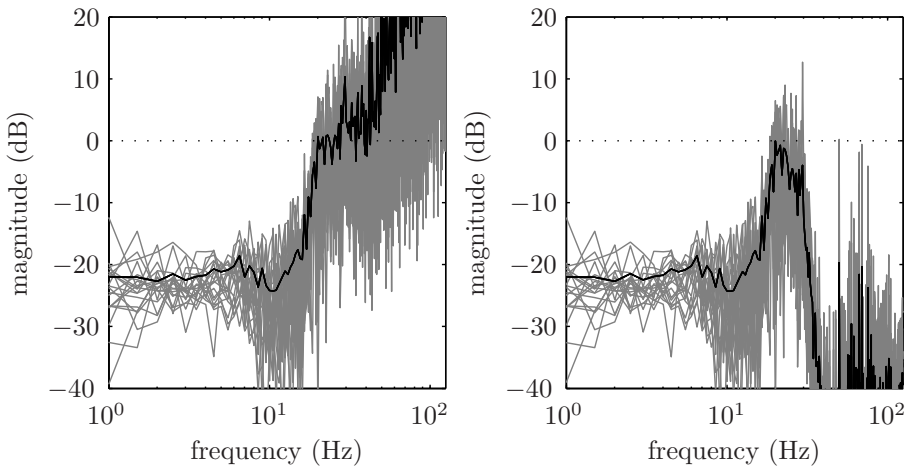
The desired behaviour for the robustness filter is realised by using an zero-phase eighth-order low-pass Butterworth filter. The cutoff frequency of the filter separates the low-frequency and the high-frequency region. The high order of the filter makes the gain change from about unity to about zero in a small frequency band and no phase-distortion is introduced at low-frequencies due to the zero-phase behaviour. The zero-phase eight-order low-pass Butterworth filter is realised by filtering the feedforward with a conventional fourth-order low-pass Butterworth filter and filtering the output of this filter with the transpose of the same filter. The transpose of the filter has the same gain, but opposite phase. Thus, filtering with the transpose doubles the attenuation of high frequencies and compensates for the phase shift. The transpose of the filter is anti-causal and implemented by filtering backward in time. The initial state of the causal part of the robustness filter is taken equal to the steady-state response of the filter's states to the first sample of its input. This way, the initial output of the filter is equal to the initial input instead of being zero. Similarly, the final state of the anti-causal part of the robustness filter is taken equal to the steady-state response of the filter's state to its final input. This way, the final output of the filter is equal to the final input instead of being zero.

The cutoff frequency of the robustness filter is selected such that the feedforward converges. The condition for convergence of the feedforward is expressed by inequality (3.38). This inequality cannot be checked mathematically, since the lifted system matrix $\boldsymbol{G}$ is not known exactly. Instead, the convergence condition could be checked, by verifying if $\|\boldsymbol{Q}(\boldsymbol{I} - \boldsymbol{L}\boldsymbol{G})\boldsymbol{u}\|_2 / \|\boldsymbol{u}\|_2 < 1$ for any vector $\boldsymbol{u}$, where $\boldsymbol{G}\boldsymbol{u}$ is obtained experimentally. Checking this convergence condition for a complete set of independent vectors $\boldsymbol{u}$ requires $N_i \times N_f$ experiments, which is very time consuming for long iterations. Only a single experiment is needed to check the convergence of the feedforward if it is known which components converge independently. For LTI systems and (infinitely) long iteration, the frequency components of the feedforward converge independently (Dijkstra, 2004). The same approximately holds for the robot system of which the frequency re-

sponse changes only slowly along the trajectories. Therefore, the convergence condition is checked for each of the frequency components of a set of broadband multisine excitations. The need for additional experiments is obviated by using the experimental data that is also used for the estimation of the parameters of models 2-3 and the model uncertainty (see section 5.3).

It is thus demanded that each of the frequency components of the broadband multisine excitation converges. This is verified by dividing power spectrum of $\boldsymbol{Q}(\boldsymbol{I}-\boldsymbol{LG})\boldsymbol{u}$ by the power spectrum of $\boldsymbol{u}$, where $\boldsymbol{Gu}$ is obtained experimentally. The resulting amplification of the power spectrum is averaged over the measurement series and the result is maximised over the two feedforward input components. As an example, figure 6.1(a) shows the amplification of the power spectrum for model 3, trajectory A, $w^2 = \frac{1}{3}$ and $\boldsymbol{Q} = \boldsymbol{I}$. The displayed amplification of the power spectrum is larger than 0 dB at high frequencies, which means that application of NILC without a robustness filter probably results in divergence of the high-frequency components of the feedforward. This confirms the necessity for a low-pass robustness filter. The previously described eighth order zero-phase low-pass Butterworth filter is applied. The highest cutoff frequency for which the averaged power transfer function does not exceed 0 dB is taken, which is referred to as the maximum allowable cutoff frequency. The maximum allowable cutoff frequency is computed with a bisection algorithm. The bisection algorithm starts with a lower frequency bound of 0 Hz and an upper frequency bound of 125 Hz (the Nyquist frequency) and stops when the difference between the lower and upper bound is less than 1 Hz. For model 3, trajectory A and $w^2 = \frac{1}{3}$ the resulting cutoff frequency is 25 Hz. Figure 6.1(b) shows the convergence of the power spectrum of the feedforward for this maximum allowable cutoff frequency. The maximum allowable cutoff frequencies for the other models, trajectories and values of w are listed in table 6.1. The table shows that the allowable cutoff frequency increases with the value of w, which is in line with the convergence analysis in section 3.4. Moreover, the allowable cutoff frequency is least for model 1, which is not able to describe the dynamics of the robot system beyond the bandwidth, and highest for model 3, which is able to describe the variation of the (high-frequency) robot dynamics along the trajectories.

The maximum allowable cutoff frequencies listed in table 6.1 are used for the implementation of the robustness filter for the experiments of which the results are described in section 6.3. The convergence analysis in subsection 3.4.4 concludes that the robustness filter not only affects the convergence but also the final error. The effect of the selected cutoff frequencies on the final error is analysed in subsection 6.2.3.



(a) No robustness filter

(b) The robustness filter with the maximum allowable cutoff frequency

Figure 6.1: Convergence of the power spectrum of the feedforward for a series of multisine feedforwards for NILC, model 3, trajectory A, and $w^2 = \frac{1}{3}$

trajectory	model	$w^2 =$	$\frac{1}{3}$	1	2
A	1		11.8	12.9	15.2
A	2		15.2	16.9	21.4
A	3		28.2	31.8	32.5
B	1		10.4	11.6	14.0
B	2		21.4	23.4	24.2
B	3		23.4	24.2	25.0

Table 6.1: Maximum allowable cutoff frequencies for NILC in Hz

6.2.2 Robust ILC

The RILC algorithm from chapter 4 computes the learning filter that minimises an objective function, which is defined such that robust convergence with convergence ratio γ is guaranteed if the objective function is negative definite for the optimal learning filter and the worst case effect of the specified model uncertainty. This condition for convergence is referred to as the sufficient condition for robust convergence (SCRC). The model uncertainty is specified by the uncertainty weighting filters. These selection of the uncertainty weighting filters and the maximum convergence ratio γ is discussed in this chapter. Moreover, the selection of the robustness filter is discussed.

Selection of the maximum convergence ratio

Guidelines for the selection of the maximum convergence ratio γ are given in subsection 4.4.3. The maximum convergence ratio γ specifies the maximum convergence ratio for the summed error if the SCRC is satisfied. Thus, decreasing the value of γ, decreases the convergence rate. However, decreasing the value of γ could also decrease the model error for which the SCRC is satisfied, which is exemplified by the decoupled analysis in subsection 4.4.2. The effect of γ on the convergence is demonstrated by testing the performance of RILC for three values of γ, viz., $\gamma = \{0.50, 0.75, 0.99\}$

Selection of the uncertainty weighting filters

Guidelines for the selection of the uncertainty weighting filters are also given in subsection 4.4.3. The uncertainty weighting matrices are selected in line with those guidelines as described in subsection 5.3.6. A dynamic pre-weighting filter $\boldsymbol{M}$ and a static post-weighting filter $\boldsymbol{N}$ are selected. The dynamics of the pre-weighting filter are based on an estimation of the model uncertainty. The tuning parameter that remains is the weight ratio β that scales the relative size of the weighting filters and specifies the 2-norm of the post-weighting filter. According to the guidelines given in subsection 4.4.3, the value of β^2 should be less than γ^2 to satisfy the condition for the existence of an optimal learning filter. Furthermore, taking β only slightly smaller than γ increases the model uncertainty for which the SCRC is satisfied if $\boldsymbol{R} = \boldsymbol{I}$. The effect of β on the convergence of the error is demonstrated by testing the performance of RILC for three values of β, viz., $\beta = \{0.20\gamma, 0.50\gamma, 0.99\gamma\}$.

Selection of the robustness filter

Guidelines for the selection of the robustness filter for RILC are also given in subsection 4.4.3. The robustness filter should be zero for the components of the error corresponding to a large model uncertainty to obtain convergence. On the other hand, the robustness filter should be unity for the other components of the

t	m	$\gamma =$	0.50	0.75	0.99	0.50	0.75	0.99	0.50	0.75	0.99
		$\frac{\beta}{\gamma} =$	0.20	0.20	0.20	0.50	0.50	0.50	0.99	0.99	0.99
A	1		-	3.9	12.7	-	6.8	10.7	4.9	6.8	8.8
A	2		-	-	17.6	-	9.8	15.6	6.8	9.8	12.7
A	3		-	-	24.4	-	11.7	21.5	7.8	14.6	16.6
B	1		-	-	12.7	-	6.8	10.7	4.9	6.8	8.8
B	2		-	-	20.5	-	8.8	17.6	6.8	10.7	14.6
B	3		-	10.7	20.5	-	10.7	17.6	7.8	10.7	14.6

t: trajectory, m: model

Table 6.2: Maximum allowable cutoff frequencies for RILC in Hz

error to reduce these error components to zero. Note that the same guidelines are used for the selection of the robustness filter for NILC (see subsection 6.2.1). Because the model uncertainty is largest at high frequencies, a zero-phase eighth-order low-pass Butterworth filter is selected for RILC as well.

The cutoff frequency of the robustness filter for RILC is selected using the same bisection algorithm as used for the selection of the cutoff frequency for NILC, though a different criterion is used. The bisection algorithm is used to find the highest cutoff frequency for which the SCRC is satisfied. The resulting cutoff frequency is referred to as the maximum allowable cutoff frequency. The computationally efficient procedure from subsection 4.3.2 is used for checking the SCRC. The resulting maximum-allowable cutoff frequencies for all trajectories, models and values of parameters γ and β are listed in table 6.2. In some cases no cutoff frequency could be found for which the SCRC is satisfied. Table 6.2 shows that the allowable cutoff frequency is least for model 1, which is not able to describe the dynamics of the robot system beyond the bandwidth. The allowable cutoff frequency for trajectory B is highest for models 2 and 3, which are both able to describe the robot's high-frequency dynamics. The allowable cutoff frequency for trajectory A is highest for model 3, which is able to describe the considerable variation of the robot dynamics along this trajectory. Moreover, the table shows that the allowable cutoff frequency increases for an increasing value of γ, which is in line with the convergence analysis in section 4.4. The relation between the allowable cutoff frequency and β depends on the value of γ; the allowable cutoff frequency decreases with an increasing value of β for $\gamma = 0.99$, while the opposite is true for $\gamma = 0.75$ and $\gamma = 0.50$.

The maximum allowable cutoff frequencies listed in table 6.2 are used for the implementation of the robustness filter for the experiments of which the results are described in section 6.3. The convergence analysis in subsection 4.4.3 concludes that the robustness filter not only affects the convergence but also the final error. The effect of the selected cutoff frequencies on the final error is analysed in subsection 6.2.3.

6.2.3 Prediction of the final error

In the previous subsections the selection of the robustness filters for NILC and RILC is discussed. In both cases a low-pass filter is selected and the selection of the cutoff frequency is based on a convergence analysis. Below the cutoff frequency the gain of the low-pass robustness filter is approximately unity and thus the low-frequency components of the error are compensated (see the convergence analyses in sections 3.4 and 4.4). Beyond the cutoff frequency the gain of the low-pass robustness filters is approximately zero and thus the high-frequency components of the error are not compensated (see the convergence analyses in sections 3.4 and 4.4). The final error can thus be predicted beforehand from the frequency components of the nominal tracking error beyond the cutoff frequency.

Thus, the robustness filter is not only needed for convergence, it also determines the final error. In this work the robustness filter is selected such that it results in convergence. An alternative approach would be to select (the cutoff frequency of) the robustness filter based on the required final error. The resulting robustness filter then dictates the required model accuracy and the parameter settings for the ILC algorithm to realise convergence.

The nominal tracking error along trajectory A is shown in figure 5.13 and the nominal tracking error along trajectory B in figure 5.20. The final tracking error is predicted by filtering the nominal tracking error with an eighth order zero-phase high-pass Butterworth filter. The resulting MAX and RMS final tracking errors as a function of the cutoff frequency of the filter are shown in figures 6.2(a) and 6.2(b). The desired MAX final error is 0.1 mm. Figure 6.2(a) shows that reducing the MAX error along trajectory A to 0.1 mm in both directions requires compensation of the frequency components of the error up to 30 Hz. This frequency is beyond the first resonance frequency of the robot mechanism, which is around 20 Hz. Considering the cutoff frequencies for NILC listed in table 6.1, the required accuracy can only be realised with NILC using model 3 and $w^2 = \{1, 2\}$. Considering the cutoff frequencies for RILC listed in table 6.2, the required accuracy cannot be realised by the application of RILC. The highest cut-off frequency for RILC is 24.4 Hz, which is realised for model 3, $\gamma = 0.99$ and $\beta = 0.20\gamma$. Figure 6.2(b) shows that reducing the MAX error along trajectory B to 0.1 mm in both directions requires compensation of the frequency components of the error up to 13 Hz. Considering the cutoff frequencies for NILC listed in table 6.1, this accuracy can be realised with NILC for all models and a sufficiently large value for w. Considering the cutoff frequencies for RILC listed in table 6.2, the required accuracy can be realised for models 2-3 using $\gamma = 0.99$.

So far, the effect of iteration-varying disturbances on the error has not been taken into account for the prediction of the final error. The effect is also not taken into account for the model-based prediction of the error in the current iteration that is reduced by the proposed ILC algorithms. As a result, the ILC

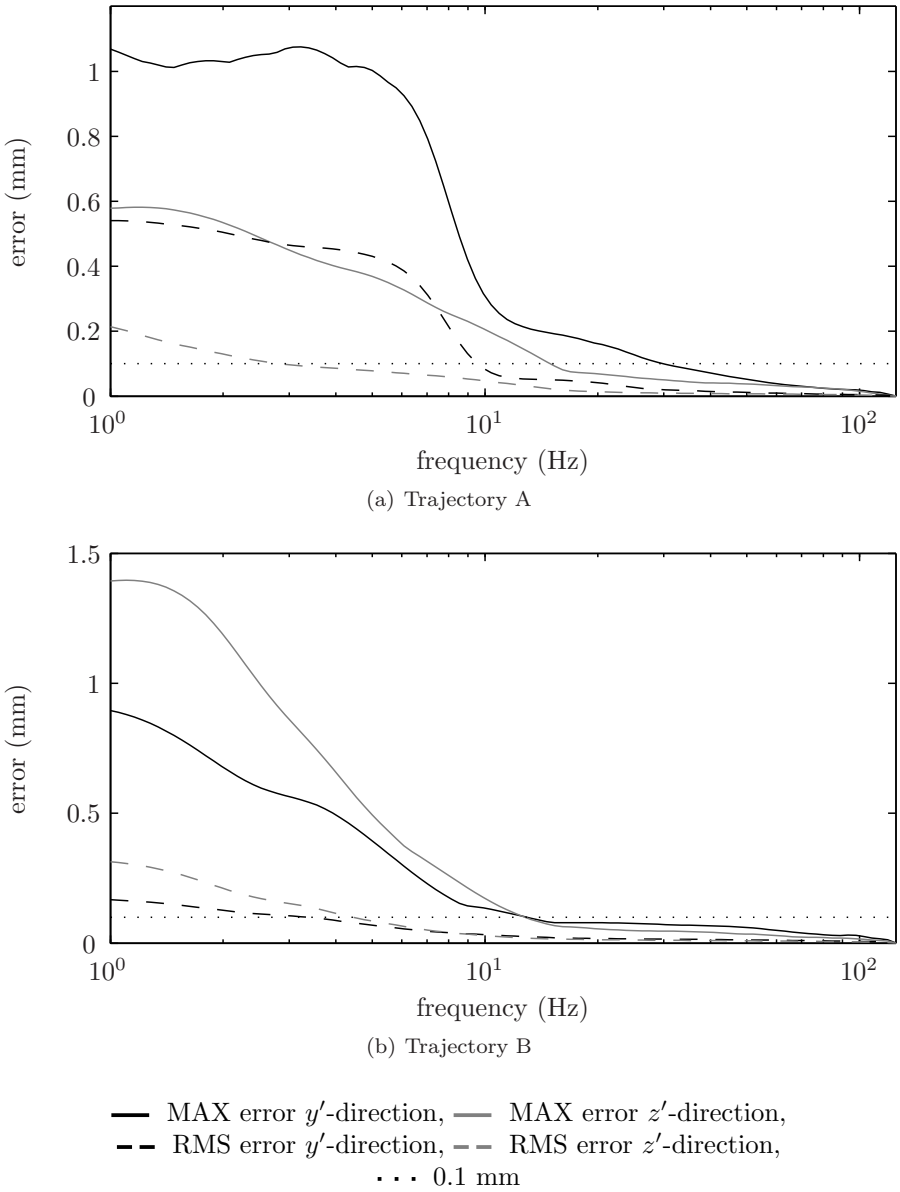
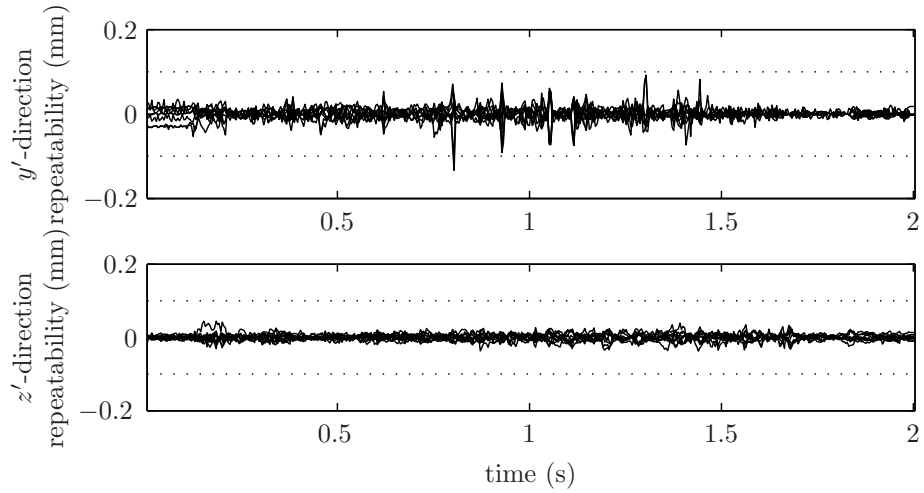
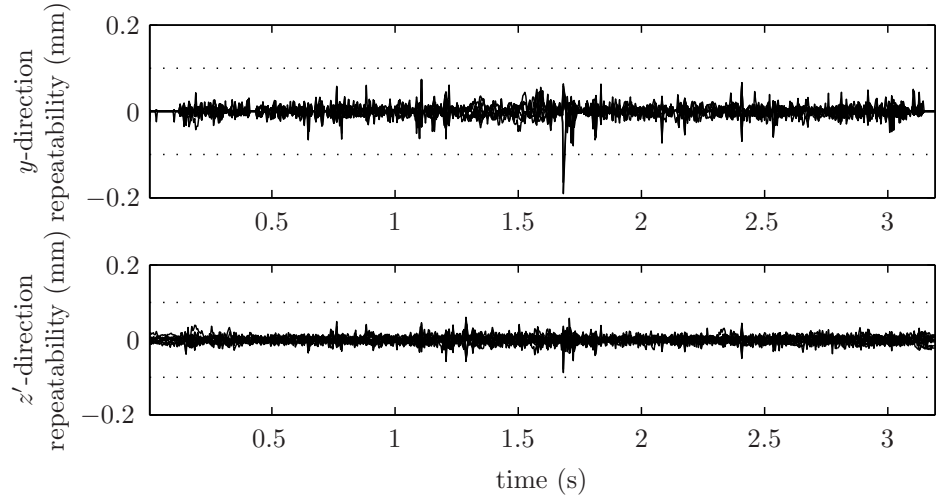


Figure 6.2: Prediction of the MAX and RMS final tracking error as a function of the cutoff frequency of the robustness filter



(a) Trajectory A



(b) Trajectory B

Figure 6.3: The repeatability of the measured tracking error for 10 repetitive motions along the trajectories

algorithms do not counteract the iteration-varying disturbances and the reduction of the tracking error by ILC is limited by the effect of iteration-varying disturbances. The effect of iteration-varying disturbances on the tracking error is measured by moving 10 times along the nominal trajectory at nominal speed. Figure 6.3 shows the deviation of the measured tracking error in each repetition from the mean of the tracking errors in the 10 repetitions. The measured repeatability of the tracking error is similar for trajectories A and B. Along most of these trajectories the level of repeatability is better than 0.04 mm. This level of repeatability is the result of the repeatability of the robot motion and the resolution of the sensor (see subsections 5.1.1 and 5.1.3). Occasionally the repeatability peaks about 0.1 mm in the y'-direction. These peaks are caused by the image-processing algorithm of the sensor software that sometimes misinterprets the location of the seam from the camera-image. The ILC algorithms are probably unable to compensate the tracking error below the measured level of repeatability.

6.3 Experimental results

This section presents the experimental results from the application of the proposed NILC and RILC algorithms to the Stäubli RX90 robot.

6.3.1 Norm-optimal ILC

NILC is applied to the Stäubli RX90 robot in line with the experimental procedure described in section 6.1. The experimental procedure is repeated for trajectories A and B (see section 5.2), models 1-3 (see section 5.3) and $w^2 = \{\frac{1}{3}, 1, 2\}$ (see subsection 6.2.1). The robustness filter is implemented as described in subsection 6.2.1.

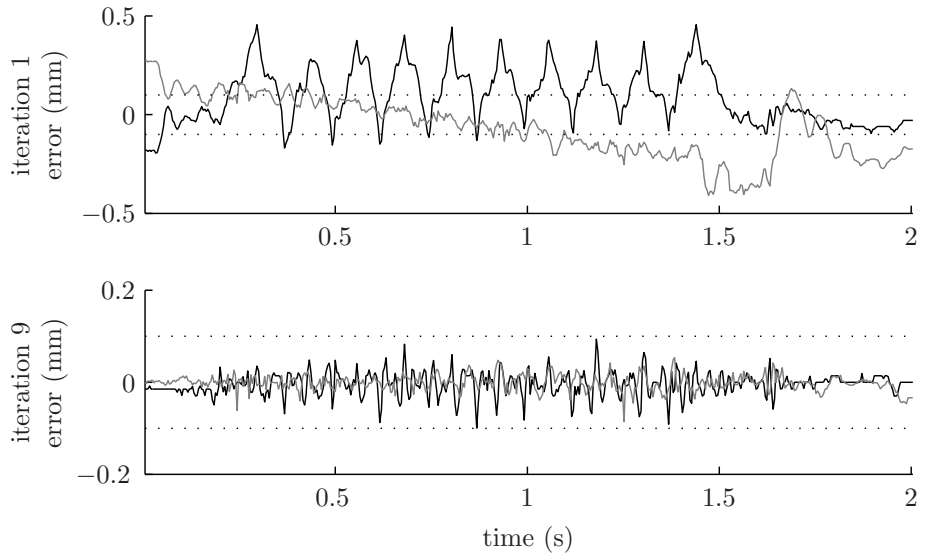
Figures C.1 and C.5 in appendix C show the MAX and RMS error in the 10 iterations of each experiment. The figures show that the MAX and RMS errors converge for all experiments, which means that the cutoff frequency for the robustness filter is selected sufficiently small. Moreover, it is shown that a small value of weight w results in fast convergence of the MAX and RMS error. This is in line with the conclusions of the convergence analysis in section 3.4. The tracking error converges to its final value in about 4 iterations for $w^2 = \frac{1}{3}$, while the error is not fully converged in 10 iterations for $w^2 = 2$.

Furthermore, the results in figures C.1 and C.5 show that the MAX final error is small for a large value of w or a model that is able to describe the configuration-dependent high-frequency dynamics of the robot system accurately. Comparison of MAX final error in the experiments and the cutoff frequencies listed table 6.1 shows that the MAX final error is smaller for a larger cutoff frequency. This agrees with the conclusion in subsection 6.2.3, that the MAX final error is smaller for a larger value of the cutoff frequency. Moreover, it is concluded in subsection 6.2.3 that the tracking error should be

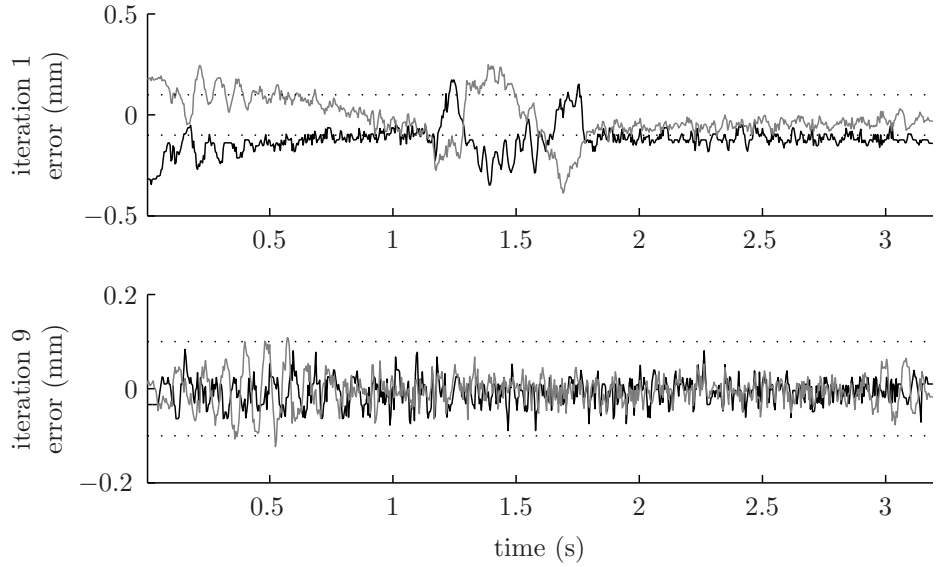
compensated up to 30 Hz to reduce the MAX tracking error to less than 0.1 mm for trajectory A, while the error should be compensated up to 13 Hz to achieve this goal for trajectory B. In most of the experiments where the cutoff frequency is beyond 30 Hz and 13 Hz for trajectories A and B respectively, the MAX final tracking error is indeed close to 0.1 mm (see figures C.1 and C.5). The MAX final tracking error for trajectory A, model 3 and $w^2 = 2$ and for trajectory B, models 1-2 and $w^2 = 2$ are a bit larger than 0.1 mm, which seems to contradict this conclusion, though the error is not fully converged in the 10 iterations for these experiments. It should be noted that the MAX final error is hardly smaller than 0.1 mm in any of the experiments. Probably this is the result of the limited repeatability of the measurement of the seam location (see subsection 6.2.3).

The desired MAX final tracking error is 0.1 mm. The MAX final error along trajectory B is approximately reduced to this value for all models and a sufficiently large value of w. The MAX final error along trajectory A is only reduced to this value using the LTV model 3, which is able to describe the configuration dependent dynamics of the robot. The tracking error along the trajectories converges fastest to the desired value using model 3 and $w^2 = \frac{1}{3}$. The tracking errors in iterations 1 and 9 of the corresponding experiments are shown in figure 6.4. The tracking errors in iterations 1 and 9 of the other experiments are shown in appendix C.

The NILC algorithm is implemented in MATLAB on the PC described in subsection 5.1.4 using the algorithm described subsection 3.3.2 and section B.1. The memory that is required to store the time-varying Riccati matrix is listed in table 6.3. The memory scales quadratically with the state-dimension and scales linearly with the time. The required memory is sufficiently small to allow implementation of the NILC algorithm on the used PC. The time to compute the time-varying Riccati matrix and the feedforward update are also listed in table 6.3. According to subsection 3.3.2 the number of computational operations of the algorithms scales linearly with the length of the iteration (if the state-dimension is constant). The ratio between the computation times for trajectories A and B are somewhat larger than the ratio between the number of time steps. Nevertheless, the computation times are sufficiently small to allow implementation of the NILC algorithm on the used PC. In particular, the computation time for updating the feedforward is less than the time that is needed by the robot to move along the trajectory. The feedforward for the next iteration can thus be computed while the robot moves back to the start of the trajectory at nominal speed.



(a) Trajectory A



(b) Trajectory B

y'-direction, z'-direction

Figure 6.4: Tracking error in iterations 1 and 9 of NILC for model 3 and $w^2 = \frac{1}{3}$

trajectory	model	N_i	$N_{\tilde{x}}$	memory to store S_i (MB)	time to compute S_i (s)	time to compute u_i (s)
A	1	499	4	0.064	0.2	0.5
A	2	499	18	1.293	3.2	0.8
A	3	499	18	1.293	3.4	0.9
B	1	796	4	0.102	0.4	1.2
B	2	796	18	2.063	7.4	1.7
B	3	796	18	2.063	7.6	1.6

Table 6.3: Computational efficiency of NILC

6.3.2 Robust ILC

RILC is applied to the Stäubli RX90 robot in line with the procedure described in section 6.1. The experimental procedure is repeated for trajectories A and B (see section 5.2), models 1-3 (see section 5.3), $\gamma = \{0.50, 0.75, 0.99\}$ and $\beta = \{0.20\gamma, 0.50\gamma, 0.99\gamma\}$ (see subsection 6.2.2). The robustness filter is implemented as described in subsection 6.2.2.

Figures C.2-C.4 and C.6-C.8 in appendix C show the MAX and RMS error in the 10 iterations of all experiments. The figures show that the MAX and RMS errors converge for all experiments, which means that the cutoff frequency for the robustness filter is selected sufficiently small (see subsection 6.2.2). Moreover, it is shown that a small value of γ results in fast convergence of the MAX and RMS error. This is in line with the conclusions of the convergence analysis in section 4.2. The MAX and RMS error converge in only 2 iterations for $\gamma = 0.50$, in 3-4 iterations for $\gamma = 0.75$, while the error is not even converged in 10 iterations for some experiments with $\gamma = 0.99$. Besides, the results in the figures show that the convergence ratio is significantly affected by the value of β. The larger the value of β, the faster the convergence. Even for $\gamma = 0.99$ the MAX and RMS error converge in only 2 iterations if $\beta = 0.99\gamma$.

Furthermore, the results in figures C.2-C.4 and C.6-C.8 show that the MAX final error is small for a large value of γ or a model that is able to describe the configuration-dependent high-frequency dynamics of the robot system accurately. Comparison of MAX final error in the experiments and the cutoff frequencies listed table 6.2 shows that in most cases the MAX final error is smaller for a larger cutoff frequency. This agrees with the conclusion in subsection 6.2.3, that the MAX final error is smaller for a larger value of the cutoff frequency. An exception to this conclusion is the MAX final error in iteration 9 for $\gamma = 0.99$, which decreases with the value of β, while the cutoff-frequency also decreases with the value of β. However, this is the result of the decrease of the convergence rate with the value of β, by which the error is not yet fully converged in the 10 iterations of the experiments if β is small and $\gamma = 0.99$.

It is concluded in subsection 6.2.3 that the tracking error should be compensated up to 30 Hz to reduce the MAX tracking error to less than 0.1 mm for trajectory A, while the error should be compensated up to 13 Hz to achieve this goal for trajectory B. The MAX final tracking error along trajectory B is indeed close to 0.1 mm if the cutoff frequency is beyond 13 Hz, which is the case for $\gamma = 0.99$ and model 2 or 3 (see figure C.6). In contrast to model 1, these models are able to describe the dynamics of the robot beyond its bandwidth. The MAX tracking error is also close to 0.1 mm for $\gamma = 0.75$ and model 2 or 3, because in those cases the cut-off frequency is 10.7, which is only a little less than the required 13 Hz. It should be noted that the MAX final error is hardly smaller than 0.1 mm in any of the experiments. Probably this is the result of the limited repeatability of the measurement of the seam location (see subsection 6.2.3). The MAX final error along trajectory A is not reduced to 0.1 mm by RILC in any of the experiments, because the cutoff frequency is less than 30 Hz for all models and values of γ and β. The smallest MAX final error realised by RILC along trajectory A, which is realised using model 3, $\gamma = 0.99$, $\beta = 0.99\gamma$, is 0.18 mm. In contrast to models 1 and 2, model 3 is able to describe the variation of the robot dynamics along trajectory A as a result of the large change of configuration.

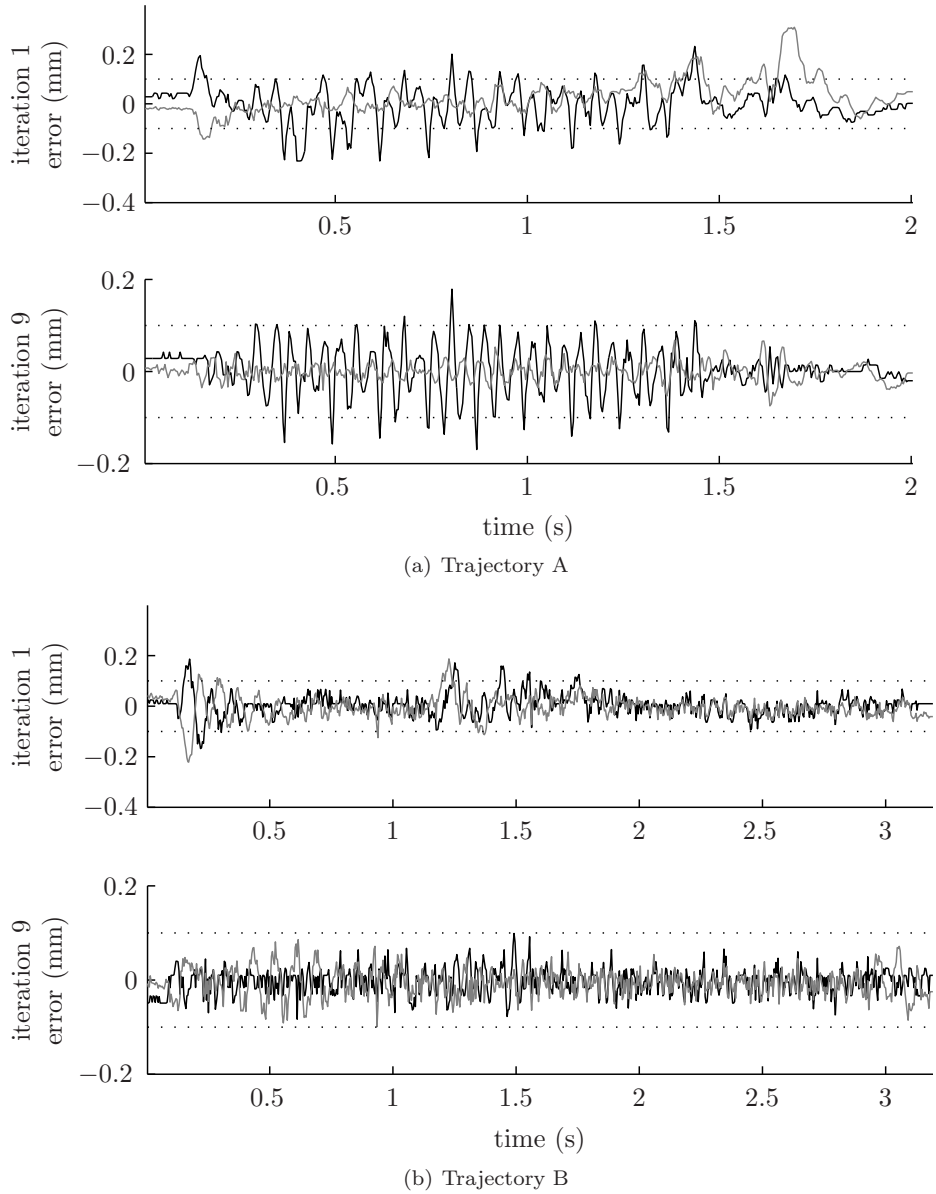
The smallest error along both trajectories is obtained for model 3, $\gamma = 0.99$ and $\beta = 0.99\gamma$. Moreover, the tracking error is already close to its final value in only 2 iterations. Thus, $\gamma = 0.99$ and $\beta = 0.99\gamma$ are settings that result in fast convergence to a small tracking error. The tracking errors in iterations 1 and 9 of the corresponding experiments are shown in figure 6.5. The tracking errors in iterations 1 and 9 of the other experiments are shown in appendix C.

The RILC algorithm is implemented in MATLAB on the PC described in subsection 5.1.4 using the algorithms described subsection 4.3.2 and section B.2. The memory that is required to store the time-varying Riccati matrix is listed in table 6.4. The memory scales quadratically with the state-dimension and linearly with time. The storage of the time-varying Riccati-matrix for RILC requires up to 20.21 Mb due to the large state-dimension of the uncertainty pre-weighting filter. Still, this memory is sufficiently small to allow implementation of RILC on the used PC. The time to compute the time-varying Riccati matrix and the feedforward update are also listed in table 6.4. According to subsection 4.3.2 the number of computational operations of the algorithms scales linearly with the length of the iteration. This cannot be verified by comparing the computation times for both trajectories due to the difference in the state dimension of the uncertainty pre-weighting filter for both trajectories. The time to compute the time-varying Riccati matrix for RILC requires up to 83.8 s, due to the large state-dimension. Still, the computation time is sufficiently small to allow implementation on the used PC. Note that the Riccati matrix needs to be computed only once for each experiment and not for each iteration. The time to compute the feedforward also allows implementation of the RILC algorithm on the used PC, but it is larger than the time that is needed to move the robot

trajectory	model	N_i	$N_{\tilde{x}}$	memory to store S_i (MB)	time to compute S_i (s)	time to compute u_i (s)
A	1	499	54	11.641	23.5	6.0
A	2	499	71	20.124	41.0	11.7
A	3	499	60	14.371	28.1	8.7
B	1	796	54	18.569	58.3	10.2
B	2	796	64	20.124	82.8	19.3
B	3	796	48	14.672	46.7	10.0

Table 6.4: Computational efficiency of RILC

along the trajectory. Thus, the feedforward update cannot be computed while the robot moves back at nominal speed. The time to compute the feedforward update can be reduced by using a faster PC or by the implementation of the RILC algorithm using a more efficient computation environment than MATLAB. The computation time can also be reduced by the selection of uncertainty weighting filters with a smaller state-dimension, which would also reduce the time and memory required for the Riccati matrix.



(a) Trajectory A

(b) Trajectory B

Figure 6.5: Tracking error in iterations 1 and 9 for model 3, $\gamma = 0.99$ and $\beta = 0.99\gamma$

6.4 Discussion

In this section the experimental results from section 6.3 are discussed in relation to the objectives of this thesis.

6.4.1 Accurate tracking

The experimental results show that the MAX final error depends on the size of the cutoff frequency of the robustness filter. The larger the value of the cutoff frequency, the lower the final error. The selection of the cutoff frequency, which is described in section 6.2, is based on convergence analysis. The cutoff frequency for NILC is based on the analysis of the convergence of the frequency components of the feedforward for a set of experimental data. The highest cutoff frequency for which all frequency components of the feedforward converge is taken. A higher cutoff frequency results in divergence of the high-frequency components of the feedforward. The selection of the cutoff frequency for RILC is based on the specification of the model uncertainty from subsection 5.3.6 and the sufficient condition for robust convergence (SCRC) formulated in section 4.2. The highest cutoff frequency for which the SCRC is satisfied is taken. The cutoff frequency for RILC is lower than the cutoff frequency for NILC, which can be explained by the following reasons. Firstly, the model uncertainty is specified conservatively as discussed in subsection 5.3.6. Secondly, the sufficient condition for convergence of RILC is more conservative than the condition for convergence of NILC as discussed in subsection 4.4.3. It is well possible that a higher cutoff frequency for RILC would still result in convergence. As a consequence of the lower cutoff frequency selected for RILC, the final error for RILC is larger than the final error for NILC.

The experimental results in section 6.3 show that the tracking error along trajectory B, which is typical for welding trajectories in industry, can be reduced to about 0.1 mm by the NILC algorithm and the RILC algorithm. Reducing the tracking error along trajectory B to 0.1 mm, requires compensation of the components of the tracking error at frequencies beyond the bandwidth of the robot system. In the experiments this is realised by NILC for all models and a sufficiently large value of w. The accuracy is only realised by RILC for models 2 and 3, which are able to describe the high-frequency dynamics of the robot. The tracking error along trajectory A, which contains considerable high-frequency components, can be reduced to 0.1 mm by the NILC algorithm. The RILC algorithm is only able to reduce the tracking error to 0.18 mm. The larger final tracking error for RILC is the result of the lower cutoff frequency. Reducing the tracking error along trajectory A to 0.1 mm requires compensation of the components of the tracking error at frequencies beyond the first resonance frequency of the robot mechanism. In the experiments this is only realised by NILC for model 3, which describes the high-frequency dynamics of the robot system and the variation of the dynamics along the trajectory.

Concluding, the tracking accuracy at the tip of the Stäubli RX90 robot, which is measured with a seam-tracking sensor, can be improved considerably by the proposed ILC algorithms. The tracking error can even be reduced close to the repeatability of the robot system. Thus, the proposed ILC algorithms are suited to realise high-accuracy tracking at the tip of an industrial robot. The high-accuracy is realised by reducing the components of the tracking error at high-frequencies, which requires a model of the robot that is able to describe the high-frequency robot dynamics. Furthermore, a model of the variation of the robot dynamics is required if the configuration of the robot changes considerable along the trajectory. The experimental results thus confirm the statement in section 1.2 that the ILC algorithm should be able to cope with varying dynamics to realise the high accuracy tracking.

6.4.2 Convergence rate

The experimental results in section 6.3 show that the MAX and RMS tracking error converge monotonically for the proposed ILC algorithms. The convergence rate depends on the parameter setting selected for the ILC algorithms.

According to the convergence analysis in section 3.4, the convergence rate for NILC is determined by the selection of the weights on the feedforward update and the error in the objective function. The relative weight on the input is specified by the value of w. Besides, as illustrated by the decoupled convergence analysis in subsection 3.4.3, the convergence rate depends on the system gain, though the system gain of the robot system does not deviate much from unity over the frequencies. The experimental results show that a small value of w indeed results in a high convergence rate. Moreover, it is shown that the effect of the value of w on the allowable cutoff frequency and thus the size of the final error is only small.

According to the convergence analysis in section 3.2, The convergence ratio for RILC is at most the selected value of γ. The experimental results show indeed that a small value for γ results in fast convergence. However, a small value of γ reduces the allowable cutoff frequency for the robustness filter considerably, which results in a larger final error. Besides the selection of γ, the selection of the uncertainty weighting filters influences the convergence rate of the error. The uncertainty weighting filters specify the model uncertainty and are selected as described in subsection 6.2.2. The relative size of the uncertainty weighting filters can be tuned by the selection of the weight ratio β, which is equal to the gain of the post-weighting filter $\boldsymbol{N}$. The value of the weight ratio should be less than γ to be able to compute an optimal feedforward update. The experimental results show that the selecting β only slightly smaller than γ results in fast convergence. Moreover, it is shown that the effect of the value of β on the allowable cutoff frequency and thus the size of the final error is only small. Realising monotonic convergence with a high convergence rate and a small final

error can thus be realised by selecting a maximum convergence ratio γ slightly smaller than 1 and selecting β slightly smaller than γ.

The experimental results thus show that, using an appropriate selection of the parameters, the MAX final tracking error can be reduced to its final level in only 3 iterations by NILC and in only 2 iterations by RILC. The number of iterations required to reach the final error with NILC can be reduced even further by selecting a smaller value for w. Concluding, the proposed algorithms result in monotonic convergence of the tracking error and a high-convergence rate can be realised by appropriate selection of the parameters of the ILC algorithms.

6.4.3 Computational efficiency

The computational efficiency of the algorithms is demonstrated by the record of the time to compute the time-varying Riccati matrix, the memory to store this matrix and the time to compute the feedforward update for the experiments. Those values are listed in table 6.3 for the NILC algorithm and in table 6.4 for the RILC algorithm.

The memory to store the time-varying Riccati matrix scales linearly with the length of the iteration and quadratically with the state-dimension. The time to compute the time-varying Riccati matrix and the time to compute the feedforward update also grow with the length of the iteration and the state-dimension. According to subsections 3.3.2 and 4.3.2 the number of computational operations for these computations scales linearly with the length of the iteration. The computation times required by the NILC algorithm are very short, the feedforward update can even be computed within the time the robot needs to move along the trajectory. The RILC is less efficient in terms of computation time and memory. This difference is caused by the larger state-dimension for RILC that is the result of the state-dimension of the uncertainty pre-weighting filter, which is considerably larger than the state-dimension of the system. Still, the memory and computation time required by the RILC algorithm are sufficiently small for implementation on the used PC, which is described in subsection 5.1.4. Concluding, the proposed NILC and RILC algorithms are sufficiently efficient to allow implementation on a contemporary PC.

Hereafter, the computational efficiency of the used NILC and RILC algorithms is compared to the computational efficiency of the lifted algorithms described subsections 3.3.1 and 4.3.1. These lifted algorithms are implemented in MATLAB on the same PC as the non-lifted algorithms. The memory to store the learning matrix $\boldsymbol{L}$, the time to compute this matrix, and the time to compute the feedforward update are listed in table 6.5 and 6.6.

The memory required to store the lifted learning matrix scales quadratically with the length of the iteration, but it is independent of the state-dimension and the type of algorithm. The memory required to store the learning matrix for the trajectories considered in the experiments is much larger than the memory re-

trajectory	model	N_i	$N_{\tilde{x}}$	memory to store $\boldsymbol{L}$ (MB)	time to compute $\boldsymbol{L}$ (s)	time to compute $\boldsymbol{u}$ (s)
A	1	499	4	7.968	5.0	0.01
A	2	499	18	7.968	5.2	0.01
A	3	499	18	7.968	5.2	0.01
B	1	796	4	20.276	20.2	0.03
B	2	796	18	20.276	20.7	0.03
B	3	796	18	20.276	20.8	0.03

Table 6.5: Computational efficiency of the lifted implementation of NILC

quired to store the time-varying Riccati-matrix for NILC, while it is comparable to the memory required to store the time-varying Riccati-matrix for RILC.

The time to compute the lifted learning matrix depends on the type of algorithm and the length of the iteration, but it is independent of the state-dimension. The time to compute the learning matrix approximately scales with the third power of the length of the iteration. For the trajectories considered in the experiments, the time to compute the learning matrix for NILC is much larger than the time required to compute the time-varying Riccati-matrix for NILC. For trajectory A, the time to compute the learning matrix for RILC is shorter than the time to compute the time-varying Riccati-matrix for RILC, while the computation times are comparable for trajectory B.

The time to compute the feedforward update using the lifted algorithms depends on the length of the iteration, but it is independent of the state-dimension and the type of algorithm. The time to compute the feedforward update by the lifted algorithms is much shorter than the time to compute the feedforward update by the non-lifted algorithms.

Summarising, the time to compute the feedforward update using the lifted algorithm is considerably shorter than the time required by the non-lifted algorithms and the time and memory required for the learning matrix allow implementation on the used PC. However, the computation time and memory required for the learning matrix grow with the second and third power of the iteration length respectively, which makes the computation of the learning matrix computationally intensive for long iterations.

trajectory	model	N_i	$N_{\tilde{x}}$	memory to store $\boldsymbol{L}$ (MB)	time to compute $\boldsymbol{L}$ (s)	time to compute $\boldsymbol{u}$ s)
A	1	499	54	7.968	13.4	0.01
A	2	499	71	7.968	14.1	0.02
A	3	499	60	7.968	13.7	0.01
B	1	796	54	20.276	53.5	0.03
B	2	796	64	20.276	55.0	0.04
B	3	796	48	20.276	55.0	0.03

Table 6.6: Computational efficiency of the lifted implementation of RILC

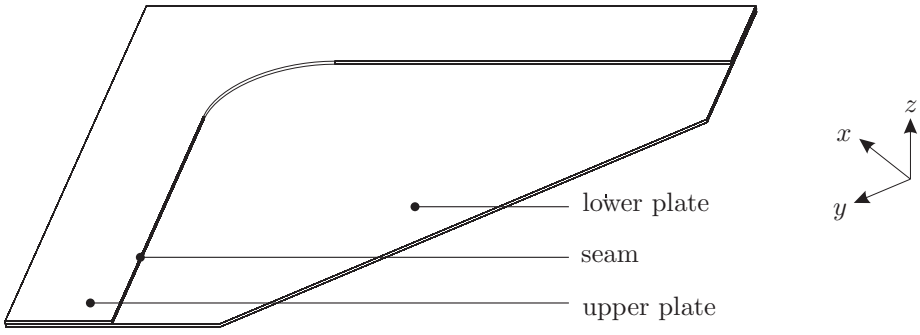
6.4.4 Summary

The application of NILC and RILC results in high-accuracy motion of the Stäubli RX90 robot. The tracking error of the robot, which is measured at its tip, can be reduced to less than 0.1 mm. Along trajectory B, which is typical laser welding trajectory in industry, the accuracy of 0.1 mm is realised by both algorithms using a model that is able to describe the dynamics of the robot at frequencies beyond the closed-loop bandwidth. Along trajectory A, which results in a considerable change of the robot configuration, the tracking accuracy of 0.1 mm is realised by the NILC algorithm using a model that is able to describe the variation of the dynamics along the trajectory. This demonstrates the value of an ILC algorithm that is able to cope with varying dynamics if the configuration of the robot changes considerably along the trajectory. RILC is able to reduce the maximum absolute tracking error along trajectory A to 0.18 mm. Further reduction cannot be realised by RILC because of the conservative specification of the model uncertainty. The proposed NILC and RILC algorithms result in monotonic convergence of the tracking error and the final tracking error can be realised in only 3 iterations using an appropriate selection of the tuning parameters. Moreover, the ILC algorithms do not add feedback action to the standard industrial CS8 controller and the computational efficiency of the algorithms is sufficient for implementation on a contemporary PC. It can thus be concluded from the experiments that all requirements imposed by the objectives of this thesis (see section 1.2) are satisfied. In section 6.5 it is shown that the improved tracking accuracy results in an improvement of the weld quality for laser welding.

6.5 Welding Results

In the previous section it is shown that ILC can improve the tracking accuracy of an industrial robot considerably. This section describes an experiment that shows the effect of the improvement of the tracking error on the weld quality. This effect is demonstrated by welding a seam using the nominal trajectory and the trajectory that is updated with ILC.

The seam is formed by the edge of a metal plate that overlaps a second plate. These plates are depicted in figure 6.6 and the dimensions are shown in figure 6.7. The geometry of the seam is similar to the geometry of trajectory B (see subsection 5.2.2). The Stäubli RX130 robot (Stäubli, 2001) is used to manipulate the welding head along the weld seam. This robot resembles the Stäubli RX90 robot, but its links are larger and its drives more powerful. The larger reach and the higher load capacity make this robot more suited for real welding tasks with a fully equipped welding head. The nominal trajectory is obtained by interpolating a cubic spline through points with an intermediate distance of 20 mm along the trajectory. The locations of those points are found using the seam-tracking sensor (see subsection 5.1.3). The x'-direction of the sensor is aligned with the local direction of the seam and the z'-direction is kept aligned with the global z-direction. The welding head thus rotates in the curved part of the trajectory. Keeping the welding head in a fixed orientation with respect to the weld seam facilitates the detection of the seam location by the seam tracking sensor and the unidirectional delivery of the shielding gas that protects the weld area from atmospheric gasses. The velocity profile is trapezoidal with a maximum velocity of 100 mm/s. The nominal tracking error is small along the straight part of the seam and considerably larger in the curved part of the trajectory as a result of the high joint velocities and



(the arrows indicate the directions of $\mathcal{O}_{xyz}$, not the origin)

Figure 6.6: The weld seam

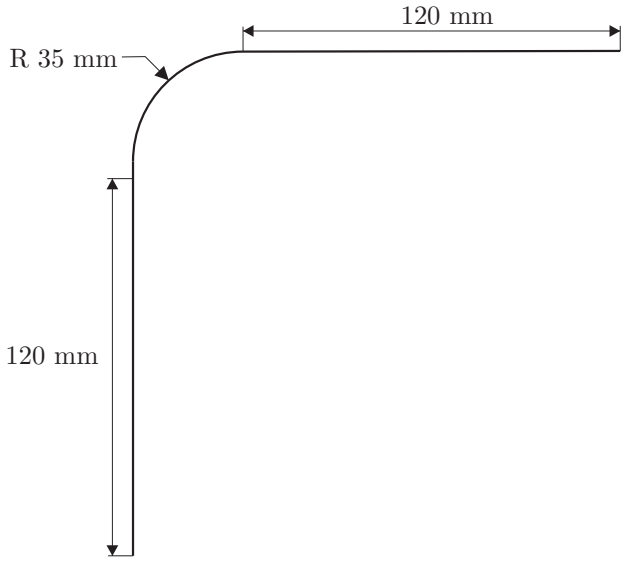
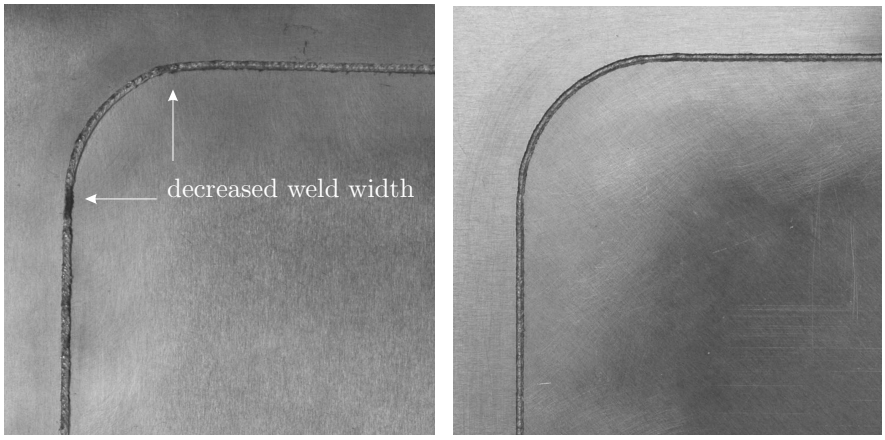


Figure 6.7: The geometry of the weld seam

accelerations in this part. The tracking error peaks at the start and the end of the curved part where the centripetal acceleration of the welding head starts and ends respectively. The tracking accuracy is improved by the application of NILC using model 1 (see section 5.3). After 10 iterations the MAX tracking error is less than 0.2 mm along the whole trajectory.

The welded plates are made of aluminium (AA5182) and the thickness is 1.1 mm. The laser power is set to 3000 W during welding. Two seams are welded, one seam using the nominal trajectory and the other using the trajectory updated with ILC. Figure 6.8 shows pictures of the resulting welds. The width of the weld that is traced using the nominal trajectory is irregular, in particular along the curved part of the seam. This is partly the result of insufficient wetting of the lower plate. The effect is most noticeable at the start and end points of the curved part. Moreover, inspection of the bottom side of the plate shows that the lower plate is not fully penetrated at those points. The parts of the seam with the degraded weld quality correspond to the parts of the trajectory that are not traced accurately. The weld quality is good and constant along the seam that is welded using the trajectory updated with ILC. The experiment thus shows the beneficial effect of the improved tracking accuracy realised with ILC on the weld quality.



(a) Nominal trajectory (b) Trajectory compensated with ILC

Figure 6.8: Welding results

Chapter 7

Conclusions and discussion

7.1 Conclusions

The aim of this thesis is the development of ILC algorithms for realising high-accuracy motion at the tip of an industrial robot. In section 1.2 the requirements on the ILC algorithm following from this objective are derived. It is observed that the high accuracy can only be realised by reducing the frequency components of the tracking error beyond the bandwidth of the feedback controller. Below the bandwidth the non-linear dynamics of the robot mechanism are linearised by the controller, but at higher frequencies the closed-loop dynamics depend on the configuration of the robot mechanism. The ILC algorithm should be able to cope with the configuration dependency of the dynamics to reduce the frequency components of the tracking error beyond the bandwidth of the feedback controller. In this work the non-linear configuration dependent dynamics are approximated as linear time-varying along a trajectory, resulting in the following requirement on the ILC algorithm:

- The ILC algorithm should be applicable to systems with linear time-varying dynamics.

In addition to this requirement, the following requirements are formulated to enhance the practical applicability :

- The ILC algorithm should result in monotonic convergence to a small final error,
- The ILC algorithm should result in a high convergence rate of the tracking error,
- The ILC algorithm should be computationally efficient,
- The ILC algorithm should be applicable to an industrial robot operating in closed-loop with its standard controller without adding feedback action.

No ILC algorithm that satisfies all these requirements is found from the literature review in chapter 2. Nevertheless, it is concluded from the review that model-based ILC is the most suitable approach to meet all requirements. Therefore, two model-based ILC algorithms are developed in this work. A norm-optimal ILC (NILC) algorithm is described in chapter 3 and a robust ILC (RILC) algorithm is described in chapter 4. The convergence properties of the algorithms are also analysed in chapters 3 and 4. The performance of the algorithms is tested experimentally by the application to an industrial robot, which is described in chapters 5. The experimental results are presented in chapter 6. In this chapter the conclusions from the previous chapters are summarised and related to the formulated requirements on the ILC algorithms. The conclusions resulting from the development of the algorithms and the convergence analyses are discussed in subsection 7.1.1 and the conclusions resulting from the experiments are discussed in subsection 7.1.2.

7.1.1 Conclusions from the developments and the analyses

Model based algorithms for LTV dynamics

The NILC algorithm developed in chapter 3 iteratively computes the feedforward that minimises the weighted sum of the norm of the feedforward update and the prediction of the error in the current iteration. The error is predicted from a model with LTV dynamics. The NILC algorithm is thus suited for the application to systems with LTV dynamics.

The RILC algorithm developed in chapter 4 iteratively applies the feedforward that minimises an objective function that is related to the growth of the sum of the error over the iterations. The growth of this sum is predicted from a model with LTV dynamics and a specified bounded uncertainty. Thus, the RILC algorithm is also suited for the application to systems with LTV dynamics.

Monotonic convergence to a small final error

The objective function for NILC (equation (3.10)) is formulated in terms of the 2-norm of the error and thus NILC aims at achieving monotonic convergence of the error. The convergence analysis in section 3.4 shows that NILC results in monotonic convergence of the error to zero if no robustness filter is used, if the system is nonsingular and if its dynamics are modelled perfectly or if the model error satisfies inequality (3.42). The error does not converge to zero if this condition is not satisfied, though it is always possible to realise monotonic convergence of the feedforward by selecting a robustness filter such that inequality (3.39) is satisfied. The robustness should filter out those feedforward components to which the system's response is not modelled accurately. However, the removal of these components of the feedforward typically results in a nonzero final error.

The objective function for RILC (equation (4.23)) is formulated in terms of the 2-norm of the sum of the error over the iterations. Minimisation of this objective function aims at achieving monotonic convergence of the error to zero. The objective function is formulated such that if it is negative for the minimising feedforward, the worst-case effect of the model uncertainty and any value of the summed error (inequality (4.24)), then this is a sufficient condition for monotonic convergence of the summed error. The error converges to zero if the convergence condition is satisfied without using a robustness filter. However, the convergence condition is not satisfied without using a robustness filter if the model uncertainty is large. On the other hand, it is possible to satisfy the convergence condition for any size of model uncertainty by selecting a suitable robustness filter, but the use of a robustness filter typically results in a nonzero final error.

Fast convergence

The analysis of the convergence properties of NILC in section 3.4 shows that the convergence rate depends on the selection of the weights in the objective function and the gain of the system. The convergence rate of the error increases by decreasing the weight on the feedforward update if the system is modelled perfectly. However, decreasing this weight also decreases the allowable size of the model error for which the convergence condition is satisfied.

The proposed design of RILC is such that the convergence ratio of the summed error is at least a specified value if the sufficient condition for robust convergence is satisfied. The demanded convergence rate can thus be set explicitly. However, the smaller the selected maximum convergence ratio, the smaller the size of the model uncertainty for which the convergence condition is satisfied.

Fast convergence versus a small final error

As concluded above, the convergence rate of the error can be increased by decreasing the weight on the feedforward update in the objective function for NILC, but this also reduces the allowable size of the model error. Similarly, decreasing the maximum convergence ratio for RILC decreases the allowable size of the model uncertainty. On the other hand, a robustness filter can be used to realise convergence with a certain convergence rate for any size of the model error or uncertainty at cost of a nonzero final error. The convergence rate can thus be increased at cost of a larger final error. The convergence rate and the final error can be reduced both if the model error or uncertainty is reduced.

Computationally efficient implementations

The computation of the feedforward update that minimises the objective for NILC is formulated as a finite-time optimal control problem (equations (3.18) in section 3.3).

The computation of the learning filter that minimises the objective for RILC is formulated as a finite-time dynamics game (equations (4.54) in section 4.3). Moreover, the check of the sufficient condition for convergence is rewritten to an anti-causal finite-time optimal control problem (equations (4.58) in section 4.3).

The formulated optimal control problems and the dynamic game can be solved efficiently using existing algorithms, which are described in appendix B. The computation time and memory required for the implementation of those algorithms scale linearly with the length of the iteration.

No additional feedback control

The NILC and RILC algorithms proposed in chapters 3 and 4 compute the feedforward that minimises an objective function related to the prediction of the error in the current iteration. This prediction is based on a model of the system dynamics and the measurement of the error in the previous iterations, such that the measurement of the error in the current iteration is not needed for the computation of the feedforward. The ILC algorithms thus do not use feedback of the error in the current iteration.

7.1.2 Conclusions from the experimental results

The performance of the proposed NILC and RILC algorithms in relation to requirements following from the objective of this thesis is tested experimentally by the application to a Stäubli RX90 robot. The tracking error of the robot is measured by an optical sensor mounted at the tip of the robot.

The Stäubli RX90 robot and the model of its dynamics are described in chapter 5. The dynamics of this robot are modelled with a time-varying ARX model structure. The performance of the ILC algorithms is tested for three models; model 1 assumes perfect tracking, model 2 assumes time-invariant dynamics and model 3 assumes slowly varying dynamics. The parameters of models 2 and 3 are estimated from system identification. The specification of the uncertainty in these models, which is required for the implementation of RILC, is derived from frequency analysis of the difference between the measured and the simulated dynamic response of the robot.

The experimental results are presented and discussed in chapter 6. Hereafter, the conclusions from this chapter are summarised and related to the requirements on the ILC algorithm formulated in section 1.2.

No change of the feedback control

ILC is used to update the setpoints for the position of the robot. This way, the integrator of the industrial CS8 controller of the Stäubli RX90 robot does not counteract the low-frequency part of the output of the ILC algorithm, which would be the case if the ILC algorithm was used to update the torque feedforward. The ILC algorithms can thus be used as an add-on to the standard industrial controller of the Stäubli RX90 robot.

Computational efficiency

The NILC algorithm and the RILC algorithm are implemented using the algorithms from appendix B, which solve the optimal control problem for NILC formulated in subsection 3.3.2 and the dynamic game for RILC formulated in subsection 4.3.2. The memory and computation time required by those algorithms for the tested trajectories allow implementation on a contemporary PC. The time required to compute the feedforward update for NILC is even less than the time it takes for the robot to move along the trajectory. The RILC algorithm requires more memory and computation time than the NILC, because of the large state-dimension of the dynamic uncertainty pre-weighting filter.

The computational efficiency of the lifted algorithms for NILC and RILC, which are described subsections 3.3.1 and 4.3.1, is also tested. The memory and computation time required by those algorithms for the tested trajectories allow implementation on a conventional PC as well. However, the computation time and memory required for the learning matrix learning matrix grow with the second and third power of the iteration length respectively, which makes the implementation of the lifted algorithms computationally intensive for long iterations.

Monotonic convergence to a small error

A zero-phase low-pass filter is selected as the robustness filter, because the uncertainty in the dynamic models of the robot is largest at high frequencies. The cutoff frequency of the robustness filter for NILC is chosen such that it results in convergence of each frequency component of the feedforward for a set of experimental data. The cutoff frequency of the robustness filter for RILC is chosen such that the sufficient condition for robust convergence is satisfied. The application of the proposed NILC and RILC algorithms with these robustness filters to the Stäubli RX90 robot results in monotonic convergence of the maximum and 2-norm of the tracking error in all experiments described in chapter 6.

Besides the aforementioned effect on the convergence, the choice of the robustness filter has a considerable effect on the final error. It is shown that the maximum error that remains after application of ILC can be estimated from filtering the error before the application of ILC with a high-pass filter with the cutoff frequency of the robustness filter. The frequency components of the error

beyond the cutoff frequency of the robustness filter are not compensated because the robustness filter removes the corresponding frequency components of the feedforward (NILC) or the summed error (RILC). Thus, the higher the cutoff frequency of the robustness filter, the smaller the remaining error. However, the maximum of the remaining error cannot be reduced to less than 0.1 mm. This lower limit is the result of the repeatability of the measurement of the tracking error of the Stäubli RX90 robot by the optical sensor.

The maximum cutoff frequency for which the convergence conditions of NILC and RILC are satisfied for the three models are given in tables 6.1 and 6.2. The cutoff frequency is the smallest for model 1, because this model does not adequately describe the dynamics of the robot beyond bandwidth of the feedback controller. The dynamics of the robot beyond the bandwidth are described by models 2 and 3, resulting in a larger cutoff frequency. The cutoff frequency is the largest for model 3, especially if the robot configuration changes considerably along the trajectory, because this model is able to describe the variation of the robot dynamics along the trajectories. The largest cutoff frequency for NILC is 32.5 Hz and the largest cutoff frequency for RILC is 25.0 Hz. Those cutoff frequencies are beyond the first resonance frequency of the robot mechanism, which is about 20 Hz. The smaller cutoff frequency for RILC is caused by the conservative specification of the model uncertainty and the conservativeness of the sufficient condition for convergence that is used for the selection of the cutoff frequency for RILC.

Fast convergence

The experiments show that the convergence ratio of NILC is reduced by selecting a small weight on the feedforward update. The tracking error is reduced to its final value in only 3 iterations if the weight on the feedforward is three times smaller than the weight on the error.

The experimental results also show that the convergence ratio of RILC is reduced by selecting a smaller maximum convergence ratio. Furthermore, it is shown that the convergence ratio strongly depends on the selection of the uncertainty weighting filters. The tracking error can be reduced to its final value in only 2 iterations by appropriate selection of the maximum convergence ratio and the uncertainty weighting filters.

Fast convergence and a small tracking error

The experimental results show that decreasing the weight on the feedforward update for NILC increases the resulting convergence rate, but hardly affects the allowable cutoff frequency. A high convergence rate and a small final error are thus realised by selecting a small weight on the feedforward update.

The experimental results also show that reducing the maximum convergence ratio for RILC increases the convergence rate, but it reduces the maximum

allowable cutoff frequency considerably and thus increases the final error. Increasing the spectral norm of the uncertainty post-weighting filter increases the convergence rate as well, but this hardly affects the allowable cutoff frequency and thus the final error. The gain of the uncertainty post-weighting filter should be less than the maximum convergence ratio to be able to compute an optimal learning filter. A high convergence rate and a small final error can thus be realised by selecting a maximum convergence ratio close to unity and selecting the spectral norm of the uncertainty post-weighting filter only a little smaller.

Model based ILC algorithms for LTV systems

The experimental results show that the tracking error of the Stäubli RX90 robot, which is measured at the end-effector of the robot, can be reduced to 0.1 mm by the application of the proposed ILC algorithms. This accuracy is sufficient for the application of the robot for laser welding. The small tracking error is only realised along a typical welding trajectory (trajectory B) by reducing the frequency components of the tracking error beyond the bandwidth of the feedback controller. It is even possible to reduce frequency components of the tracking error beyond the first resonance frequency of the robot mechanism, which is required to realise the required level of accuracy along a more complex trajectory (trajectory A). These high-frequency components of the tracking error can only be reduced by ILC using a model that is able to describe the robot dynamics at these frequencies (models 2 and 3). Moreover, if the configuration of the robot changes considerably along the trajectory (trajectory A) then the variation of the dynamics has to be taken into account by the model (model 3) to reduce the high-frequency components of the tracking error by the application of ILC. This demonstrates the value of an ILC algorithm that is able to cope with LTV dynamics.

Summarising, the proposed ILC algorithms are able to realise high-accuracy motion at the tip of an industrial robot by reducing the frequency components of the tracking error beyond the bandwidth of the feedback controller. In section 6.5 it is shown that the realised reduction of the tracking error actually results in an improvement of the weld quality for a typical laser welding task. If the configuration of the robot changes considerably along the trajectory, then the high accuracy is only realised using a model that describes the variation of the robot dynamics along the trajectory. The tracking error can be reduced monotonically to its final value in only a few iterations. Moreover, the ILC algorithms are used as an add-on to the standard industrial controller without adding any feedback action and the algorithms are sufficiently efficient for implementation on a contemporary PC. The proposed ILC algorithms thus satisfy all requirements following from the objective of this work.

7.2 Recommendations for further research

The results of this thesis raise several issues that deserve further investigation. These issues are suggested as subject of further research and may lead to improved performance of the proposed ILC algorithms on several aspects

Improved convergence analysis for norm-optimal ILC

The convergence analysis for NILC is based on checking the convergence condition (3.39). This condition can be checked if the model error is known, though this is mostly not the case in practice. Alternatively, the condition can be checked experimentally for a set of linearly independent feedforward inputs, though this requires a large number of experiments. In this thesis the convergence condition is checked for each frequency component of the feedforward for a set of experimental data, assuming an independent growth of the frequency components of the feedforward (see section 6.2.1).

A more fundamental approach to check convergence of NILC is desirable. Possibly such convergence analysis for NILC could be based on the use of a specification of the model uncertainty, similar to the convergence analysis that is used for RILC. The convergence analysis for RILC is based on the solution of an optimal control problem, which can be solved efficiently. Considering the similarities in the structure of the solutions for NILC and RILC it is expected that a similar convergence analysis can be used for NILC. However, the convergence analysis may be conservative, because it is based on a sufficient condition for convergence.

Improved modelling of the robot dynamics

In this thesis the dynamic model of the Stäubli RX90 robot is based on the identification of the parameters of a time-varying ARX model from experimental data. The resulting model describes the configuration-dependent dynamics of the robot beyond its bandwidth, but it only valid for the trajectory along which the dynamic response is measured. A new model must be estimated for each new trajectory.

It is desirable to have model that is able to describe the high-frequency dynamics of the robot along any trajectory. Probably this can be realised by a robot model that is based on the physics of the robot mechanism. Such model should include the effect of flexibilities in the mechanism on the dynamics to be able to describe the robot dynamics at sufficiently high frequencies. Hardeman (2008) has formulated such model, but the identification of all parameters of the model is still subject of research.

Improved specification of the model uncertainty

In this thesis the specification model uncertainty is derived from the frequency spectrum of the difference between the measured and the simulated dynamic response of the robot. The method yields a conservative specification of the model uncertainty. Firstly, this is the result of using the difference between the measured and the simulated error for the estimation of the model uncertainty, while this difference is not only the result of the model error, but also the effect of stochastic noise. Secondly, the uncertainty in the modelled relations between the two directions of the feedforward and the error is assumed to be independent, while probably some dependency is present in the components of the additive model error. Moreover, the specified model uncertainty is time-invariant, while the robot dynamics vary along the trajectory and the model error is probably time-varying as well.

It is desirable to develop a better method for the estimation of the model uncertainty, which should result in a less conservative specification of the model uncertainty. A reduction of the model uncertainty can probably be used to decrease the convergence ratio or the final error for RILC. Possibly, an improved specification of the model uncertainty can be derived from an estimation of the uncertainty in the estimated parameters of the robot model, which could be either the TVARX model or a model with a physical parametrisation. If the resulting specification of the model uncertainty is time-varying, the proposed RILC algorithm is still usable as it is able to cope with a time-varying model uncertainty.

Suitability for other applications

The requirements on the ILC algorithm following from the objective of this work are formulated using the application to the Stäubli RX90 robot, which is used laser welding, as a reference. The performance of developed the ILC algorithms is also tested on this system. Nevertheless, the ILC algorithms proposed in this thesis are expected to be more generally applicable. It would be interesting to investigate the applicability of the algorithms to more DOFs of the Stäubli RX90 robot, to other robots or to other (mechanical) systems. The algorithms are developed for systems with LTV dynamics and they should thus be applicable to any system with LTV dynamics. The dynamics of a mechanical system with configuration dependent (weakly non-linear) dynamics can be approximated as being LTV for small deviations from a repetitively traced trajectory.

Reducing the effect of iteration-varying disturbances

The final error that remains after the application of the proposed ILC algorithms with a sufficiently high cutoff frequency is predominantly the result of iteration-varying disturbances (compare figure 6.3 and figures 6.4-6.5(b)). The reduction

of the effect of iteration-varying disturbances should thus be investigated to realise if a smaller final error is desired.

One way to reduce the effect of iteration-varying disturbances is to include terms related to the reduction of iteration-varying disturbances in the objective functions as proposed by Dijkstra (2004); Norrlöf and Gunnarsson (2002a); Saab (2004); Yang et al. (2003). The objective functions of the ILC algorithms proposed in this thesis can easily be extended with additional terms related to objectives like the reduction of the effect iteration-varying disturbances. An alternative method to reduce the effect of iteration-varying disturbances is to use measurements of the error from multiple previous iterations or the current iteration for the computation of the feedforward (see the discussion in subsection 2.1.2).

Adaptive type ILC

The ILC algorithms proposed in this work could be used as a basis for the design of adaptive-type ILC. The adaptive add-ons can be used to overcome some of the inherent disadvantages of the proposed ILC algorithms.

An interesting adaptive add-on is the updating of the model using the input-output data measured in the iterations. This method may reduce the model uncertainty, which can be used to decrease the convergence ratio or final error. Starting with a (trajectory independent) model of the low-frequency dynamics, such add-on can be used to estimate the high-frequency dynamics, which makes the prior estimation of a model for each new trajectory obsolete. Preliminary experimental results with this technique are promising. However, more research is needed on the so-called burst-phenomenon (see, e.g., Phan and Frueh, 1999; Tsakalis, 1994), which is described hereafter. The feedforward input to the system, which is computed by the ILC algorithm, might not excite the system dynamics sufficiently, resulting in bad identifiability of some of the model parameters. In the presence of noise this may result in erroneous estimation of the model parameters. The feedforward update computed by ILC with such erroneous model might result in a growth of the tracking error in the next iteration. The advantage of the adaptation of the model is that the growth of the tracking error probably results in a reduction of the model error and a reduction of the tracking error in future iterations. However, the tracking error may grow large in some iterations because of the described mechanism.

Another interesting adaptive add-on is to learn the feedforward along several different trajectories with the proposed ILC algorithms and to use these feedforwards to train a feedforward controller (see, e.g., Arif et al., 2002; Cheah, 2001; Gorinevsky, 1995; Gorinevsky et al., 1997; Lange and Hirzinger, 1995, 1999a,b). Such feedforward controller should then be able to generate a feedforward that compensates for the error along new trajectories. In contrast to non-adaptive ILC algorithms as considered in this work such feedforward controller would not be trajectory specific.

Appendix A

Literature on the application of ILC to robots

A list of publications that consider the application of ILC to robotic manipulators is given in table A.2. The table also lists some properties of the ILC algorithms considered in those publications. The abbreviations used in table A.2 are listed in table A.1.

property	I	type of results presented in publication
	a	only theoretical analysis
	s	simulation
	e	experimental results
property	II	type of ILC considered in publication
	g	gain-type ILC
	m	model-type ILC
	a	adaptive-type ILC
property	III	system class considered in publication
	LTI	Linear Time Invariant
	LTV	Linear Time-Varying
	NL	Non-Linear
	EMR	Equations of Motion of a Rigid robot
	EMF	Equations of Motion of a Flexible robot
	PS	System that satisfies passivity property
	HS	Hamiltonian System
	PVI	system with velocity proportional to input
property	IV	compensation of tracking error at high frequencies
	y	yes
	n	no
property	V	input signal modified by ILC
	t	torque feedforward

	p	position setpoints
	v	velocity setpoints
	a	acceleration setpoints
property	VI	location of measurement of tracking error
	m	motor side of transmission
	a	arm side of transmission
	b	both, motor and arm side of transmission
	c	arm side of transmission computed from motor side
property	VII	measure to prevent amplification at high frequencies
	-	none
	BF	basis functions
	LPF	low-pass robustness filter
	LPA	other type of low-pass filtering
property	VIII	feedback controller applied in parallel with ILC
	-	none
	PD	feedback of Position and its Derivative
	PID	PD plus feedback of Integral of position
	PDD	PD plus feedback of 2^{nd} order Derivative of position
	PDG	PD plus Gravity compensation
	PDS	PD plus Sliding mode control
	FBL	FeedBack Linearisation
	LQG	Linear Quadratic reGulator
	IC	Industrial Controller (no further specification)
property	IX	degrees of freedom in simulation or experiment

Table A.1: Properties listed in table A.2

publication	I	II	III	IV	V	VI	VII	VIII	IX
Amann et al. (1996a)	s	m	LTV	n	t	m	-	-	1
Arif et al. (1999)	s	g	LTV	n	t	m	-	-	1
Arif et al. (2000)	s	g	LTV	n	t	m	-	-	1
Arif et al. (2002)	s	g	NL	n	t	m	-	-	2
Arif et al. (2003)	s	g	LTV	n	t	m	-	-	2
Arimoto (1990)	a	g	LTV	n	t	m	-	PDG	-
Arimoto et al. (1984)	a	g	LTV	n	t	m	-	-	-
Arimoto et al. (2000)	s	g	PS	n	t	c	-	PD	3
Avrachenkov (1998)	s	m	NL	n	t	m	-	-	2
Bondi et al. (1998)	s	g	EMR	n	t	c	-	PDD	2
Bukkems et al. (2005)	e	m	LTI	n	a	m	LPF	FBL	3
Cheng and Wen (1993)	s	m	LTI	y	t	a	BF	PD	2
Choi and Lee (2000)	s	a	EMR	n	t	m	-	PD	2
De Luca et al. (1992)	e	g	LTI	n	t	m	LPF	PD	2
De Luca and Ulivi (1992)	s	m	LTI	n	p	m	LPF	PD	2
Deman et al. (1999)	a	g	PVI	n	v	a	LPF	PID	-
Driessen and Sadegh (2004)	s	g	EMR	n	t	m	-	-	2
Elci et al. (2002)	e	g	LTI	n	p	m	BF	IC	7
Fujimoto and Sugie (2003)	e	g	HS	n	t	m	-	PD	2
Gorinevsky (1995)	s	m	LTV	y	t	b	BF	PD	2
Gorinevsky et al. (1997)	e	m	LTV	y	t	m	BF	PID	2
Guglielmo and Sadegh (1996)	e	a	EMR	y	t	c	BF	PD	4
Gunnarsson et al. (2007)	e	m	LTI	y	p	b	LPA	LQG	1
Gunnarsson and Norrlöff (2001)	e	m	LTI	n	p	m	LPF	IC	1
Hamamoto and Sugie (2002)	e	g	EMR	n	t	c	BF	PD	2
Hatzikos et al. (2004)	s	m	NL	n	t	m	-	-	1
Jiang et al. (1994)	s	a	EMR	n	t	c	-	PDS	2
Kavli (1993)	e	m	LTI	n	p	m	LPA	IC	1
Kawamura et al. (1988)	e	g	LTV	n	t	m	-	PDG	3
Lange and Hirzinger (1995)	e	m	LTI	n	p	m	LPA	IC	6

Lange and Hirzinger (1999a)	e	m	LTI	n	p	a	LPA	IC	6
Lange and Hirzinger (1999b)	e	m	LTI	n	p	a	LPA	IC	2
Longman (2000)	e	g	LTI	n	p	m	LPF	IC	7
Mita and Kato (1985)	s	g	LTI	n	t	m	LPF	FBL	3
Miyazaki et al. (1986)	e	g	EMF	y	t	b	LPF	-	2
Norrlöf (2000)	e	m	LTI	n	p	m	LPF	IC	3
Norrlöf and Gunnarsson (2002a)	e	a	LTI	n	p	m	LPF	IC	3
Norrlöf and Gunnarsson (2002b)	e	m	LTI	n	p	m	LPF	IC	3
Oh et al. (1988)	s	a	LTV	n	t	m	-	-	2
Polushin and Tayebi (2004)	e	a	EMR	n	t	m	LPF	PD	6
Poo et al. (1996)	e	m	EMR	n	t	m	-	-	2
Tang et al. (2000)	e	g	EMR	n	t	m	BF	PD	3
Tayebi (2004)	s	a	EMR	n	t	m	-	PD	2
Tayebi and Islam (2006)	e	a	EMR	n	t	m	LPA	PD	3
Togai and Yamano (1985)	e	g	LTI	n	t	m	-	IC	2
Tso and Ma (1992)	s	m	EMR	n	t	m	-	-	2
Velthuis et al. (1996)	s	g	LTI	n	t	m	BF	PD	1
Wada et al. (1993)	e	g	FMR	y	t	a	LPF	PD	1
Wang (1995)	a	g	EMF	y	t	a	-	PD	-
Xu and Xu (2004)	s	a	NL	n	t	m	-	P	1
Ye and Wang (2005)	s	g	LTI	n	p	m	LPF	IC	1
This work	e	m	LTV	y	p	a	LPF	IC	2

Table A.2: Literature on the application of ILC to robotic manipulators

Appendix B

Solution to optimal control problems

In this appendix the solutions to two optimal control problems are discussed. The affine quadratic discrete-time optimal control problem is considered in section B.1. The solution to this problem used to compute the optimal feedforward update for norm-optimal ILC (chapter 3). The affine quadratic two-person zero-sum dynamic game is considered in section B.2. The solution to this problem is used to compute the optimal learning filter for robust ILC (chapter 4).

B.1 Affine quadratic discrete-time optimal control problem

Considerer the control problem consisting of the following objective, objective function and state equation

$$\check{u}_i = \arg\min_{u_i} J, \tag{B.1a}$$

$$J = \sum_{i=1}^{N_i-1} \left({x_{i+1}}^T Q_{i+1} x_{i+1} + {u_i}^T R_i^{(u)} u_i + {v_i}^T R_i^{(v)} v_i \right), \tag{B.1b}$$

$$x_{i+1} = A_i x_i + B_i^{(u)} u_i + B_i^{(v)} v_i \tag{B.1c}$$

where it is assumed that $R_i^{(u)} > 0$ and the initial state x_1 and the deterministic input v_i are known. Lewis and Syrmos (1995) solve this problem for an LTI system and time-varying gains in the objective function. Başar and Olsder (1995) solve the problem for an LTV system and time-varying gains. Both solutions yield the same optimal input for an LTI system, but the derivation presented by Lewis and Syrmos (1995) is more straightforward. Therefore the line of that derivation is used hereafter to derive the solution of the optimal

control problem for an LTV system and time-varying gains in the objective function.

The state equation is considered as a constraint equation that is accounted for in the optimisation problem using the Lagrange-multiplier technique. Introducing the Lagrange multiplier λ_i, the objective function J becomes

$$J = \sum_{i=1}^{N_i-1} \Big({x_{i+1}}^T Q_{i+1} x_{i+1} + {u_i}^T R_i^{(u)} u_i + {v_i}^T R_i^{(v)} v_i \\ +2\lambda_{i+1}^T \Big(A_i x_i + B_i^{(u)} u_i + B_i^{(v)} v_i - x_{i+1} \Big) \Big). \quad \text{(B.2)}$$

A stationary point of J is found by equating the derivative of J with respect to u_i, x_i and λ_i to zero. Equating the derivative of J with respect to λ_i to zero yields the state equation. Equating the derivative of J with respect to u_i to zero yields

$$\begin{aligned} &\frac{\partial J}{\partial u_i} = O, \\ &\Rightarrow 2R_i^{(u)} u_i + 2B_i^{(u)T} \lambda_{i+1} = O, \\ &\Rightarrow u_i = -R_i^{(u)-1} B_i^{(u)T} \lambda_{i+1}. \end{aligned} \quad \text{(B.3)}$$

Equating the derivative of J with respect to x_i to zero yields

$$\begin{aligned} &\frac{\partial J}{\partial x_i} = O, \\ &\Rightarrow \begin{cases} 2Q_i x_i - 2\lambda_i + 2A_i^T \lambda_{i+1} = O & \text{for } i = 1 \dots N_i - 1, \\ 2Q_{N_i} x_{N_i} - 2\lambda_{N_i} = O, \end{cases} \\ &\Rightarrow \begin{cases} \lambda_i = A_i^T \lambda_{i+1} + Q_i x_i & \text{for } i = 1 \dots N_i - 1, \\ \lambda_{N_i} = Q_{N_i} x_{N_i}. \end{cases} \end{aligned} \quad \text{(B.4)}$$

The (unknown) input is eliminated from the state-equation by inserting equation (B.3) in equation (B.1c), yielding

$$x_{i+1} = A_i x_i + P_i \lambda_{i+1} + B_i^{(v)} v_i, \quad \text{(B.5)}$$

where

$$P_i = -B_i^{(u)} R_i^{(u)-1} B_i^{(u)T}. \quad \text{(B.6)}$$

Equations (B.4) and (B.5) define two coupled state-equations running in the opposite time-direction. The causal and the anti-causal state-equations are decoupled by the introduction of the following state-transformation

$$\lambda_i = S_i x_i + \eta_i. \quad \text{(B.7)}$$

Substituting this state-transformation in equation (B.5) gives

$$x_{i+1} = A_i x_i + P_i S_{i+1} x_{i+1} + P_i \eta_{i+1} + B_i^{(v)} v_i. \quad \text{(B.8)}$$

An expression for x_{i+1} can be derived from this equation using the following definition

$$L_i = I - P_i S_{i+1}. \tag{B.9}$$

If L_i is invertible, then x_{i+1} can be expressed as

$$x_{i+1} = L_i^{-1}\left(A_i x_i + P_i \eta_{i+1} + B_i^{(v)} v_i\right). \tag{B.10}$$

Substitution of the state transformation in equation (B.4) gives

$$\left\{\begin{array}{rl} S_i x_i + \eta_i & = A_i^T S_{i+1} x_{i+1} \\ & \quad + A_i^T \eta_{i+1} + Q_i x_i \quad \text{for } i = 1 \ldots N_i - 1, \\ S_{N_i} x_{N_i} + \eta_{N_i} & = Q_{N_i} x_{N_i}. \end{array}\right. \tag{B.11}$$

Subsequent substitution of equation (B.10) gives

$$\left\{\begin{array}{rll} S_i x_i + \eta_i & = A_i^T S_{i+1} L_i^{-1}\left(A_i x_i + P_i \eta_{i+1} + B_i^{(v)} v_i\right) & \text{for } i = \\ & \quad + A_i^T \eta_{i+1} + Q_i x_i & 1 \ldots N_i - 1, \\ S_{N_i} x_{N_i} + \eta_{N_i} & = Q_{N_i} x_{N_i}. & \end{array}\right. \tag{B.12}$$

This equation holds for any x_i if

$$\left\{\begin{array}{rl} S_i & = A_i^T S_{i+1} L_i^{-1} A_i + Q_i \quad \text{for } i = 1 \ldots N_i - 1, \\ S_{N_i} & = Q_{N_i}, \end{array}\right. \tag{B.13}$$

and

$$\left\{\begin{array}{rl} \eta_i & = A_i^T S_{i+1} L_i^{-1}\left(P_i \eta_{i+1} + B_i^{(v)} v_i\right) + A_i^T \eta_{i+1} \quad \text{for } i = 1 \ldots N_i - 1, \\ \eta_{N_i} & = O. \end{array}\right. \tag{B.14}$$

Note that equation (B.13) is the non-stationary Riccati difference equation and (B.14) is an anti-causal state convolution

Finally, the state transformation is substituted in the equation (B.3) to express the input u_i in the states x_i and η_i

$$u_i = -R_i^{(u)-1} B_i^{(u)T}\left(S_{i+1} x_{i+1} + \eta_{i+1}\right). \tag{B.15}$$

For this optimal input, the value of J can be expressed as (Başar and Olsder, 1995, page 240)

$$\begin{aligned} J = \left(2\eta_1 + S_1 x_1\right)^T x_1 + \sum_{i=1}^{N_i - 1} \Bigg(& \left(2\eta_{i+1} + S_{i+1} B_i^{(v)} v_i\right)^T L_i^{-1} B_i^{(v)} v_i \\ & - \eta_{i+1}^T Q_{i+1}^{(\eta)} \eta_{i+1} + {v_i}^T R_i^{(v)} v_i \Bigg), \end{aligned} \tag{B.16}$$

where

$$Q_{i+1}^{(\eta)} = B_i^{(u)} \left(R_i^{(u)} + B_i^{(u)T} S_{i+1} B_i^{(u)} \right)^{-1} B_i^{(u)T}. \tag{B.17}$$

Assuming $R_i^{(u)} > 0$, this value of J is a unique minimum with respect to u_i if the following necessary and sufficient condition is satisfied (Başar and Olsder, 1995)

$$R_i^{(u)} + B_i^{(u)T} S_{i+1} B_i^{(u)} > 0. \tag{B.18}$$

This condition also implies that L_i is invertible. Moreover, it can be shown that if $R_i^{(u)} > 0$ and $Q_i > 0$, then $S_i > 0$ and thus condition (B.18) is satisfied.

Similarly, assuming $R_i^{(u)} < 0$, J would have a unique maximum with respect to u_i if

$$R_i^{(u)} + B_i^{(u)T} S_{i+1} B_i^{(u)} < 0. \tag{B.19}$$

This condition also implies that L_i is invertible. Moreover, it can be shown that if $R_i^{(u)} < 0$ and $Q_i < 0$, then $S_i < 0$ and thus condition (B.18) is satisfied.

Summarising, the procedure to compute the minimising input u_i consists of the following steps:

- compute the time-varying Riccati matrix S_i from equation (B.13), using the definitions of L_i and P_i in equations (B.9) and (B.6), where it is assumed that L_i is invertible,
- check condition (B.18) for the existence of a minimising input u_i
- compute the costate η_i from equation (B.14),
- compute the state x_i from equation (B.10),
- compute the optimal input u_i from equation (B.15).

B.2 Affine quadratic two-person zero-sum dynamic game

Considerer the control problem consisting of the following objective, objective function and state equation

$$\breve{u}_i = \arg\min_{u_i}\max_{q_i} J, \tag{B.20a}$$

$$J = \sum_{i=1}^{N_i-1}\left({x_{i+1}}^T Q_{i+1} x_{i+1} + {u_i}^T R_i^{(u)} u_i + {q_i}^T R_i^{(q)} q_i + {v_i}^T R_i^{(v)} v_i\right), \tag{B.20b}$$

$$x_{i+1} = A_i x_i + B_i^{(u)} u_i + B_i^{(q)} q_i + B_i^{(v)} v_i \tag{B.20c}$$

where it is assumed that $R_i^{(u)} > 0$, $R_i^{(u)} < 0$ and the initial state x_1 and the deterministic input v_i are known. The Nash solution to this problem is derived hereafter. The resulting inputs $\boldsymbol{q}^k$ and $\boldsymbol{u}^k$ independently optimise the objective function for the worst case effect of the other variable such that a deviation of either of the inputs from their optimum yields a smaller or a larger objective function respectively. Basar and Olsder (1995) give the solution for an LTV system with $R_i^{(u)} = I, R_i^{(q)} = -I$. A solution for general weighting matrices is given by Hung and Yang (2002). Their solution is presented in this section in a notation that complies with the previous section.

The state equation is considered as a constraint equation that is accounted for in the optimisation problem using the Lagrange-multiplier technique. Introducing the Lagrange multiplier λ_i, the objective function J becomes

$$\begin{aligned} J = \sum_{i=1}^{N_i-1}\Big({x_{i+1}}^T Q_{i+1} x_{i+1} + {u_i}^T R_i^{(u)} u_i + {q_i}^T R_i^{(q)} q_i \\ +2\lambda_{i+1}^T\left(A_i x_i + B_i^{(u)} u_i + B_i^{(q)} q_i + B_i^{(v)} v_i - x_{i+1}\right)\Big) \end{aligned} \tag{B.21}$$

A stationary point of J is found by equating the derivative of J with respect to u_i, q_i, x_i and λ_i to zero. Equating the derivative of J with respect to λ_i to zero yields the state equation. Equating the derivative of J with respect to u_i and q_i to zero yields

$$\begin{aligned} &\frac{\partial J}{\partial u_i} = O, \\ &\quad \Rightarrow 2R_i^{(u)} u_i + 2B_i^{(u)T}\lambda_{i+1} = O, \\ &\quad \Rightarrow u_i = -R_i^{(u)-1} B_i^{(u)T}\lambda_{i+1}, \end{aligned} \tag{B.22}$$

$$\begin{aligned} &\frac{\partial J}{\partial q_i} = O, \\ &\quad \Rightarrow 2R_i^{(q)} q_i + 2B_i^{(q)T}\lambda_{i+1} = O, \\ &\quad \Rightarrow q_i = -R_i^{(q)-1} B_i^{(q)T}\lambda_{i+1}. \end{aligned} \tag{B.23}$$

Differentiation of J with respect to x_i yields

$$\begin{aligned}
&\frac{\partial J}{\partial x_i} = O, \\
&\Rightarrow \begin{cases} 2Q_i x_i - 2\lambda_i + 2A_i^T \lambda_{i+1} = O & \text{for } i = 1 \dots N_i - 1, \\ 2Q_{N_i} x_{N_i} - 2\lambda_{N_i} = O, \end{cases} \\
&\Rightarrow \begin{cases} \lambda_i = A_i^T \lambda_{i+1} + Q_i x_i & \text{for } i = 1 \dots N_i - 1, \\ \lambda_{N_i} = Q_{N_i} x_{N_i}. \end{cases}
\end{aligned} \tag{B.24}$$

The (unknown) inputs u_i and q_i are eliminated from the state-equation by inserting equations (B.22) and (B.23) in equation (B.20c), yielding

$$x_{i+1} = A_i x_i + P_i \lambda_{i+1} + B_i^{(v)} v_i, \tag{B.25}$$

where

$$P_i = -B_i^{(u)} R_i^{(u)-1} B_i^{(u)T} - B_i^{(q)} R_i^{(q)-1} B_i^{(q)T}. \tag{B.26}$$

Equations (B.24) and (B.25) define two coupled state-equations running in the opposite time-direction. These equations are decoupled by the introduction of the following state-transformation

$$\lambda_i = S_i x_i + \eta_i. \tag{B.27}$$

Substituting this state transformation into equation (B.25) gives

$$x_{i+1} = A_i x_i + P_i S_{i+1} x_{i+1} + P_i \eta_{i+1} + B_i^{(v)} v_i. \tag{B.28}$$

An expression for x_{i+1} can be derived from this equation using the following definition

$$L_i = I - P_i S_{i+1}. \tag{B.29}$$

If L_i is invertible, then x_{i+1} can be expressed as

$$x_{i+1} = L_i^{-1} \left(A_i x_i + P_i \eta_{i+1} + B_i^{(v)} v_i \right). \tag{B.30}$$

Substitution of the state transformation in equation (B.24) gives

$$\begin{cases} S_i x_i + \eta_i &= A_i^T S_{i+1} x_{i+1} \\ &\quad + A_i^T \eta_{i+1} + Q_i x_i \quad \text{for } i = 1 \dots N_i - 1, \\ S_{N_i} x_{N_i} + \eta_{N_i} &= Q_{N_i} x_{N_i}. \end{cases} \tag{B.31}$$

Subsequent substitution of equation (B.30) gives

$$\begin{cases} S_i x_i + \eta_i &= A_i^T S_{i+1} L_i^{-1} \left(A_i x_i + P_i \eta_{i+1} + B_i^{(v)} v_i \right) \quad \text{for } i = \\ &\quad + A_i^T \eta_{i+1} + Q_i x_i \qquad\qquad\qquad\qquad\qquad\qquad 1 \dots N_i - 1, \\ S_{N_i} x_{N_i} + \eta_{N_i} &= Q_{N_i} x_{N_i}. \end{cases} \tag{B.32}$$

This equation holds for any x_i if

$$\begin{cases} S_i &= A_i^T S_{i+1} L_i^{-1} A_i + Q_i \quad \text{for } i = 1 \dots N_i - 1, \\ S_{N_i} &= Q_{N_i}, \end{cases} \tag{B.33}$$

and

$$\begin{cases} \eta_i &= A_i^T S_{i+1} L_i^{-1} \left(P_i \eta_{i+1} + B_i^{(v)} v_i \right) + A_i^T \eta_{i+1} \quad \text{for } i = 1 \dots N_i - 1, \\ \eta_{N_i} &= O. \end{cases} \tag{B.34}$$

Note that equation (B.33) is the non-stationary Riccati difference equation and (B.34) is an anti-causal state convolution.

Finally, the state transformation is substituted in the equations (B.22) and (B.23) to express the inputs u_i and q_i in the states x_i and η_i

$$u_i = -R_i^{(u)-1} B_i^{(u)T} \left(S_{i+1} x_{i+1} + \eta_{i+1} \right), \tag{B.35}$$

$$q_i = -R_i^{(q)-1} B_i^{(q)T} \left(S_{i+1} x_{i+1} + \eta_{i+1} \right). \tag{B.36}$$

For these optimal inputs, the value of J can be expressed as (Hung and Yang, 2002)

$$\begin{aligned} J = (2\eta_1 + S_1 x_1)^T x_1 + \sum_{i=1}^{N_i - 1} \Bigg(& \left(2\eta_{i+1} + S_{i+1} B_i^{(v)} v_i \right)^T L_i^{-1} B_i^{(v)} v_i \\ & - \eta_{i+1}^T Q_{i+1}^{(\eta)} \eta_{i+1} + {v_i}^T R_i^{(v)} v_i \Bigg), \end{aligned} \tag{B.37}$$

where

$$\begin{aligned} Q_{i+1}^{(\eta)} = & - \left(I + B_i^{(u)} R_i^{(u)-1} B_i^{(u)T} S_{i+1} \right)^{-1} B_i^{(q)} \\ & \left(R_i^{(q)} + B_i^{(q)T} S_{i+1} \left(I + B_i^{(u)} R_i^{(u)-1} B_i^{(u)T} S_{i+1} \right)^{-1} B_i^{(q)} \right)^{-1} \\ & B_i^{(q)T} \left(I + B_i^{(u)} R_i^{(u)-1} B_i^{(u)T} S_{i+1} \right)^{-T} \\ & - B_i^{(u)} \left(R_i^{(u)} + B_i^{(u)T} S_{i+1} B_i^{(u)} \right)^{-1} B_i^{(u)T}. \end{aligned} \tag{B.38}$$

The stationary point of J is a minimum with respect to u_i if the objective function has a minimum with respect to u_i for any q_i. This can be checked by considering q_i as an external (fixed) input and using the theory from section B.1 to determine if the remaining optimal control problem has a minimum with respect to u_i. This involves the computation of the matrix sequences P_i, L_i and S_i according to equations (B.6), (B.9) and (B.13) respectively. Then the objective function has a minimum with respect to u_i if condition (B.18) is

satisfied. Similarly, the stationary point of J is a maximum with respect to q_i if the objective function has a maximum with respect to q_i for any u_i. Again, this can be checked by considering u_i as an external (fixed) input and using the theory from section B.1 to determine if the remaining optimal control problem has a maximum with respect to q_i. This involves the computation of the matrix sequences P_i, L_i and S_i according to equations (B.6), (B.9) and (B.13) respectively, where matrices $R_i^{(u)}$ and $B_i^{(u)}$ are replaced by $R_i^{(q)}$ and $B_i^{(q)}$ respectively. Then the objective function has a maximum with respect to q_i if condition (B.19) is satisfied.

A more efficient way to check the conditions for the stationary point J to be a minimum with respect to u_i and a maximum with respect to q_i could possibly be derived directly from conditions on S_i in equation (B.33), though this possibility is not investigated further in this work.

The procedure above checks the existence of the Nash solution. The conditions for the existence of the Nash solution are sufficient for the existence of the Stackelberg solution (see subsection 4.3.1).

Summarising, a procedure to compute the minimising input u_i and the maximising input q_i consists of the following steps:

- check conditions for the existence of a maximising input q_i and a minimising input u_i according to the previously described procedure using the theory from section B.1,
- compute the time-varying Riccati matrix S_i from equation (B.33), using the definitions of L_i and P_i in equations (B.29) and (B.26), where it is assumed that L_i is invertible
- compute the costate η_i from equation (B.34),
- compute the state x_i from equation (B.30),
- compute the maximising input q_i from equation (B.36),
- compute the minimising input u_i from equation (B.35),

Appendix C

Experimental results

This appendix contains figures that show the results from the experiments described in chapter 6. Figures C.1-C.8 show the MAX and RMS tracking error in all iterations of these experiments. Figures C.9-C.24 show the tracking errors in iterations 1 and 9 of the experiments. The figures correspond to the experiments listed in table C.1. The subfigures that are crossed out correspond to the RILC experiments for which the SCRC is not satisfied (see subsection 6.2.2).

trajectory	method	γ	figure with MAX and RMS final error	figure with error in iteration 1	figure with error in iteration 9
A	NILC	-	C.1	C.9	C.10
A	RILC	0.99	C.2	C.11	C.12
A	RILC	0.75	C.3	C.13	C.14
A	RILC	0.50	C.4	C.15	C.16
B	NILC	-	C.5	C.17	C.18
B	RILC	0.99	C.6	C.19	C.20
B	RILC	0.75	C.7	C.21	C.22
B	RILC	0.50	C.8	C.23	C.24

Table C.1: The experiments displayed in figures C.1-C.24

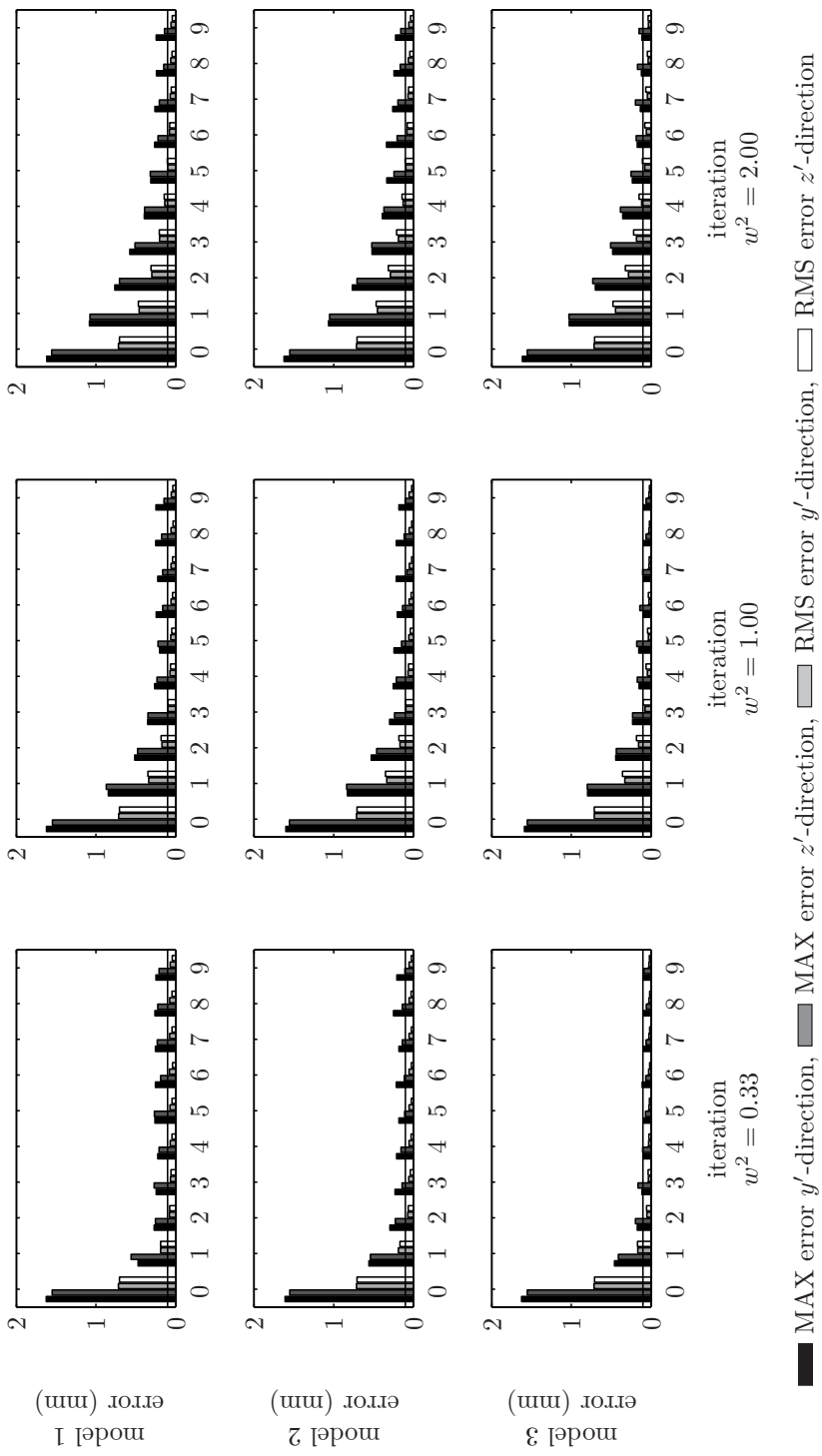


Figure C.1: MAX and RMS error along trajectory A for NILC

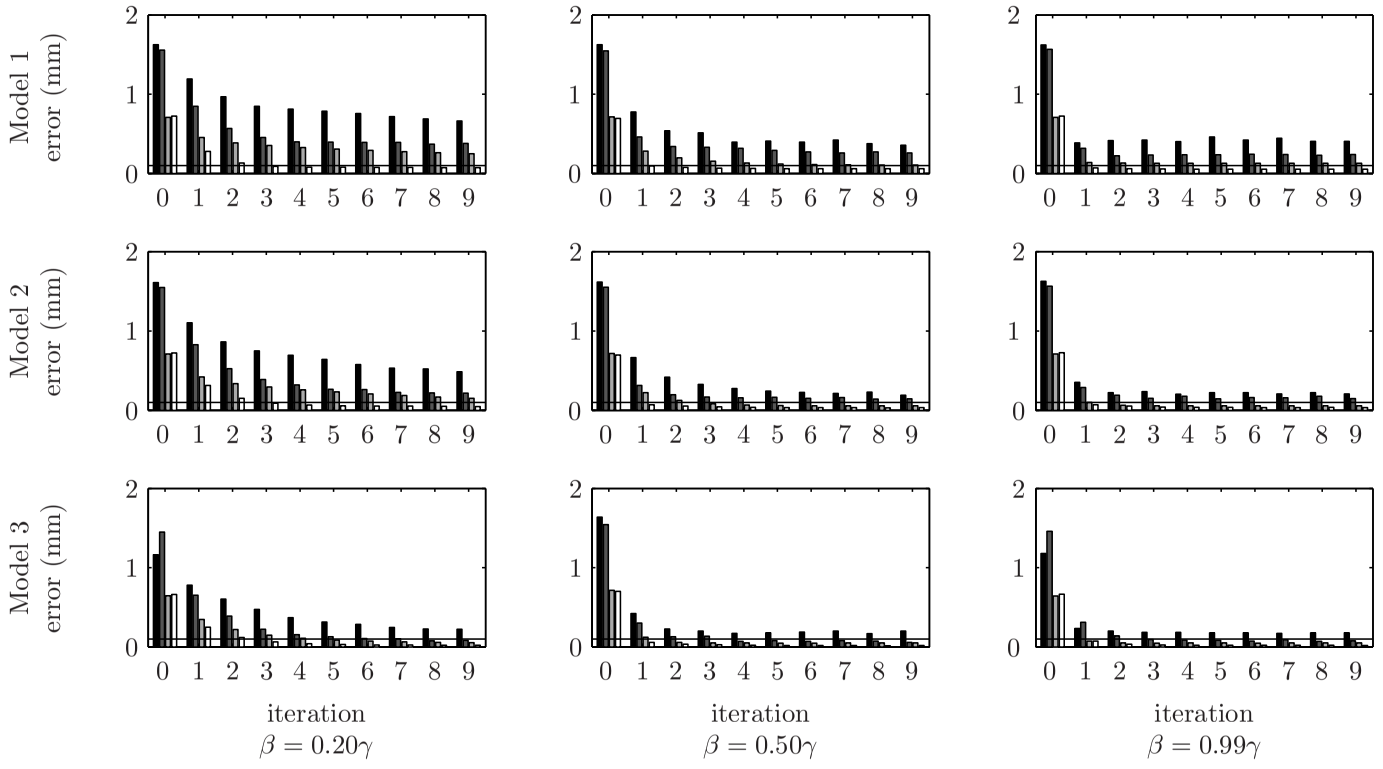


Figure C.2: MAX and RMS error along trajectory A for RILC with $\gamma = 0.99$

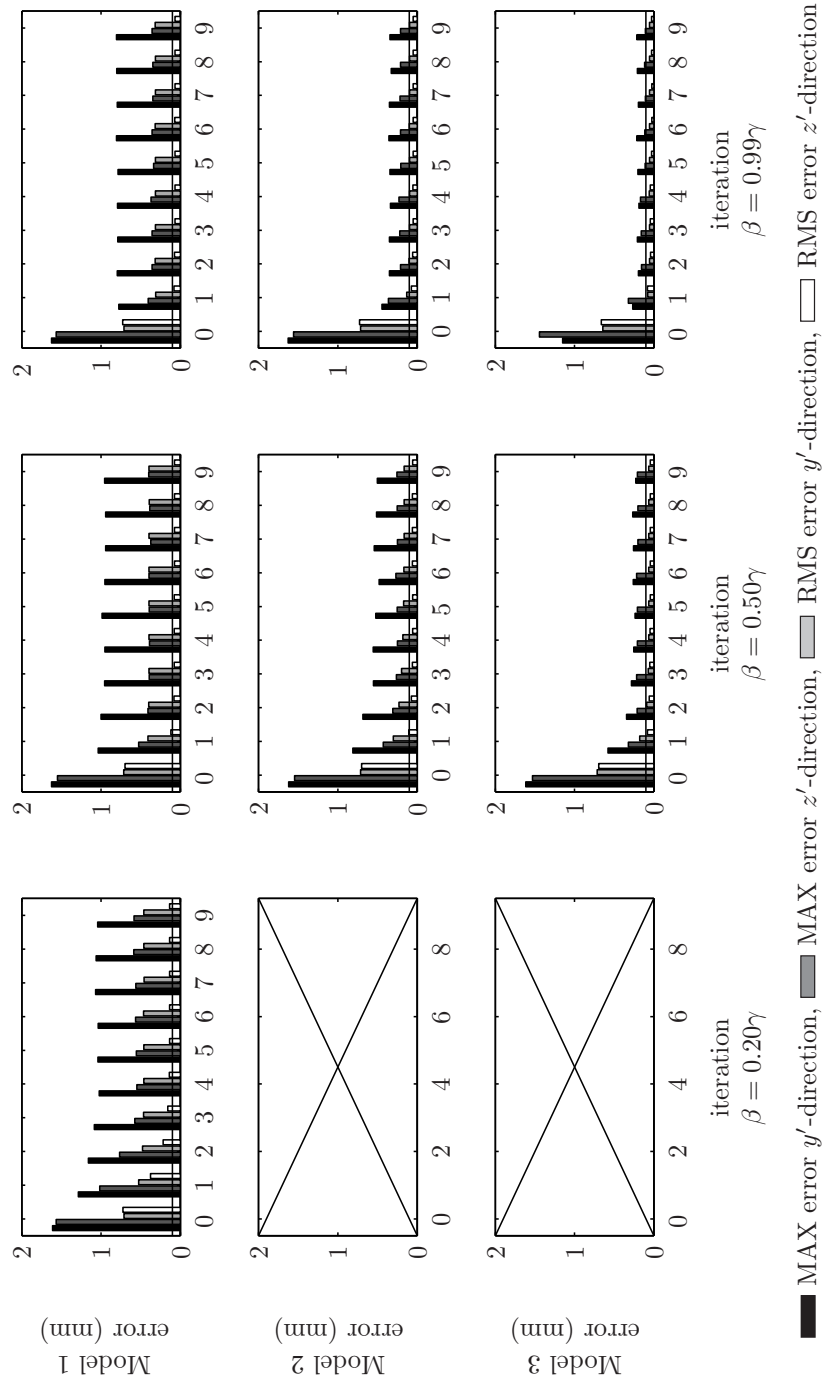


Figure C.3: MAX and RMS error along trajectory A for RILC with $\gamma = 0.75$

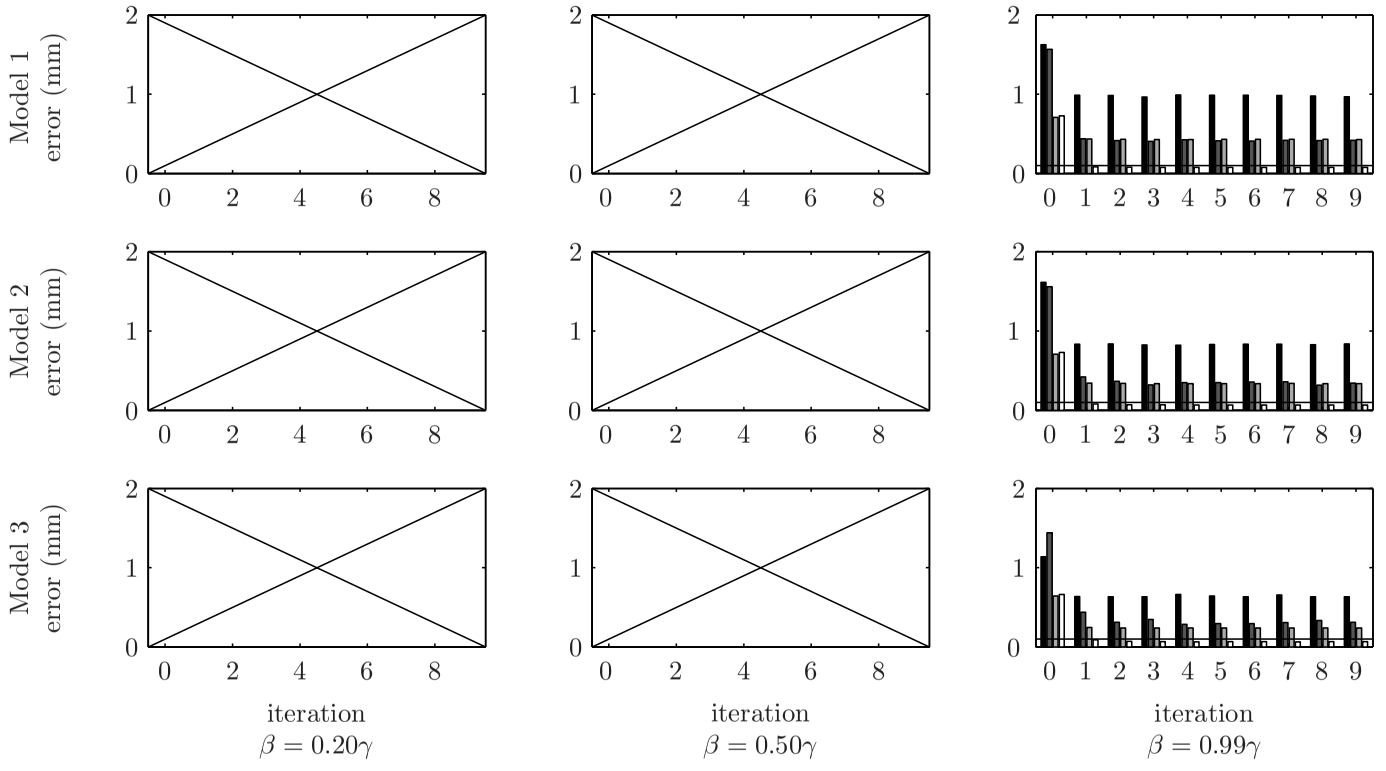


Figure C.4: MAX and RMS error along trajectory A for RILC with $\gamma = 0.50$

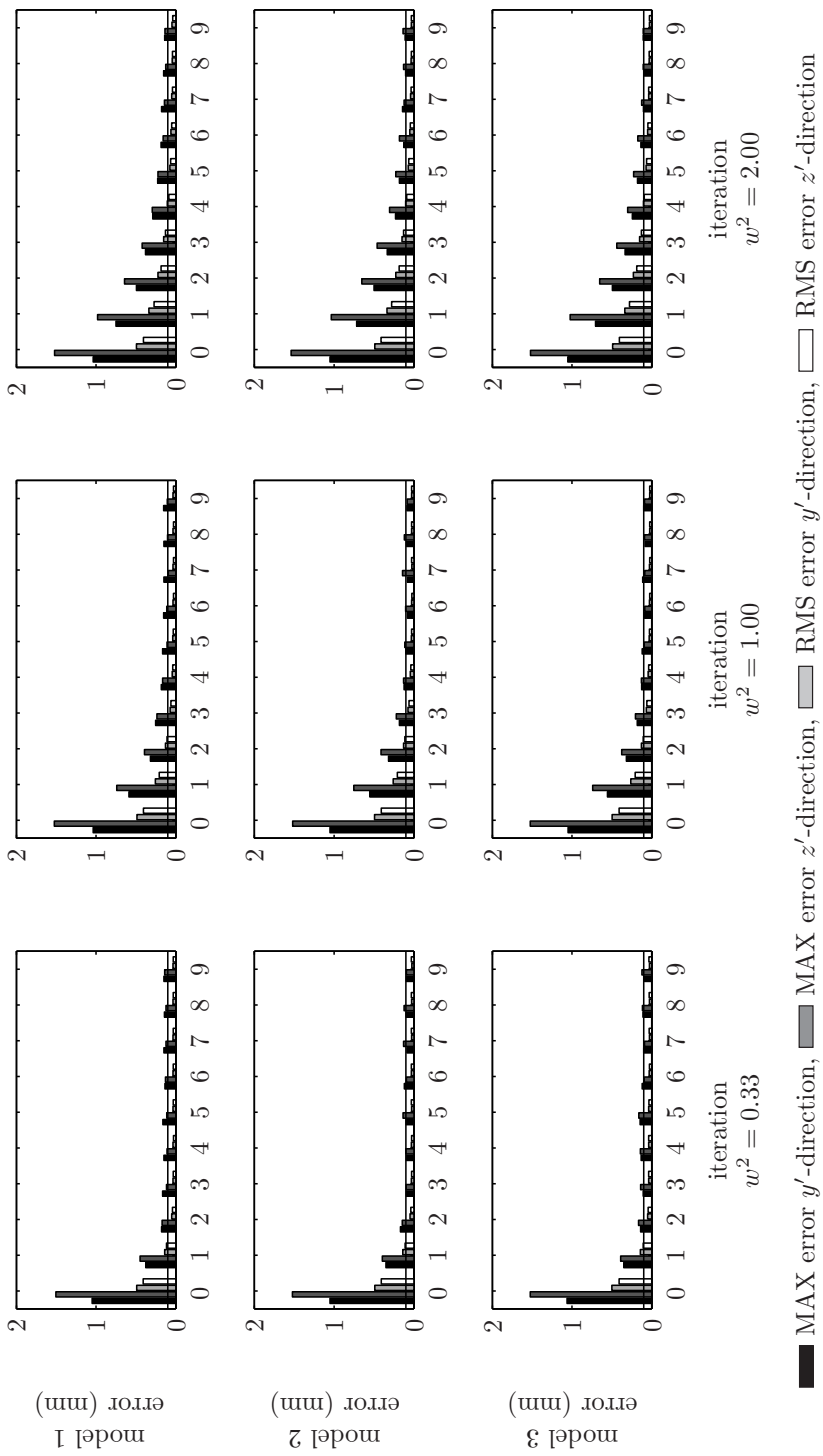


Figure C.5: MAX and RMS error along trajectory B for NILC

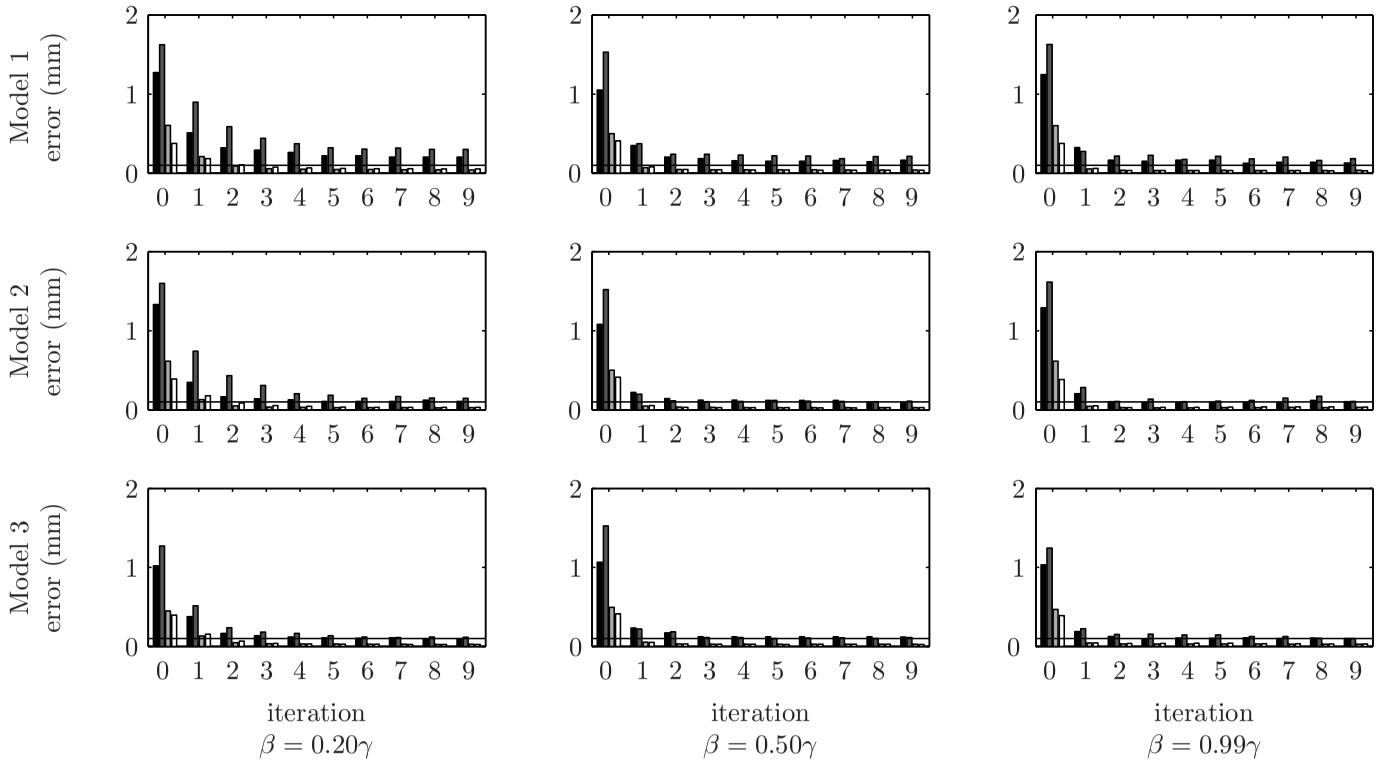


Figure C.6: MAX and RMS error along trajectory B for RILC with $\gamma = 0.99$

Figure C.7: MAX and RMS error along trajectory B for RILC with $\gamma = 0.75$

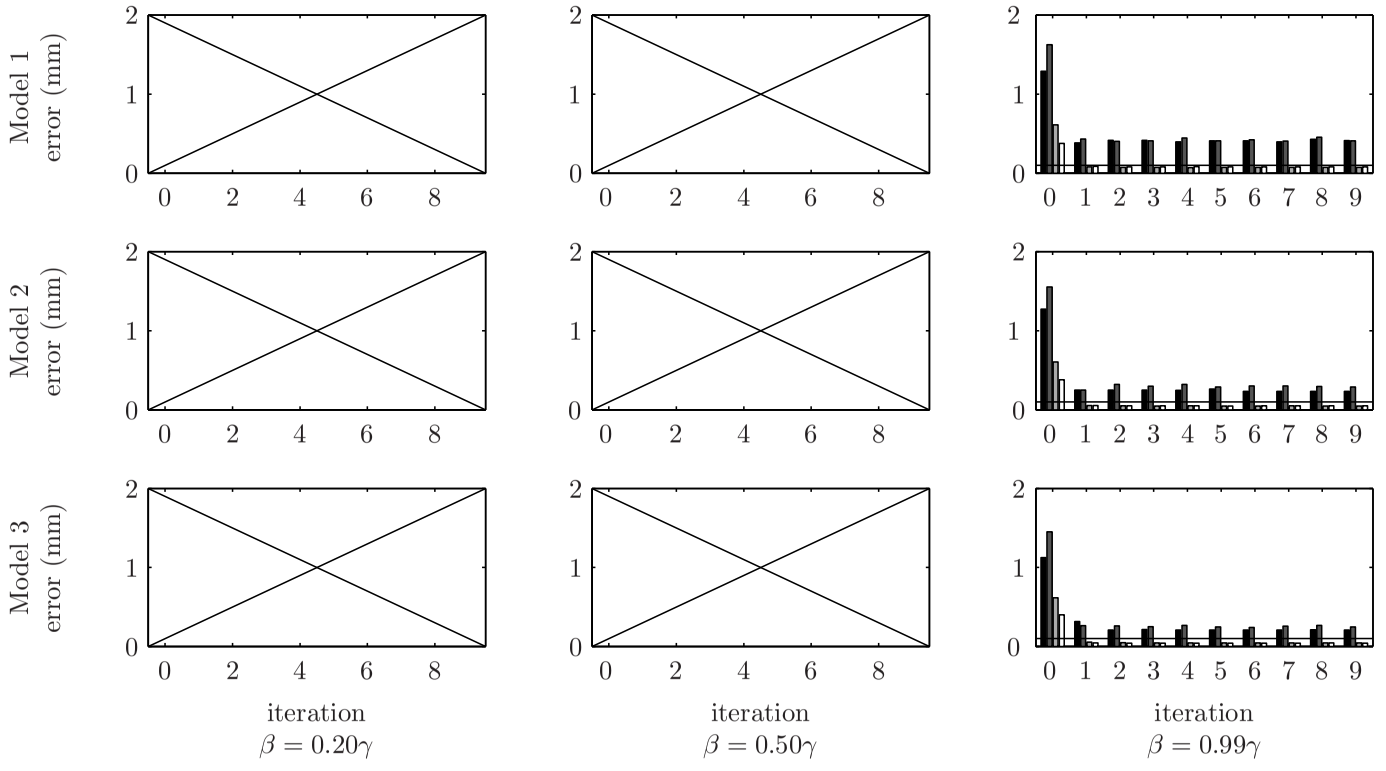


Figure C.8: MAX and RMS error along trajectory B for RILC with $\gamma = 0.50$

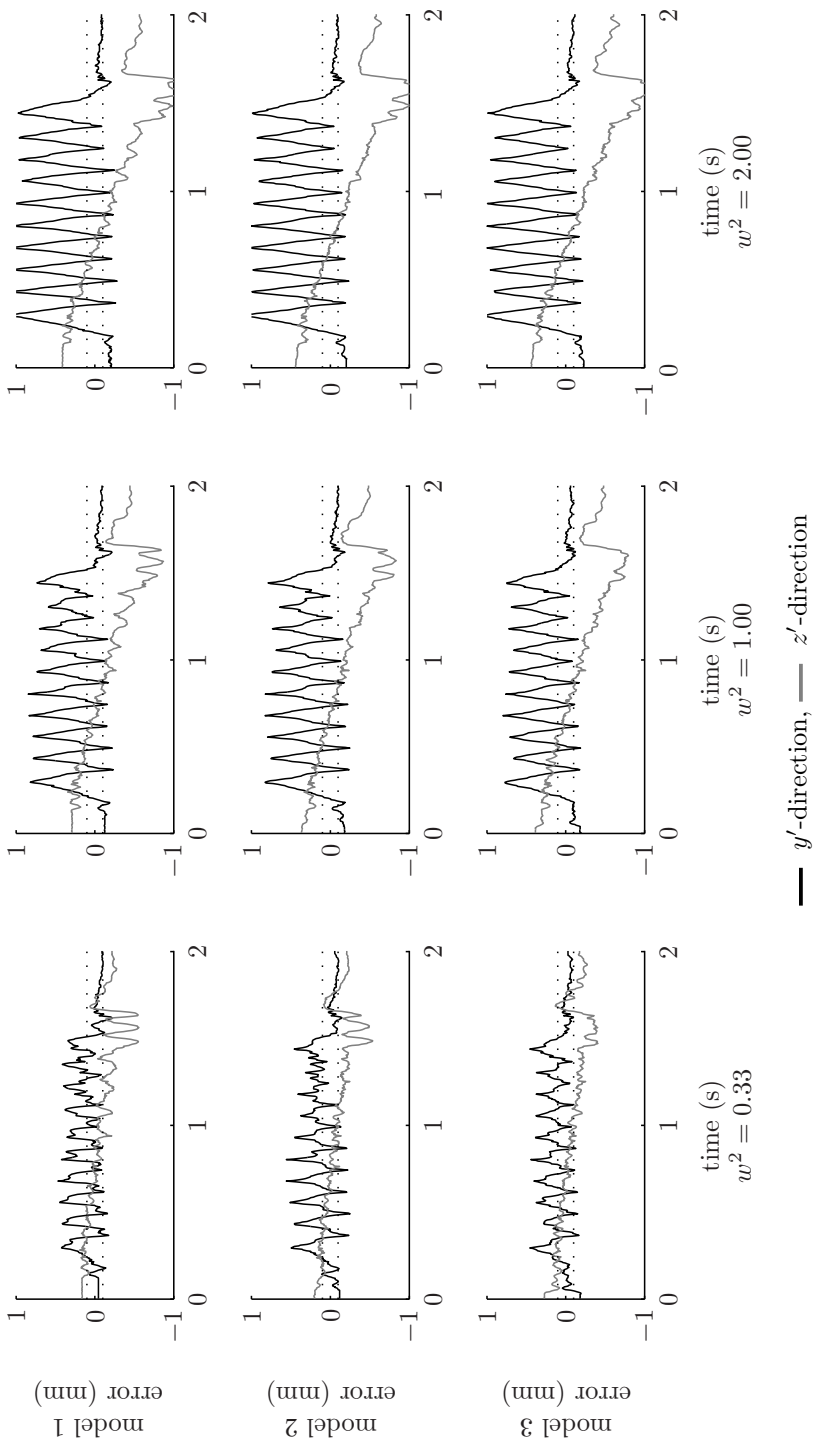


Figure C.9: Tracking error along trajectory A in iteration 1 of NILC

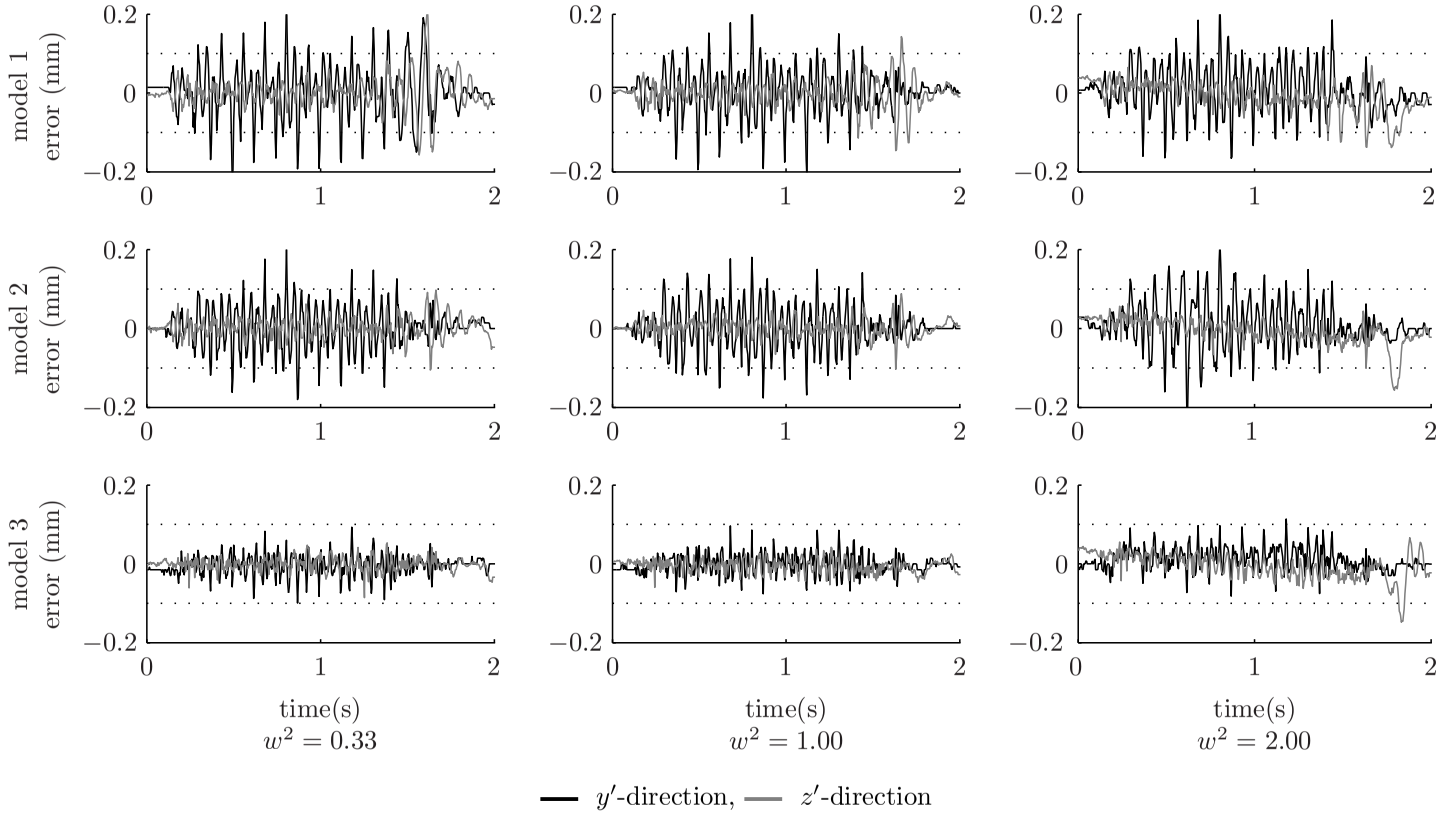


Figure C.10: Tracking error along trajectory A in iteration 9 of NILC

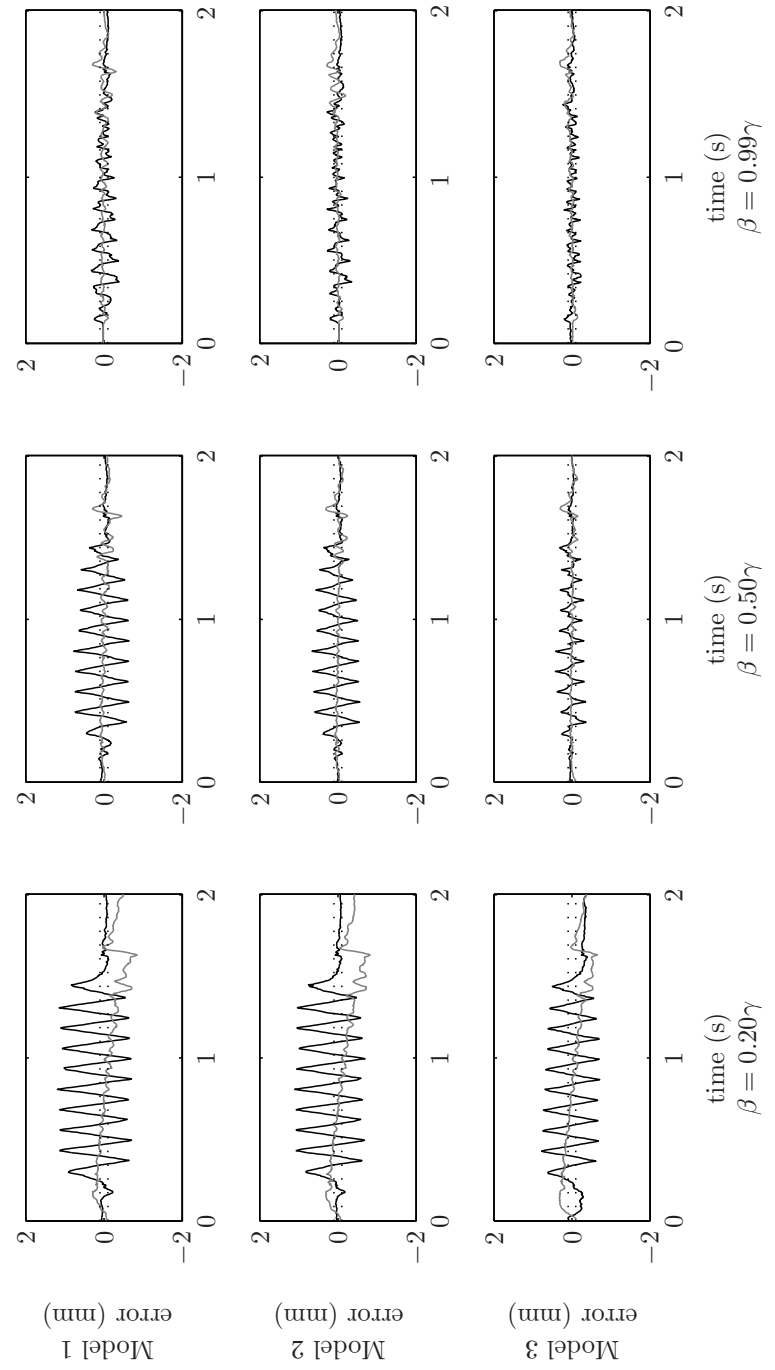


Figure C.11: Tracking error along trajectory A in iteration 1 of RILC with $\gamma = 0.99$

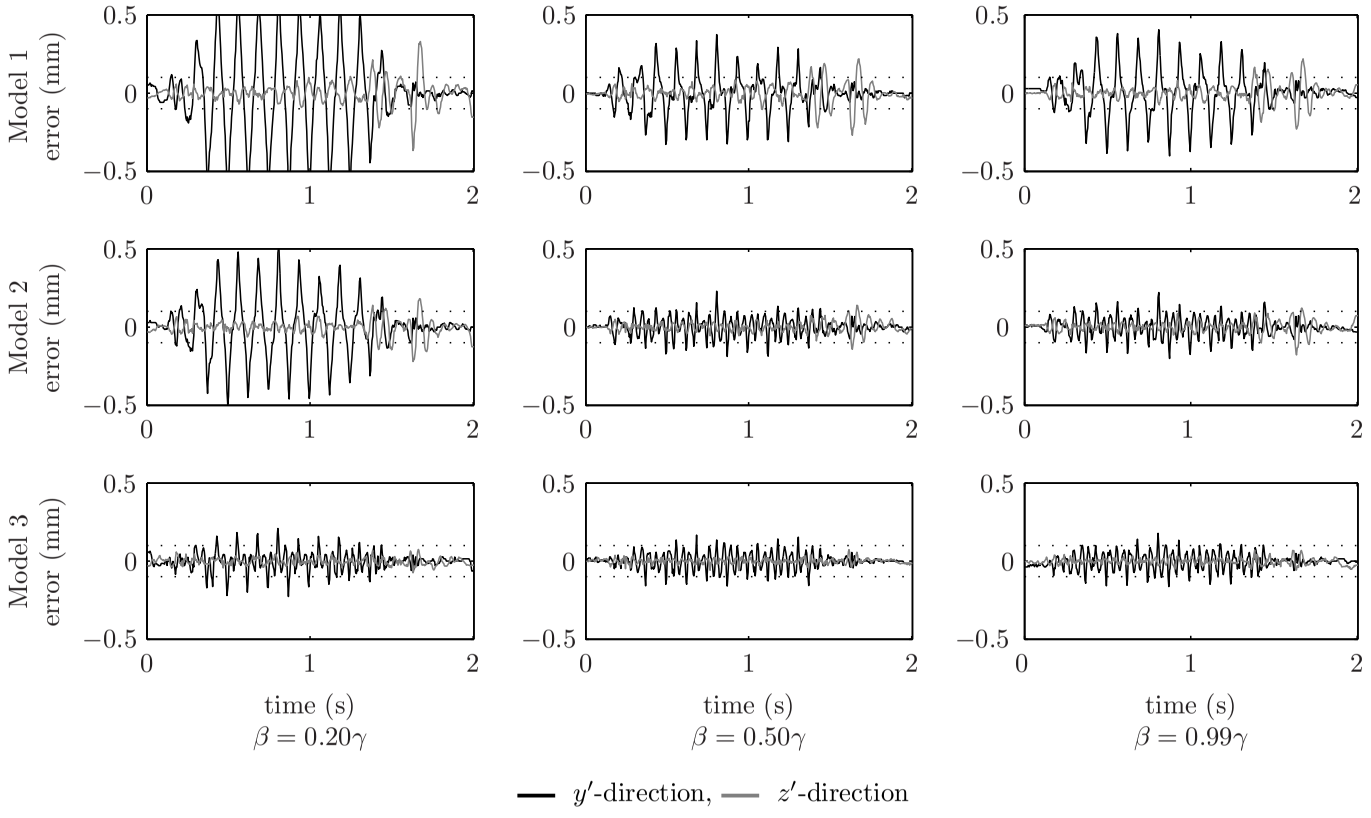


Figure C.12: Tracking error along trajectory A in iteration 9 of RILC with $\gamma = 0.99$

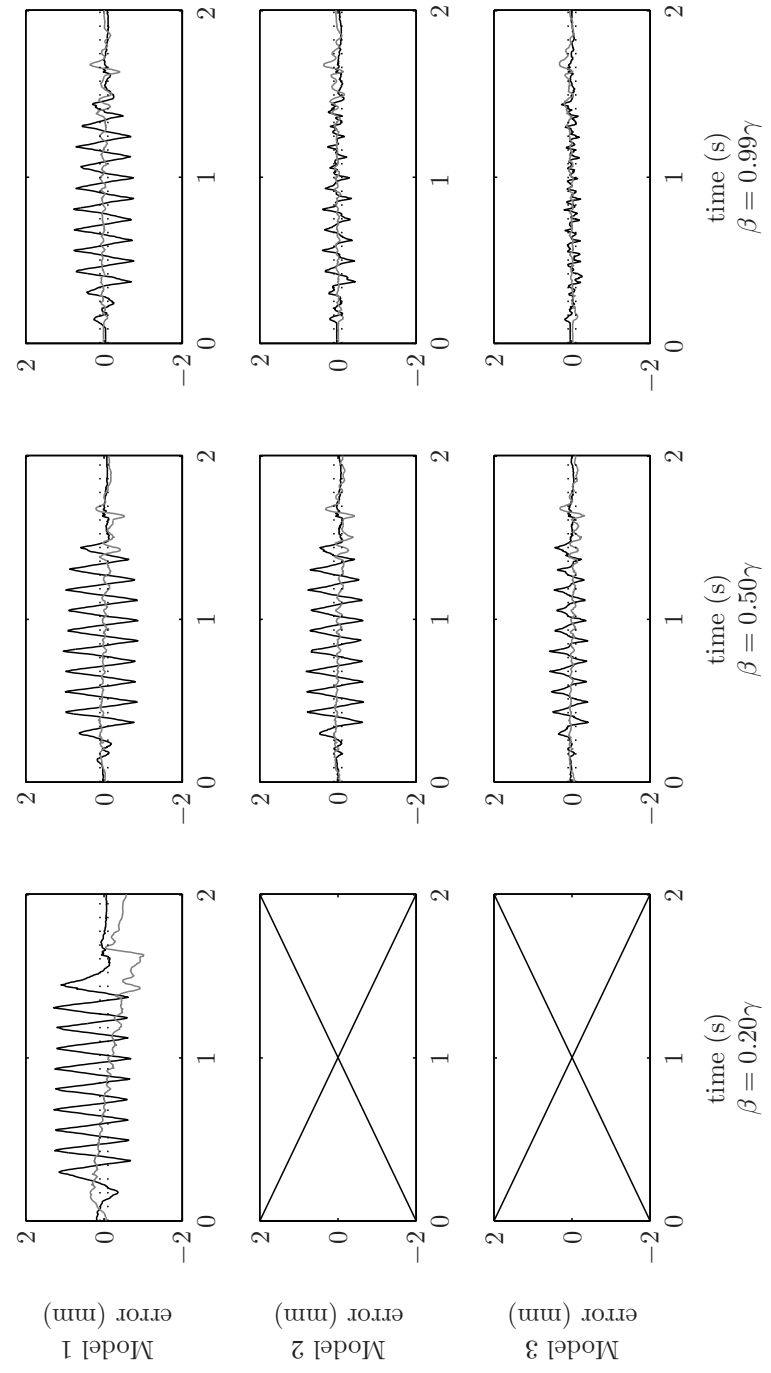


Figure C.13: Tracking error along trajectory A in iteration 1 of RILC with $\gamma = 0.75$

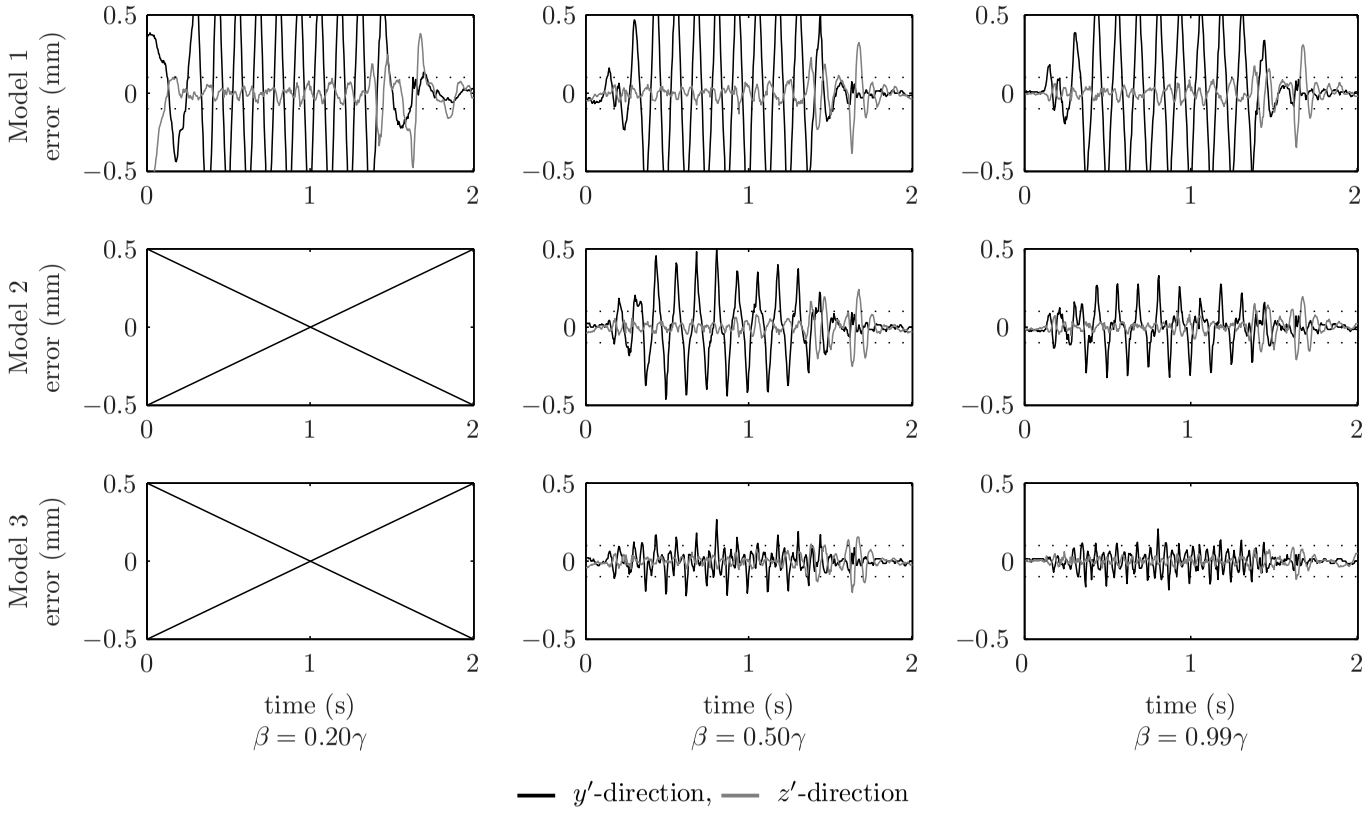


Figure C.14: Tracking error along trajectory A in iteration 9 of RILC with $\gamma = 0.75$

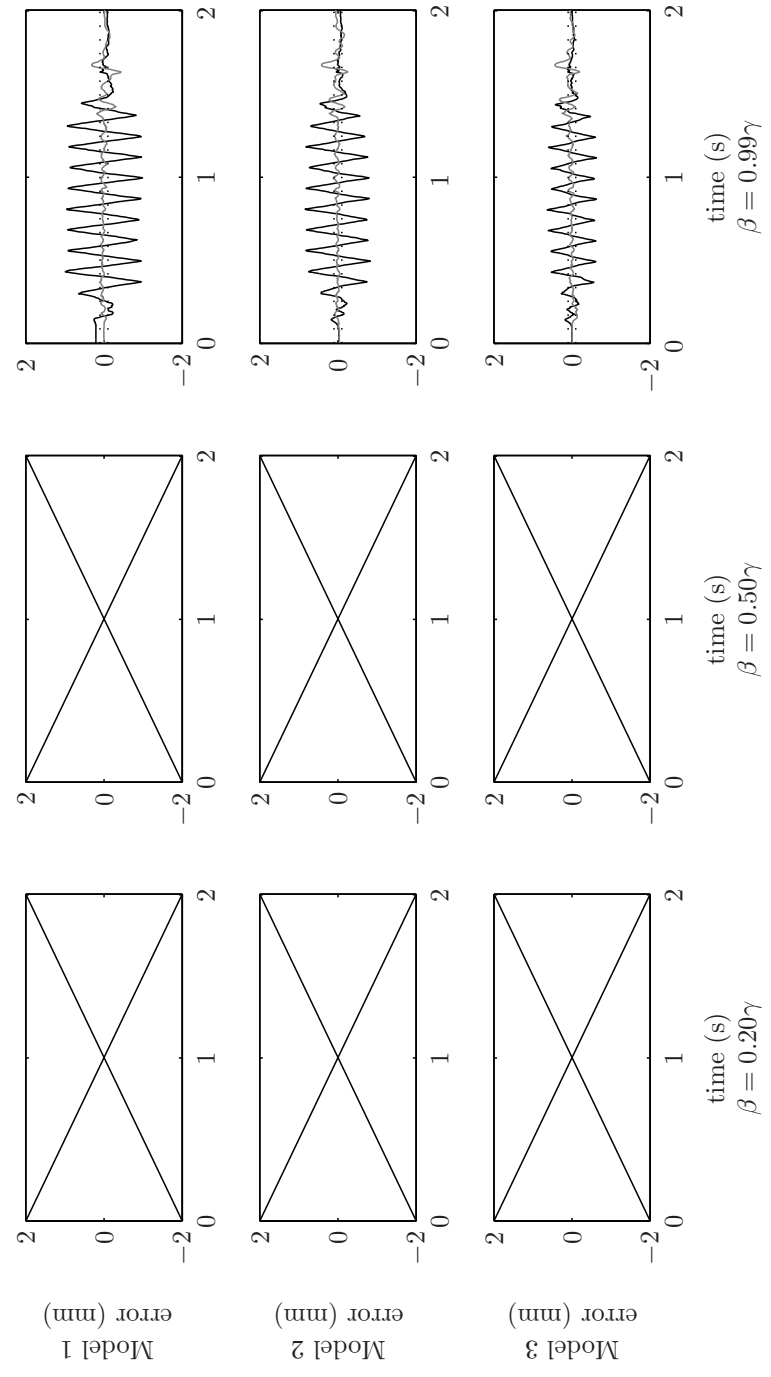


Figure C.15: Tracking error along trajectory A in iteration 1 of RILC with $\gamma = 0.50$

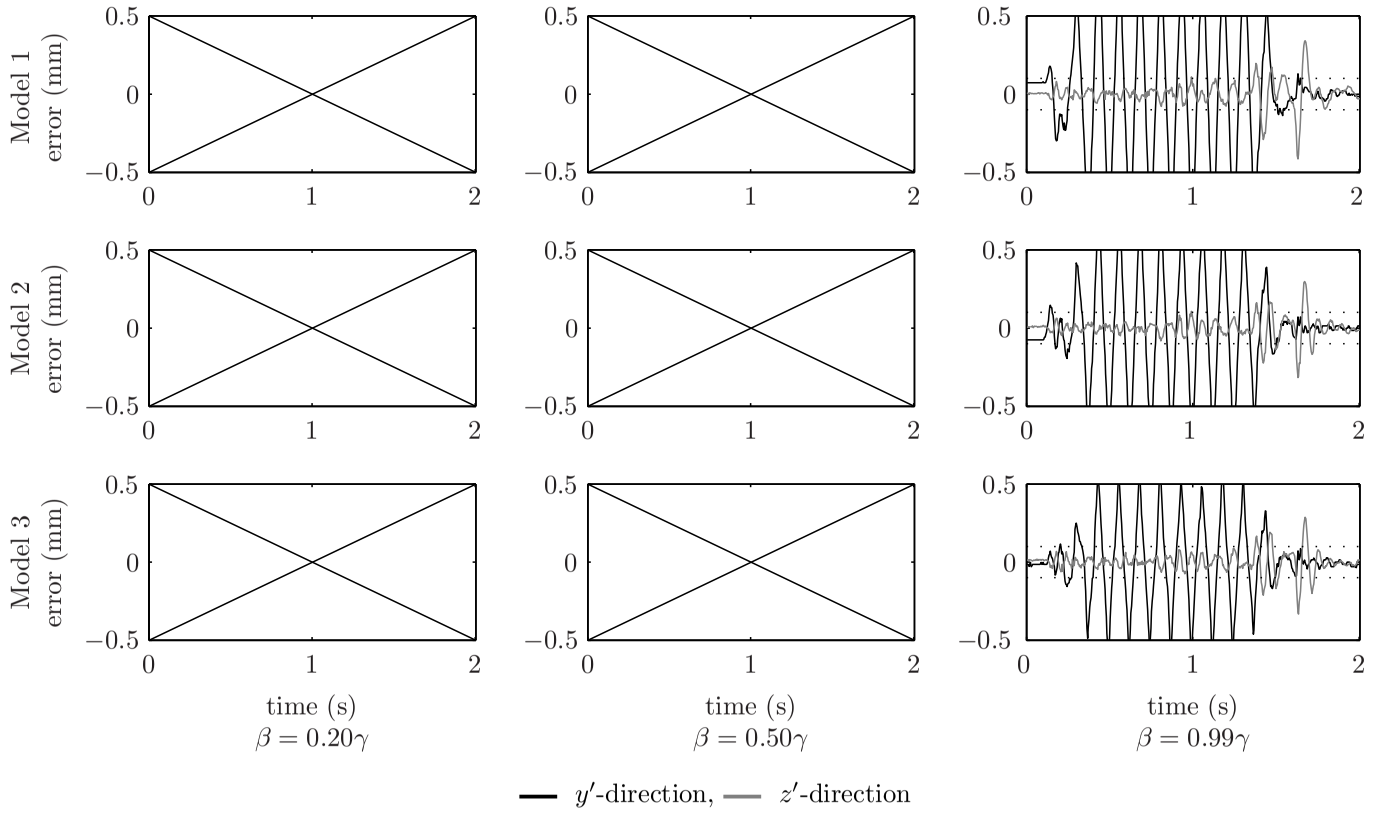


Figure C.16: Tracking error along trajectory A in iteration 9 of RILC with $\gamma = 0.50$

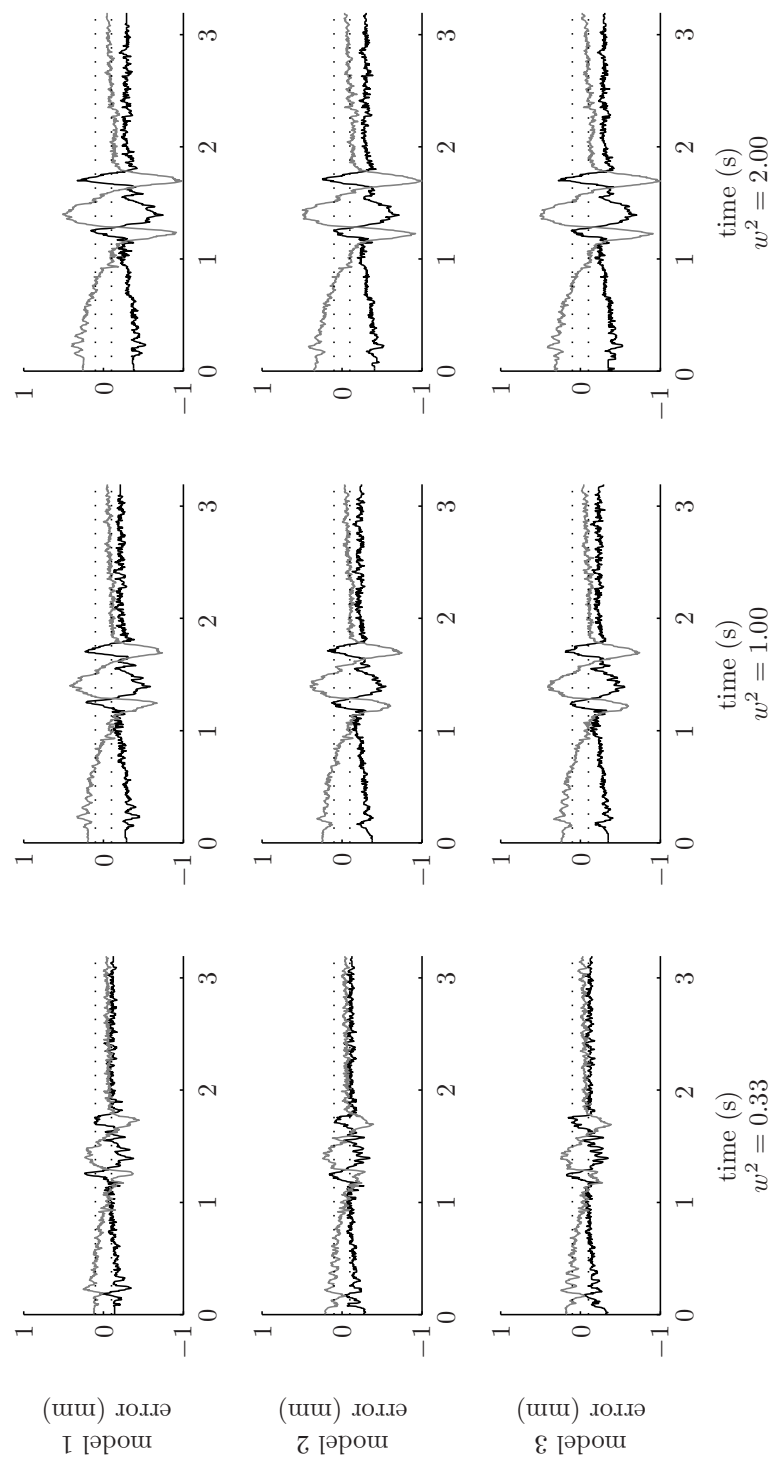


Figure C.17: Tracking error along trajectory B in iteration 1 of NILC

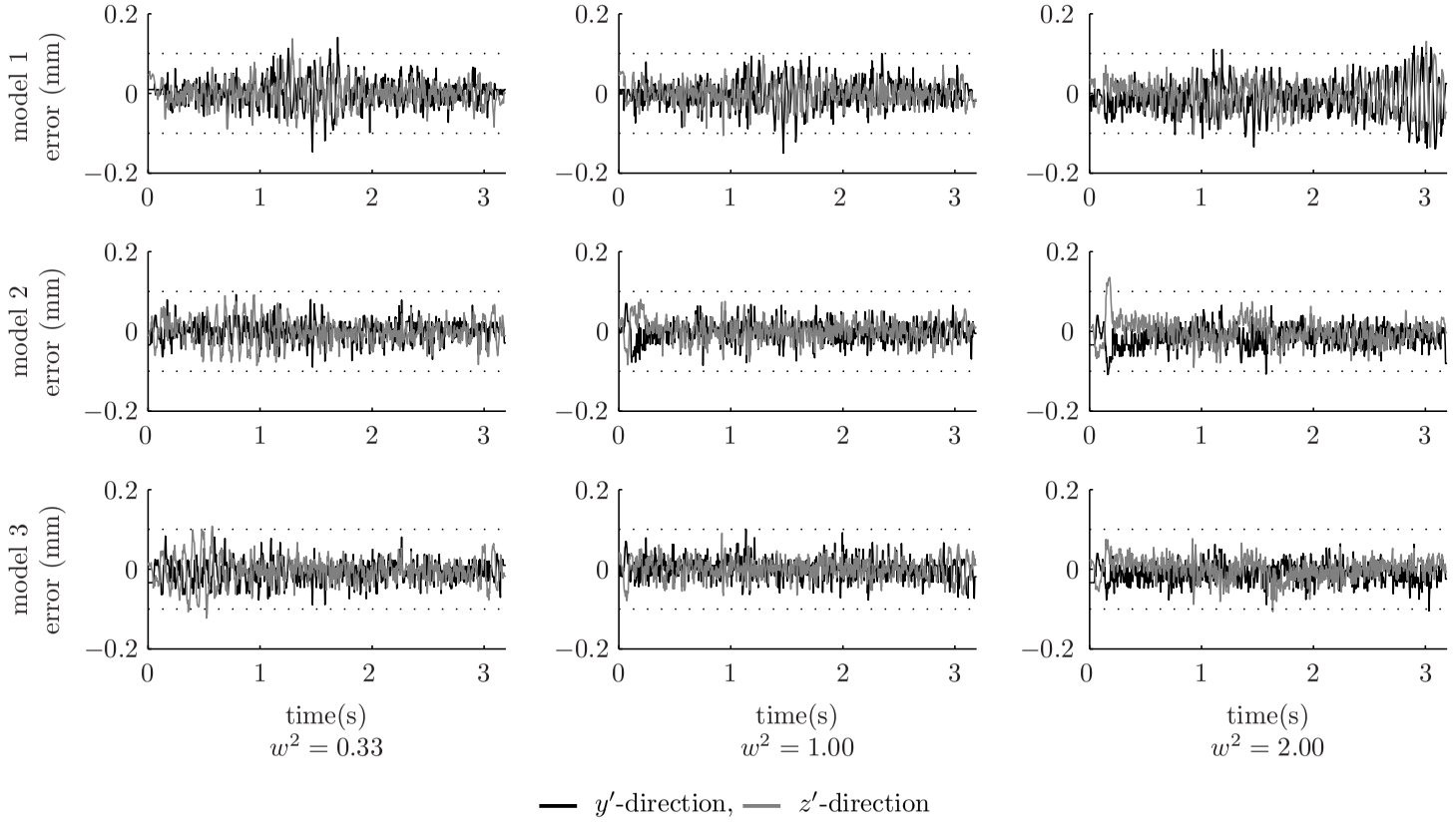


Figure C.18: Tracking error along trajectory B in iteration 9 of NILC

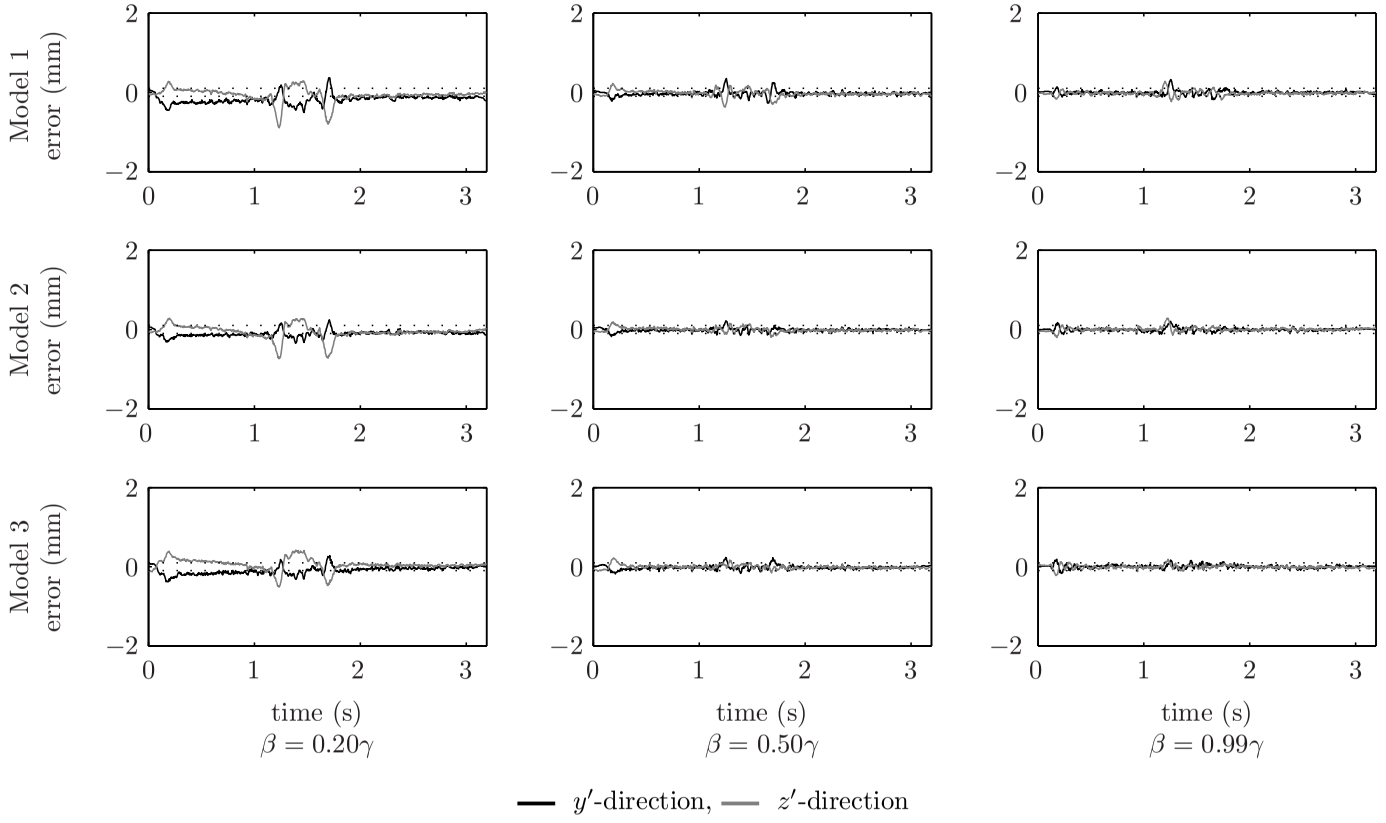


Figure C.19: Tracking error along trajectory B in iteration 1 of RILC with $\gamma = 0.99$

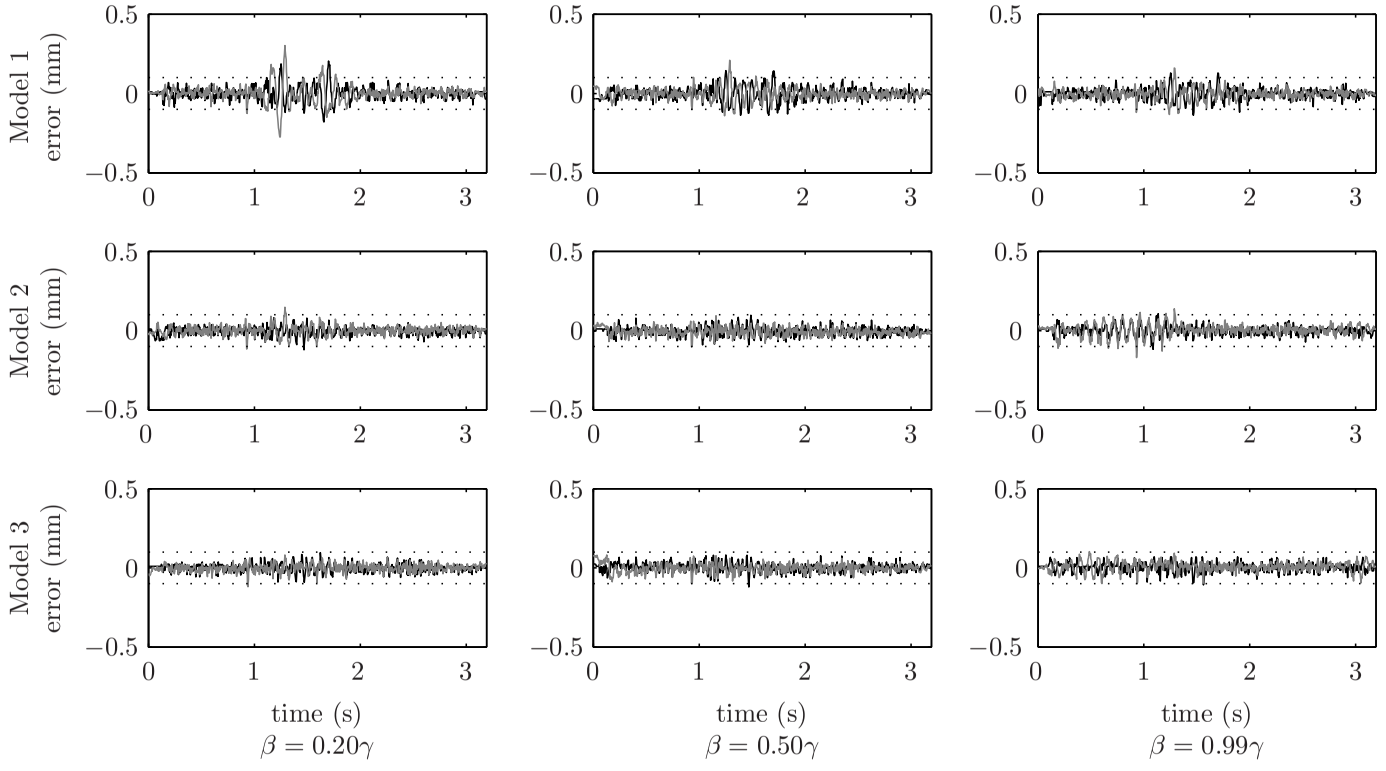


Figure C.20: Tracking error along trajectory B in iteration 9 of RILC with $\gamma = 0.99$

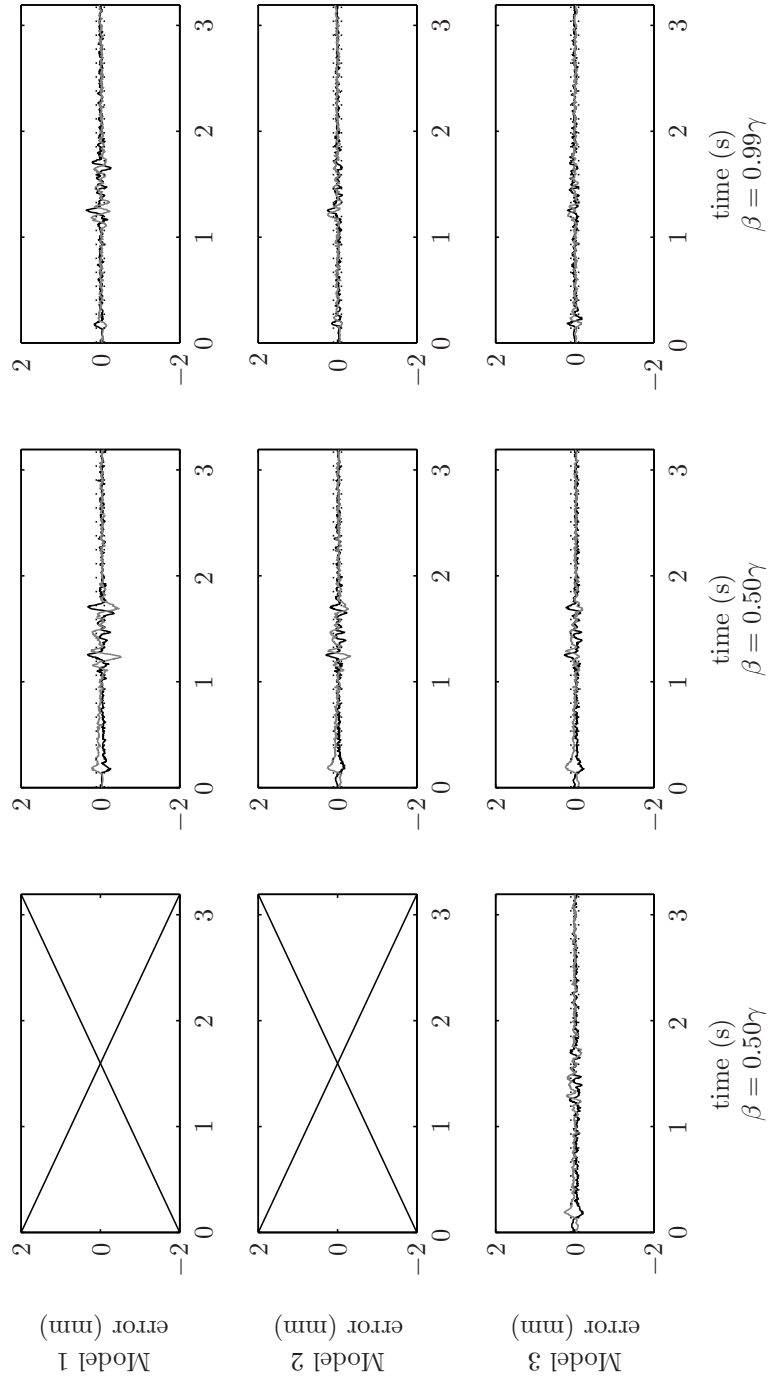


Figure C.21: Tracking error along trajectory B in iteration 1 of RILC with $\gamma = 0.75$

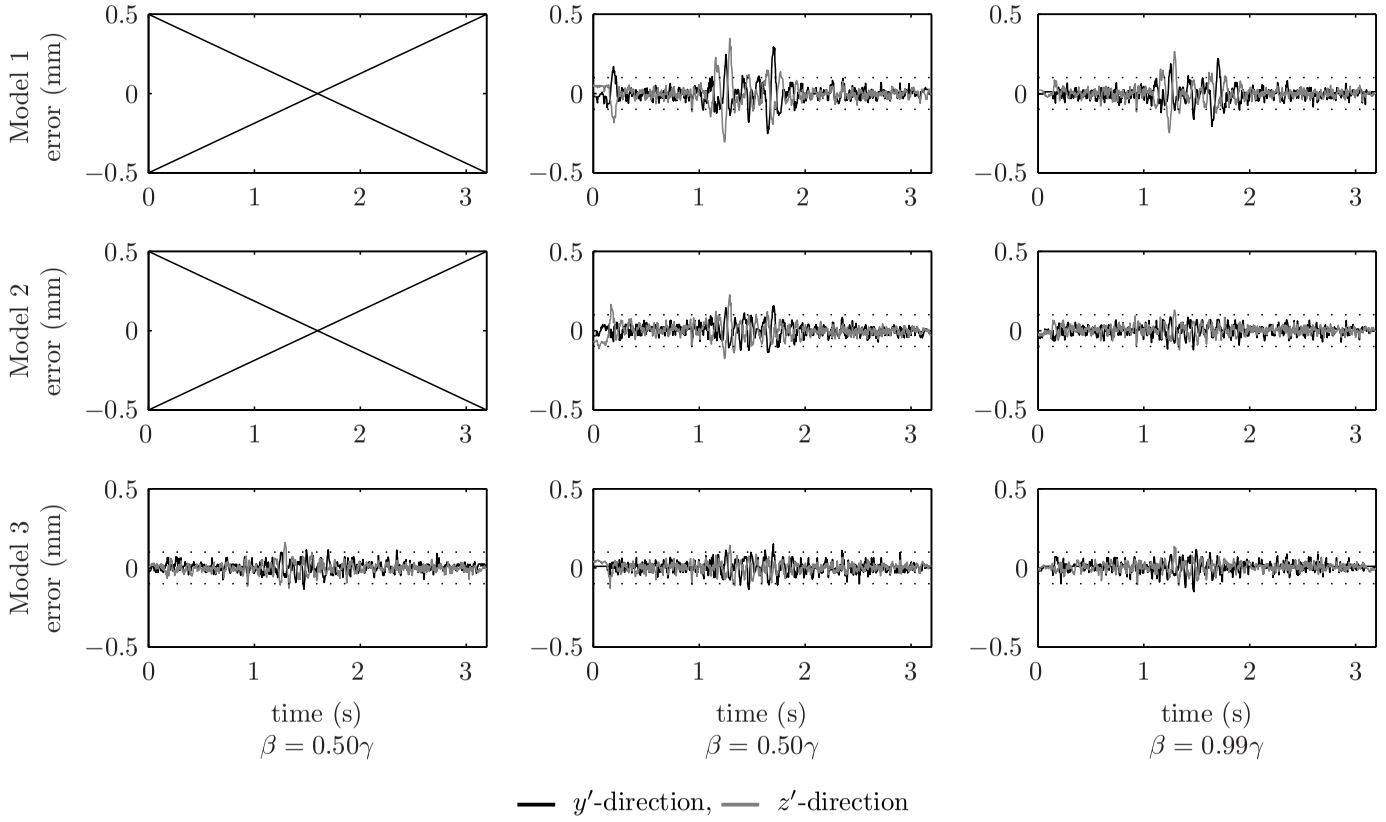


Figure C.22: Tracking error along trajectory B in iteration 9 of RILC with $\gamma = 0.75$

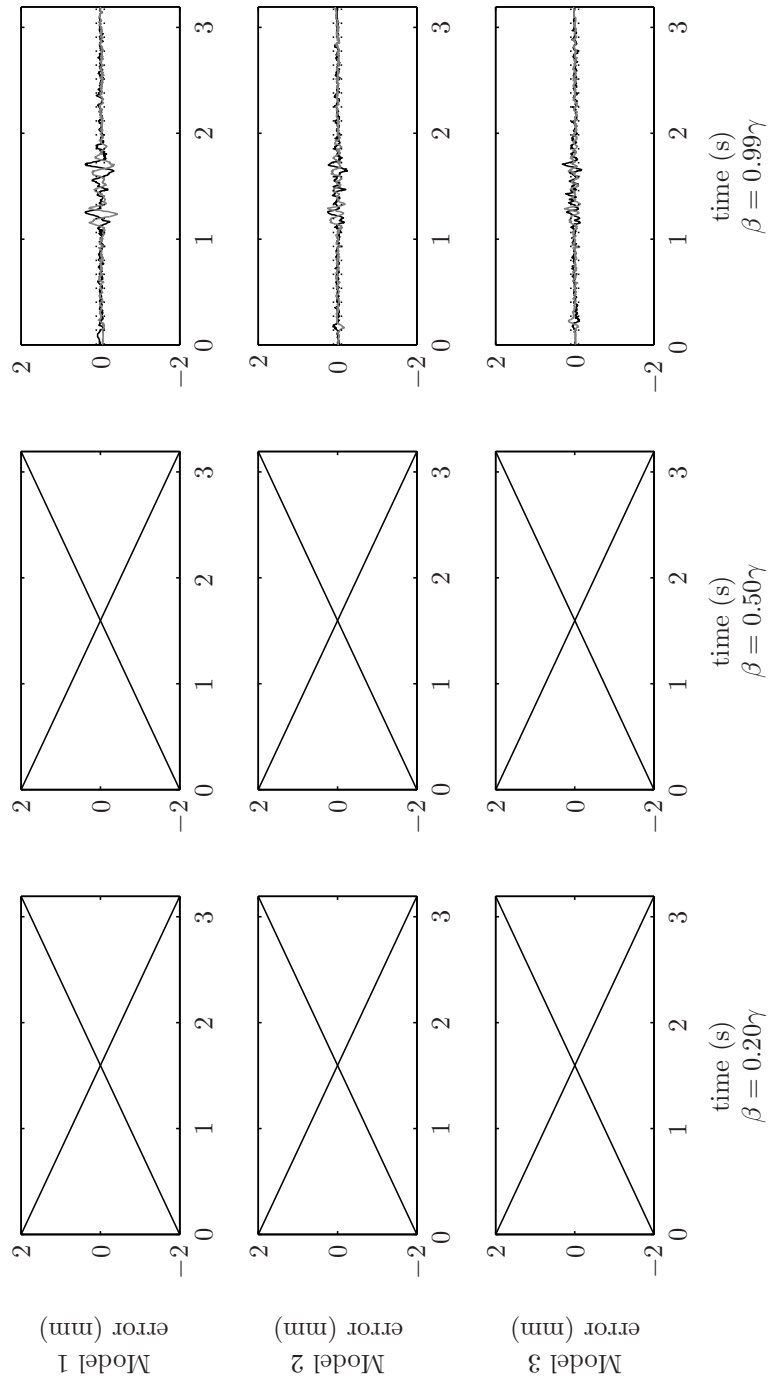


Figure C.23: Tracking error along trajectory B in iteration 1 of RILC with $\gamma = 0.50$

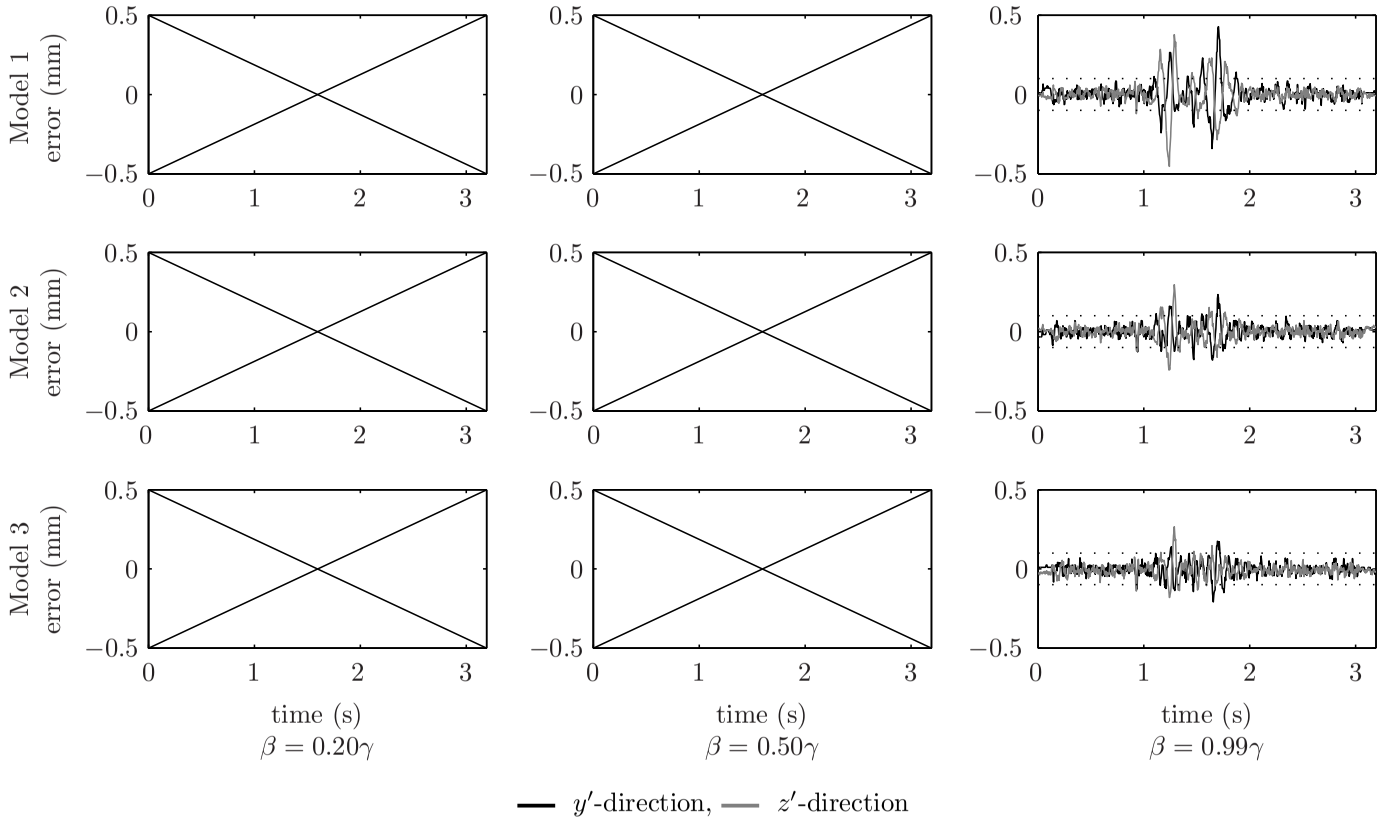


Figure C.24: Tracking error along trajectory B in iteration 9 of RILC with $\gamma = 0.50$

Publications

Hakvoort, W.B.J. , Aarts, R.G.K.M. , Dijk, J. van, and Jonker, J.B. (2006). Iterative learning control for improved end-effector accuracy of an industrial robot. In *8th International IFAC Symposium on Robot Control - SYROCO 2006 (CDROM)*, Bologna, Italy. IFAC.

Hakvoort, W.B.J. , Aarts, R.G.K.M. , Dijk, J. van, and Jonker, J.B. (2007). Model-based iterative learning control applied to an industrial robot with elasticity. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 4185–4190, New Orleans, LA, USA. IEEE.

Hakvoort, W.B.J. , Aarts, R.G.K.M. , Dijk, J. van, and Jonker, J.B. (2008). Lifted system iterative learning control applied to an industrial robot. *Control Engineering Practice*, 16(4):377–391.

Hakvoort, W.B.J. , Aarts, R.G.K.M. , Dijk, J. van, and Jonker, J.B. (2009). A computationally efficient algorithm of iterative learning control for discrete-time linear time-varying systems. *Automatica*, submitted.

Bibliography

Amann, N. , Owens, D.H. , and Rogers, E. (1996a). Iterative learning control for discrete-time systems with exponential rate of convergence. *IEE Proc.-Control Theory Appl.*, 143(2):217–224.

Amann, N. , Owens, D.H. , and Rogers, E. (1996b). Iterative learning control using optimal feedback and feedforward actions. *International Journal of Control*, 65(2):277–293.

Amann, N. , Owens, D.H. , and Rogers, E. (1998). Predictive optimal iterative learning control. *Int. J. Control*, 69(2):203–226.

Amann, N. , Owens, D.H. , Rogers, E. , and Wahl, A. (1996c). An H_∞ approach to linear iterative learning control design. *International Journal of Adaptive Control and Signal Processing*, 10:767–781.

Arif, M. , Ishihara, T. , and Inooka, H. (1999). Iterative learning control utilizing the error prediction method. *Journal of Intelligent and Robotic Systems*, 25:95–108.

Arif, M. , Ishihara, T. , and Inooka, H. (2000). Prediction-based iterative learning control (pilc) for uncertain dynamic nonlinear systems using system identification technique. *Journal of Intelligent and Robotic Systems*, 27:291–304.

Arif, M. , Ishihara, T. , and Inooka, H. (2002). Experience-based iterative learning controllers for robotic systems. *Journal of Intelligent and Robotic Systems*, 35:381–396.

Arif, M. , Ishihara, T. , and Inooka, H. (2003). A learning control for a class of linear time varying systems using double differential of error. *Journal of Intelligent and Robotic Systems*, 36:223–234.

Arimoto, S. (1990). Robustness of learning control for robot manipulators. In *IEEE International Conference on Robotics and Automation*, pages 1528–1533. IEEE.

Arimoto, S. , Kawamura, S. , and Miyazaki, F. (1984). Bettering operation of dynamic systems by learning: A new control theory for servomechanism or mechatronic systems. *Journal of Robotic Systems*, 1(2):123–140.

Arimoto, S. , Kawamura, S. , Miyazaki, F. , and Tamaki, S. (1985). Learning control theory for dynamical systems. In *Proceedings of the 24th Conference on Decision and Control*, pages 1375–1380. IEEE.

Arimoto, S. , Nguyen, P.T.A. , and Naniwa, T. (2000). Learning of robot tasks on the basis of passivity and impedance concepts. *Robotics and Autonomous systems*, 32:79–87.

Avrachenkov, K.E. (1998). Iterative learning control based on quasi-newton methods. In *proceedings of the 37th IEEE Conference on Dicision and Control*, pages 170–174, Tampa, Florida USA. IEEE.

Avrachenkov, K.E. and Longman, R.W. (2003). Iterative learning control for over-determined, under-determined, and ill-conditioned systems. *Int. J. Appl. Math. Comput. Sci.*, 13(1):113–122.

Başar, T. and Olsder, G.J. (1995). *Dynamic Noncooperative Game Theory*. Academic Press, London, UK, 2 edition.

Beigi, H.S.M. (1997). New adaptive and learning-adaptive control techniques based on an extension of the generalized secant method. *J. of Intelligent Automation and Soft Comp.*, 3(2):171–184.

Bondi, P. , Casalino, G. , and Gambardella, L. (1998). On the iterative learning control theory for robotic manipulators. *IEEE Journal of Robotics and Automation*, 4(1):14–22.

Buchheit, K. , Pandit, M. , and Befort, M. (1994). Optimal iterative learning control of an extrusion plant. In *International conference on Control*, pages 652–657. IEE.

Bukkems, B. , Kostić, D. , Jager, B. de, and Steinbuch, M. (2005). Learning-based identification and iterative learning control for direct-drive robots. *IEEE Transactions on Control Systems Technology*, 13(4):537–549.

Cheah, C.C. (2001). Robustness of time-scale learning of robot motions to uncertainty in acquired knowledge. *Journal of Robotic Systems*, 18(10):599–608.

Cheng, W. and Wen, J.T. (1993). Feedforward learning control with application to trajectory tracking of a flexible beam. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 411–416. IEEE.

Choi, J.Y. and Lee, J.S. (2000). Adaptive iterative learning control of uncertain robotic systems. *IEE Proc.-Control Theory Appl.*, 147(2):217–223.

Corke, P.I. (1996). A robotics toolbox for MATLAB. *IEEE Robotics and Automation Magazine*, 3(1):24–32.

De Luca, A. , Paesano, G. , and Ulivi, G. (1992). A frequency-domain approach to learning control: Implementation for a robot manipulator. *IEEE Transactions on Industrial Electronics*, 39(1).

De Luca, A. and Ulivi, G. (1992). Iterative learning control of robots with elastic joints. In *Proceedings of the IEEE International conference on Robotics and Automation*, pages 1920–1926, Nice, France. IEEE.

Deman, L. , Konno, A. , and Uchiyama, M. (1999). Flexible manipulator trajectory learning control with input preshaping method. In *Proceedings of the 38th SICE annual Conference*, pages 967–972, Morioka, Japan.

Devasia, S. , Chen, D. , and Paden, B. (1996). Nonlinear inversion-based output tracking. *IEEE Transactions on Automatic Control*, 41(7):930–942.

Dijkstra, B.G. (2004). *Iterative Learning control with applications to a wafer-stage.* Phd-thesis, Delft University of Technology, The Netherlands.

Dijk, J. van, Tinsel, R.B.G. , and Schrijver, E. (2001). Comparison of two iterative learning concepts applied on a manipulator with cogging force disturbances. In *IFAC Workshop on Adaptation and Learning in Control and Signal Processing (ALCOSP)*, pages 71–76, Cernobbio-Como, Italy. IFAC.

Driessen, B.J. and Sadegh, N. (2002). Multi-input square iterative learning control with input rate limits and bounds. *IEE Transactions on Systems, Man, And Cybernetics - Part B: Cybernetics*, 32(4):545–550.

Driessen, B.J. and Sadegh, N. (2004). Convergence theory for multi-input discrete-time iterative learning control with coulomb friction, continuous ouputs, and input bounds. *International Journal of Adaptive Control and Signal Processing*, 18:457–471.

Duley, W.W. (1998). *Laser Welding.* Wiley, New York.

Elci, H. , Longman, R.W. , Phan, M.Q. , Juang, J.N. , and Ugoletti, R. (2002). Simple learning control made practical by zero-phase filtering: Applications to robotics. *IEEE Transactions on circuits and systems*, 49(6):753–767.

Falldorf (2002). *Inspector Tracker, Documentation Profile Sensor.* Falldorf & Co. GmbH, Bremen, Germany, version 3.1 edition.

Fang, X. , Chen, P. , and Shao, J. (2005). Optimal higher-order iterative learning control of discrete-time linear systems. In *IEE Proc.-control Theory Appl.*, pages 43–48. IEE.

Fang, Y. and Chow, T.W.S. (1998). Iterative learning control of linear discrete-time multivariable systems. *Automatica*, 34(11):1459–1462.

French, M. , Munde, G. , Rogers, E. , and Owens, D.H. (1999). Recent developments in adaptive iterative learning control. In *Proceedings of the 38th Conference on Decision and Control*, pages 264–269, Phoenix, Arizona USA. IEEE.

French, M. and Rogers, E. (2000). Non-linear iterative learning by an adaptive lyapunov technique. *Int. J. Control*, 73(10):10.

French, M. , Rogers, E. , Wibowo, H. , and Owens, D.H. (2001). A 2d systems approach to iterative learning control based on nonlinear adaptive control techniques. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages II 429–II 432. IEEE.

Frueh, J.A. and Phan, M.Q. (2000). Linear quadratic optimal learning control. *Int. J. Control*, 73(10):832–839.

Fujimori, A. , Gunnarsson, S. , and Norrlöf, M. (2004). A gain scheduling control of nonlinear systems along a reference trajectory. Technical report LiTH-ISY-R-2654, Linköping University, Linköping, Sweden.

Fujimoto, K. and Sugie, T. (2003). Iterative learning control of hamiltonian systems: I/o based optimal control approach. *IEEE Transactions on Automatic Control*, 48(10):1756–1761.

Galkowski, K. , Lam, J. , Rogers, E. , Xu, S. , Sulikowski, B. , Paszke, W. , and Owens, D.H. (2003). Lmi based stability analyis and robust controller design for discrete linear repetitive processes. *International Journal of Robust and Nonlinear Control*, 13:1195–1211.

Ghosh, J. and Paden, B. (2004). Pseudo-inverse based iterative learning control for linear nonminimum phase plants with unmodeled dynamics. *Journal of Dynamic Systems, Measurement and Control*, 126:661–665.

Golub, G.H. and Loan, C.F. van (1996). *Matrix Computations*. The Johns Hopkins University Press, Baltimore, Maryland, USA, 3rd edition.

Gorinevsky, D. (1995). An application of on-line parametric optimization to task-level learning control. In *Proceedings of the American control Conference*, pages 862–866, Seattle, Washington, USA.

Gorinevsky, D. , Trofs, D.E. , and Goldenberg, A.A. (1997). Learning approximation of feedforward control dependence on the task parameters with application to direct-drive manipulator tracking. *IEEE Transactions on robotics and automation*, 13(4):567–581.

Graaf, M.W. de (2007). *Sensor-Guided Robotic Laser Welding*. Phd-thesis, University of Twente, Enschede, The Netherlands.

Guglielmo, K. and Sadegh, N. (1996). Theory and implementation of a repetitive robot controller with cartesian trajectory description. *Journal of Dynamic Systems, Measurement and Control*, 118(2):15–21.

Gunnarsson, S. , Norrlöf, M. , Rahic, E. , and Özbek, M. (2007). On the use of accelerometers in iterative learning control of a flexible robot arm. *International Journal of Control*, 80(3):363–373.

Gunnarsson, S. and Norrlöff, M. (2001). On the design of ilc algorithms using optimization. *Automatica*, 37:2011–216.

Hakvoort, W.B.J. , Aarts, R.G.K.M. , Dijk, J. van, and Jonker, J.B. (2009). A computationally efficient algorithm of iterative learning control for discrete-time linear time-varying systems. *Automatica*, submitted.

Hamamoto, K. and Sugie, T. (2002). Iterative learning control for robot manipulators using the finite dimensional input subspace. *IEEE Transactions on robotics and automation*, 18(4):632–635.

Hardeman, T. (2008). *Modelling and Identification of Industrial Robots including Drive and Joint Flexibilities*. Phd-thesis, University of Twente, Enschede, The Netherlands.

Harte, T.J. , Hätönen, J. , and Owens, D.W. (2005). Discrete-time inverse model-based iterative learning control: stability monotonicity and robustness. *International journal of control*, 78(8):577–586.

Hätönen, J. , Owens, D.H. , and Feng, K. (2006). Basis functions and parameter optimisation in high-order iterative learning control. *Automatica*, 42:287–294.

Hätönen, J.J. , Owens, D.H. , and Moore, K.L. (2004). An algebraic approach to iterative learning control. *Int., J. Control*, 77(1):45–54.

Hatzikos, V. , Hätönen, J. , and Owens, D.H. (2004). Genetic algorithms in norm-optimal linear and non-linear iterative learning control. *Int. J. Control*, 77(2):188–197.

Huang, S.N. , Tan, K.K. , and T.H.Lee (2002). Necessary and sufficient condition for convergence of iterative learning algorithm. *Automatica*, 38:1257–1260.

Hung, Y.S. and Yang, F. (2002). Robust H_∞ filtering for discrete time-varying uncertain systems with a known deterministic input. *International Journal of Control*, 75(15):1159–1169.

Jiang, Y.A. , Clements, D.J. , Hesketh, T. , and Park, J.S. (1994). Adaptive learning control of robot manipulators in task space. In *Proceedings of the American Control Conference*, pages 207–211, Baltimore, Maryland, USA.

Kang, J-L. , Tang, W-S. , and Mao, Y-Y. (2005). A new iterative learning control algorithm for output tracking of nonlinear systems. In *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, pages 1240–1243, Guangzhou, China.

Kavli, T. (1993). Frequency domain synthesis of trajectory learning controllers for robot manipulators. *Modeling, identification and control*, 14(3):161–174.

Kawamura, S. , Miyazaki, F. , and Arimoto, S. (1988). Realization of robot motion based on a learning method. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):126–134.

Kim, W.C. , Chin, I.S. , Lee, K.S. , and Choi, J. (2000). Analysis and reduced-order design of quadratic criterion-based iterative learning control using singular value decomposition. *Computers and Chemical Engineering*, 24:1815–1819.

Kollmorgan (2001). *VarCom Reference Guide, Kollmorgan Servostar S and Servostar CD*. Danaher Motion, USA, m-ss-001-2082 firmware version 4.1.8 and before edition.

Kurek, J.E. (2000). Counterexample to iterative learning control of linear discrete-time multivariable systems. *Automatica*, 36(2):327–328.

Kurek, J.E. and Zaremba, M.B. (1993). Iterative learning control synthesis based on 2-d system theory. *IEEE Transactions on Automatic Control*, 38(1):121–125.

Lange, F. and Hirzinger, G. (1995). Learning of a controller for non-recurring fast movements. *Advanced Robotics*, 10(2):229–244.

Lange, F. and Hirzinger, G. (1999a). Adaptive minimization of the maximal path deviations of industrial robots. In *European Control Converence ECC'99*.

Lange, F. and Hirzinger, G. (1999b). Learning accurate path control of industrial robots with joint elasticity. In *Proceedings of the 1999 IEEE international conference on robotics and automation*, Detroit, Michigan. IEEE.

Lee, K.S. , Chin, I.S. , and Lee, H.J. (1999). Model predictive control technique combined with iterative learning for batch processes. *AIChE Journal*, 45(10):2175–2187.

Lee, K.S. , Lee, J.H. , and Kim, W.C. (2000). Model-based iterative learning control with a quadratic criterion for time-varying linear systems. *Automatica*, 36:641–657.

Lewis, F.L. and Syrmos, V.L. (1995). *Optimal Control*. Wiley-Interscience, 2 edition.

Li, X-D , Ho, J.K.L. , and Chow, T.W.S. (2005). Iterative learning control for linear time-variant discrete systems based on 2-d system theory. *IEE Proc.-Control Theory Appl.*, 152(1):13–18.

Ljung, L. (1999). *System Identification - Theory For the User*. PTR Prentice Hall, Upper Saddle River, N.J., USA, 2nd edition.

Longman, R.W. (2000). Iterative learning control and repetitive control for engineering practice. *Int. J. Control*, 73(10):930–954.

Longman, R.W. , Peng, Y-T. , Kwon, T. , Lus, H. , Betti, R. , and Juang, J-N. (2003). Adaptive inverse iterative learning control. *Advances in the Astronautical Sciences*, 114:113–132.

Markusson, O. , Hjalmarsson, H. , and Norlöff, M. (2002). A general framework for iterative learning control. In *15th IFAC World Congress, Barcelona*.

Mita, T. and Kato, E. (1985). Iterative control and its application to motion control of robot arm - a direct approach to servo-problems -. In *Proceedings of the 24th Conference on Decision and Control*, pages 1393–1398, Ft. Lauderdale, FL, USA. IEEE.

Miyazaki, F. , Kawamura, S. , Matsumori, M. , and Arimoto, S. (1986). Learning control scheme for a class of robot systems with elasticity. In *Proceedings of 25th Conference on Decision and Control*, pages 74–79, Athens, Greece. IEEE.

Moore, K.L. (1998). Iterative learning control: An expository overview. *Applied and Computational Controls, Signal Processing, and Circuits*, 1(1).

Moore, K.L. , Chen, Y. Q. , and Ahn, H-S. (2005). Algebraic H_∞ design of higher-order iterative learning controllers. In *Proceedings of the 2005 IEE International Symposium on Intelligent Control*, pages 1213–1218, Limassol, Cyprus. IEEE.

Norrlöf, M. (2000). *Iterative Learning Control, Analysis, Design and Experiments*. Phd-thesis, Linköping University.

Norrlöf, M. and Gunnarsson, S. (2001). Disturbance aspects of iterative learning control. *Engineering Applications of Artificial Intelligence*, 14(1):87–94.

Norrlöf, M. and Gunnarsson, S. (2002a). An adaptive iterative learning control algorithm with experiments on an industrial robot. *IEEE Transactions on robotics and automation*, 18(2):245–251.

Norrlöf, M. and Gunnarsson, S. (2002b). Experimental comparison of some classical iterative learning control algorithms. *IEEE Transactions on robotics and automation*, 18(4):636–641.

Norrlöf, M. and Gunnarsson, S. (2002c). Time and frequency domain convergence properties in iterative learning control. *int. j. control*, 75(14):1114–1126.

Norrlöf, M. and Gunnarsson, S. (2005). A note on causal and cite iterative learning control algorithms. *Automatica*, 41:345–350.

Oh, S-R. , Bien, Z. , and Suh, I.H. (1988). An iterative learning control method with application for the robot manipulator. *IEEE Journal of Robotics and Automation*, 4(5):508–514.

Olde Benneker, J. and Gales, A. (2007). Laserlassen vs. conventionele lastechnieken. Tech-Info-blad TI.07.34, FME-CWM, Zoetermeer, The Netherlands.

Owens, D.H. , Amann, N. , Rogers, E. , and French, M. (2000). Analysis of linear iterative learning control schemes - a 2d systems/repetitive processes approach. *Multidimensional Systems and Signal Processing*, 11:125–177.

Owens, D.H. and Feng, K. (2003). Parameter optimization in iterative learning control. *International Journal of Control*, 76(11):1059–1069.

Owens, D.H. and Munde, G. (2000). Error convergence in an adaptive iterative learning controller. *International Journal of Control*, 73(10):851–857.

Pertin, F. and Bonnet-des-Tuves, J.M. (2004). Real time robot controller abstraction layer. In *Proceedings of the International Symposium on Robotics 2004*.

Pervozvanskii, A. A. and Avrachenkov, K. E. (1997). Learning control algorithms: Convergence and robustness. In *Proceedings of the 1997 Australian Control Conference*, pages 366–371.

Petsounis, K.A. and Fassois, S.D. (2000). Non-stationary functional series tarma vibration modelling and analysis in a planar manipulator. *Journal of Sound and Vibration*, 231(5):1355–1376.

Phan, M.Q. and Frueh, J.A. (1999). Model reference adaptive learning control with basis functions. In *Proceedings of the 38th Conference on Decision and Control*, pages 251–257. IEEE.

Phan, M.Q. , Longman, R.W. , and Moore, K.L. (2000). Unified formulation of linear iterative learning control. In *AAS/AIAA Space Flight Mechanics Meeting*, pages Paper No. AAS 00–106. AAS/AAIA.

Polushin, I.G. and Tayebi, A. (2004). An iterative learning control scheme for robot manipulators without velocity measurement. *IFAC Workshop on Adaptation and Learning in Control and Signal Processing*, pages 675–679.

Poo, A.N. , Lim, K.B. , and Ma, Y.X. (1996). Application of discrete learning control to a robotic manipulator. *Robotics and Computer-Integrated Manufacturing*, 12(1):55–64.

Römer, G.R.B.E. (2002). Lassen van metalen met hoogvermogen lasers. Praktijkaanbeveling PA.02.12, FME CWM, Zoetermeer, The Neterhlands.

Roover, D. de (1996). Synthesis of a robust iterative learning controller using an H_∞ approach. In *Proceedings of the 35th Conference on Decision and Control*, pages 3044–3049, Kobe, Japan. IEEE.

Roover, D. de and Bosgra, O.H. (2000). Synthesis of robust multivariable iterative learning controllers with application to a wafer stage motion system. *int. j. control*, 73(10):968–979.

Saab, S.S. (2004). A stochastic iterative learning control algorithm with application to an induction motor. *Int. J. Control*, 77(2):144–163.

Songschon, S. and Longman, R. W. (2003). Comparison of the stability boundary and the frequency response stability condition in learning and repetitive control. *Int. J. Appl. Math. Comput. Sci.*, 13(2):169–177.

Stäubli (2001). *Instruction Manual, ARM RX130B FAMILY CHARACTERISTICS*. Stäubli Faverges SCA, Faverges, France, d18320334a - 06/2001 edition.

Stäubli (2003a). *Instruction Manual, ARM RX90B FAMILY CHARACTERISTICS*. Stäubli Faverges SCA, Faverges, France, d28045104a - 05/2003 edition.

Stäubli (2003b). *Instruction Manual, CS8 Controller*. Stäubli Faverges SCA, Faverges, France, d28045604a - 03/2003 edition.

Tang, X. , Lilong, C. , and Huang, W. (2000). A learning controller for robot manipulators using fourier series. *IEEE Transactions on robotics and automation*, 16(1):36–45.

Tayebi, A. (2004). Adaptive iterative learning control for robot manipulators. *Automatica*, 40(7):1195–1203.

Tayebi, A. and Islam, S. (2006). Adaptive iterative learning control for robot manipulators: Experimental results. *Control Engineering Practice*, 14:843–851.

Tienhoven, J. van (2008). Design of hydroformed parts for laser welding. Midterm Report MC8.05204, Materials Innovation Institute, Delft, The Netherlands.

Tjepkema, D. (2008). Identification of the stäubli rx90b robot using accelerometers. Masters thesis, University of Twente, Enschede, The Netherlands.

Togai, M. and Yamano, O. (1985). Analysis and design of an optimal learning control scheme for industrial robots: a discrete system approach. In *Proceedings of the 24th IEEE conference on decision and control*, pages 1399–1404, Ft. Lauderdale, USA. IEEE.

Tomizuka, M. (1987). Zero phase error tracking algorithm for digital control. *Journal of Dynamic Systems, Measurement and Control*, 109:65–68.

Tousain, R. , Meché, E. van der, and Bosgra, O. (2001). Design strategies for iterative learning control based on optimal control. In *Proceedings of the 40th IEEE Conference on Decision and Control*, pages 4463–4468, Orlando, Florida, USA. IEEE.

Tsakalis, K.S. (1994). Performance limitations of adaptive parameter estimation and system identification algorithms in the absence of excitation. In *Proceedings of the American Control Conference*, pages 1280–1264, Baltimore, Maryland. IEEE.

Tso, S.K. and Ma, Y.X. (1992). Cartesian-based learning control for robots in discrete-time formulation. *IEEE Transactions on systems, man, and cybernetics*, 22(5):1198–1204.

Velthuis, W.J.R. , Vries, T.J.A. de, and Amerongen, J. van (1996). Learning feed forward control of a flexible beam. In *Proceedings of the 1996 IEEE International Symposium on Intelligent Control*, pages 103–108.

Wada, M. , Tsukarhara, T. , and Tsuda, K. (1993). Learning control of elastic joint robot and its application to the industrial robot manipulator. In *Proceedings of the IEEE International conference on Robotics and Automation*, pages 417–422, Piscataway, NJ, USA. IEEE.

Waiboer, R.R. (2007). *Dynamic Modelling, Identification and Simulation of Industrial Robots*. Phd-thesis, University of Twente, Enschede, The Netherlands.

Wang, D. (1995). A simple iterative learning controller for manipulators with flexible joints. *Automatica*, 31(31):1341–1344.

Wernholt, E. and Gunnarsson, S. (2006). Nonlinear identification of a physically parameterized robot model. Technical Report LiTH-ISY-R-2739, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden.

Wijdeven, J. van de and Bosgra, O. (2007a). Noncausal finite-time robust iterative learning control. In *Proceedings of the 46th Conference on Decision and Control*, New Orleans, LA, USA. IEEE.

Wijdeven, J. van de and Bosgra, O. (2007b). Robust iterative learning control. In *Book of Abstracts 26th Benelux Meeting on Systems and Control*, page 140, Lommel, Belgium.

Xu, J-X. and Xu, J. (2004). On iterative learning from different tracking tasks in the presence of time-varying uncertainties. *IEEE Transactions on Systems Man and Cybernetics*, 34(1):589–597.

Xu, J-X. and Yan, R. (2003). Fixed point theorem-based iterative learning control for ltv systems with input singularity. *IEEE Transactions on Automatic Control*, 48(3):487–492.

Yang, D.R. , Lee, K.S. , Ahn, H.J. , and Lee, J.H. (2003). Experimental application of a quadratic optimal iterative learning control method for control of wafer temperature uniformity in rapid thermal processing. *IEEE Transactions on Semiconductor Manufacturing*, 16(1):36–44.

Ye, Y. and Wang, D. (2005). Zero phase learning control using reversed time input runs. *Journal of Dynamic Systems, Measurement and Control*, 127:133–139.