ISSN 0280-5316
ISRN LUTFD2/TFRT--5858--SE

Optimal tracking and identification of paths for industrial robots

Henrik Nilsson
Björn Olofsson

Department of Automatic Control
Lund University
June 2010

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
June 2010

Document Number
ISRN LUTFD2/TFRT--5858--SE

Author(s)
Henrik Nilsson and Björn Olofsson

Supervisor
Anders Robertsson, Johan Åkesson
Automatic Control, Lund
Rolf Johansson Automatic Control, Lund (Examiner)

Sponsoring organization

Title and subtitle
Optimal tracking and identification of paths for industrial robots (Optimal följning och identifiering av banor för industrirobotar)

Abstract
In many application areas in industrial production, industrial robots are utilised for performing various tasks. Frequently a predefined geometric path exists, such that the robot should track this path with its tool centre point. The tracking is often to be performed with certain criteria specified, such as minimisation of time or energy. Accordingly, path tracking problems can often conveniently be formulated as optimisation problems. This thesis concerns the problem of time-optimal path tracking for industrial robots. The path tracking is experimentally evaluated on a robot from ABB of type IRB140 available in the Robotics Lab at the Department of Automatic Control, Lund University. In the thesis, mainly the optimisation software JModelica.org has been used for optimisation purposes. In cases where the path only is defined by a motion of a tool along a contour of an object, experimental methods are required in order to determine the corresponding geometric motion of the robot. In the thesis a contact-force control approach for determining of the joint positions along the desired path is considered. Further, in a time-optimal path tracking, one control signal is saturated in every time instance. Consequently, the robustness to modelling errors and disturbances is low. In order to make the control more robust, an earlier developed control structure called path velocity controller is implemented and tested in the robot system. Both contact-force controlled path identification and optimal path tracking are evaluated in simulations and in experiments on the robot system.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
88

Recipient's notes

Security classification

Contents

Preface

The current thesis is the final part of our education leading to a Master of Science degree in Engineering Physics. The thesis took place at the Department of Automatic Control at Lund University.

We would like to express our great gratitude to our supervisors Associate Professor Anders Robertsson and Assistant Professor Johan Åkesson, both with the Department of Automatic Control, for many interesting discussions and advice during our project. Anders has introduced us to the robot system in the Robotics Lab and helped us with robot control in general. Johan has facilitated our work with optimisation in general and JModelica.org in particular.

Further, we would like to thank those who have helped us in various ways and in different stages of the project and those who have read our report and given comments on its content and form.

The subject of this thesis is optimal path tracking for industrial robots. The background for the subject is a PhD thesis entitled *Path Constrained Robot Control* performed in the Robotics Lab at the Department of Automatic Control in the early 1990s by Ola Dahl. This thesis concerns the subject of time-optimal control of an industrial robot. In recent years, a new optimisation platform called JModelica.org has been developed at the Department of Automatic Control. With the PhD thesis by Ola Dahl and the new optimisation platform JModelica.org as foundation, a Master thesis project entitled *Optimal Control and Path Following for Industrial Robots* was done in 2008–2009 by Martin Hast concerning optimal path tracking. During that project new ideas emerged that could not be explored due to the limited time available for the project. These ideas led to the current project, which was performed as a continuation of the Master thesis by Martin Hast.

Lund, June 2010

Acronyms and symbols

Acronyms

ABB	Asea Brown Boveri
DAE	Differential Algebraic Equation
NLP	Nonlinear Program
ORCA	Open Robot Control Architecture
PVC	Path Velocity Controller
TCP	Tool Centre Point

General symbols

q	joint position vector
$\dot{q}$	joint velocity vector
$\ddot{q}$	joint acceleration vector
τ	torque vector
$M(q)$	inertia matrix
$C(q, \dot{q})$	Coriolis and centrifugal matrix
D	viscous friction matrix
$g(q)$	gravitational vector
$J(q)$	Jacobian
v	linear velocity
ω	angular velocity
T_{44}	transformation matrix
s	path parameter
$\dot{s}$	path velocity
$\ddot{s}$	path acceleration
$f(s)$	path
$f'(s)$	derivative of the path
$f''(s)$	second derivative of the path
$\Gamma_1(s)$	parameter in general rewritten robot dynamics
$\Gamma_2(s, \dot{s})$	parameter in general rewritten robot dynamics
$m(s)$	parameter in rewritten robot dynamics
$c(s)$	parameter in rewritten robot dynamics
$g(s)$	parameter in rewritten robot dynamics
η	weighting parameter in the cost function
$()^r$	reference
$\dot{()}$	first derivative with respect to time
$\ddot{()}$	second derivative with respect to time
$()'$	first derivative with respect to s
$()''$	second derivative with respect to s

Convex optimisation formulation

α	corresponds to $\ddot{s}$
β	corresponds to $\dot{s}^2$

Path velocity controller

σ	path parameter (corresponds to s)
$\dot{\sigma}$	path velocity (corresponds to $\dot{s}$)
$\ddot{\sigma}$	path acceleration (corresponds to $\ddot{s}$)
β_1	first coefficient of the parametrised control law
β_2	second coefficient of the parametrised control law
v_f	path velocity feedback
γ	adaptively updated scaling parameter
α	gain of the path velocity feedback
k	time-constant in the γ-adaptation

Force control and path identification

f	force
f_N	normal force
f_c	control signal for the force controller
M_z	torque in the z-direction
M_c	control signal for the torque controller
n	normal of the plane of the path
v_t	tangential direction
$\|\| \cdot \|\|$	2-norm

1. Introduction

The following chapter introduces concepts and ideas that are important in this thesis. Initially, the main theme, namely path tracking for industrial robots, is presented. This naturally leads on to the closely related subjects of robotics and optimisation. Further, the method that has been used in this thesis is presented. Finally, an outline of the report is given.

1.1 Path tracking for industrial robots

In a number of application areas in industrial production, industrial manipulators are utilised. A few examples of these, like moving a part from point A to point B, painting in the automotive industry and gluing in a manufacturing industry can be mentioned. In all of these application areas, the work that the robot performs means that the working environment for the employees in the industry is improved. This is especially the case in the mentioned application areas of painting and gluing, because these might have a negative health impact on the employees in the industry.

Strategy for motion planning

The problem of motion planning for industrial robots is often divided into two phases. This method is commonly referred to as the decoupled approach [LaValle, 2006]. The two phases can be stated as:

1. Planning of the path.
2. Determining of a control strategy of the robot system, such that the path is tracked as close as possible.

The reason for this approach is that the complexity of the second phase, the control phase, is reduced if a predefined path is available.

Planning phase In the first phase a path from a starting point A to an endpoint B is defined, such that the robot is supposed to follow this path with its tool centre point, abbreviated TCP. This is done without considering the entire dynamics of the robot [Verscheure, 2009]. The path can be defined as the result of a desire to move for example a tool along a contour of an object or it can be the result of a mathematical path planning process from point A to point B. In the first case, experimental methods are required in order to determine the corresponding motion of the robot. In this thesis contact-force control has been considered for this task. When the path in the latter case is planned in the space $\mathbb{R}^3$, such as physical obstacles are considered, see Figure 1.1. Also the corresponding paths for other robots within the same working area are taken into account in order to avoid collisions. However, mathematical path planning algorithms have not been examined in this thesis.

Control phase In the second phase a control strategy is determined offline such that the path that was planned in the first phase is tracked as close as possible. Thereby, the whole dynamics of the robot system has to be

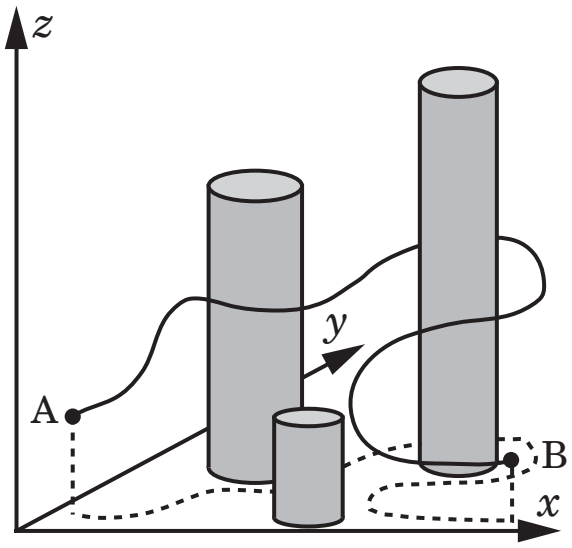

Figure 1.1 In the figure a schematic illustration of a path and obstacles in the space $\mathbb{R}^3$ can be seen.

considered [Verscheure, 2009]. This means that a model of the current robot has to be available. With the model at hand, suitable feedforward control signals can be decided in advance. These control signals are then sent to the robot system during the traverse of the path. Often, there are multiple criteria that are desirable to fulfill during the traverse of the path, *e.g.*, as a result of time and energy considerations. Hence, a suitable formulation of the path tracking problem is in the form of an optimisation problem.

Modelling errors and disturbances Because of the limited validity of the robot model and disturbances from different sources, the theoretically calculated control signals along the path will not work satisfactory on a real robot system. Therefore it is appropriate to use feedback from the measured signals of the robot as a complement to the feedforward signals. The feedback is then responsible for the real-time path tracking, in the face of modelling errors and disturbances.

Criteria that have to be considered during control

In the previous section it was mentioned that there are often certain criteria that have to be fulfilled during the control. Naturally, it is common that the robot is to traverse the path with as high speed as possible, *i.e.*, the elapsed time for the traverse from starting point to endpoint is minimised. The time consumed by the robot, in for example a manufacturing industry, is completely decisive for the production rate and therefore the income of the company. Another criterion is the energy consumed during the traverse. It is always of interest to minimise the consumption of energy in order to make the wear of the robot as low as possible.

The above specifications are of a type that can be called soft criteria because it is desirable, but not critical, that they are obeyed during the traverse of the path. However, in robot control there are also other criteria, that can be called hard criteria, which have to be considered. These criteria arise naturally as a result of the physical limitations of what is possible to achieve in the robot system in terms of control signals and internal

limitations. As an example, the motors that realise the torques on the robot joints have limitations in terms of the angular acceleration that they can realise in the robot system.

As a consequence of the mentioned specifications, the control of the path tracking can be determined as the solution of an optimisation problem. In the optimisation problem, a cost function which is to be minimised is introduced. This function typically expresses time and energy consumption. The minimisation of the cost function is subjected to certain constraints. The constraints in this application are such as the robot dynamics, physical limitations of control signals and the path itself.

1.2 Method

In order to obtain optimal path tracking for industrial robots, optimisation by means of different software devoted to solving optimisation problems numerically, has been done. The software used in the thesis will be described later. Simulation and implementation of control systems for industrial robots have been done with the tools SIMULINK/Real-Time Workshop in MATLAB. For simulation purposes, the simulation tool Dymola has also been used. Dymola uses the modelling language Modelica for formulation of the simulation model.

Further, a force control approach has been considered for identification of a path to be tracked. This strategy can be used for determining of the geometric motion of the robot when the path to be tracked is defined by a tool that is to be moved along a contour of an object. Force control allows interaction between the robot and the environment and is accordingly well suited for the task of path identification.

Execution of a control system implemented in SIMULINK on a real robot system is made possible on an IRB140 industrial robot [ABB Robotics, 2009] from the international company ABB. This robot is available in the Robotics Lab at the Department of Automatic Control, Lund University. The low level control of the robot is performed by a control cabinet of model IRC5, also from ABB. Further, the control cabinet at hand has a redesigned interface that allows execution of a control system implemented in SIMULINK. To be able to use the interface, the model built in SIMULINK has to be transformed into C-code. This procedure is automatically done by *Real-Time Workshop* in MATLAB. With the obtained C-code and corresponding object file, the control system can be executed on the real robot system and thereby experimental results can be collected. More details regarding the robot system in the Robotics Lab can be found in Appendix A.

1.3 Outline of the thesis

The thesis is organised in six main chapters and three appendices. The theoretical background for this thesis is presented in Chapter 2. This includes basic optimisation theory, robotics and solution methods for path tracking problems. Chapter 3 gives an example of a path tracking problem and illustrates the use of the solution methods presented in Chapter 2. In Chapter 4, robot modelling and path identification are discussed. Results

are presented from model identification experiments and path identification experiments on the real robot system. The second last chapter, Chapter 5, discusses optimisation of the path tracking problem in JModelica.org and presents experimental results from optimal path tracking of a contact-force identified path. The final chapter, Chapter 6, summarises this thesis and gives an outline of extensions and possible improvements of the work made in this thesis. Finally, the first appendix contains some practical details concerning the Robotics Lab and the two following appendices contain some relevant Modelica code sections and SIMULINK implementations.

2. Theory

This chapter presents the theory on which this work is founded. This includes basic optimisation theory, basic robotics and different methods to solve optimal path tracking problems. Further, a method is described that can be used to control the robot system, given an optimisation result.

2.1 Optimisation

From the introductory chapter it follows that the path tracking problem, with the additional desire that the time is minimised, leads to an optimisation problem. Generally, an optimisation problem has two parts. The first part is a function, referred to as the cost function, whose value is minimised or maximised over the free optimisation variables. The second part consists of the so called constraints that have to be considered during the optimisation. Mathematically, a general optimisation problem can be stated as below.

Definition 2.1—General optimisation problem

$$\begin{aligned} &\text{Minimise} && f(x) \\ &\text{such that} && g_i(x) \leq 0 \quad , \quad i = 1, \ldots, j \\ &&& h_i(x) = 0 \quad , \quad i = 1, \ldots, k \end{aligned}$$

where x is the optimisation variables, $f(x)$, $g_i(x)$ and $h_i(x)$ are arbitrary functions, j is the number of inequality constraints and k is the number of equality constraints. □

Convex optimisation

A special class of optimisation problems is the convex optimisation problems. Problems belonging to this class have special very tractable characteristics that make them less hard to solve than other optimisation problems. These characteristics are discussed below. For a more general treatment of the subject of convex optimisation the reader is referred to [Boyd and Vandenberghe, 2004]. In order to formulate a general convex optimisation problem the concepts of convex set and convex function have to be defined. The definitions are done according to [Boyd and Vandenberghe, 2004].

Definition 2.2—Convex set
The set A is said to be a convex set if, for every pair of points x_1 and x_2 in A, it holds that

$$\theta x_1 + (1 - \theta) x_2 \in A$$

for all $\theta \in [0, 1]$. □

Definition 2.3—Convex function
A function g is said to be a convex function if its domain D_g is a convex set and, for every pair of values x_1 and x_2 in D_g, it holds that

$$g(\theta x_1 + (1-\theta)x_2) \leq \theta g(x_1) + (1-\theta)g(x_2)$$

for all $\theta \in [0,1]$. □

The latter definition can be interpreted geometrically in two dimensions as follows: If the line segment between two arbitrary points, $(x_1, g(x_1))$ and $(x_2, g(x_2))$, on the function curve is above the function curve, then the function is convex. An example of a convex function in one dimension is the second order polynomial $g(x) = x^2$.

General convex optimisation problem A general convex optimisation problem can now be defined according to [Boyd and Vandenberghe, 2004].

Definition 2.4—Convex optimisation problem

$$\begin{aligned} &\text{Minimise} && f(x) \\ &\text{such that} && g_i(x) \leq 0 \quad , \quad i = 1, \ldots, j \\ &&& a_i^T x = b_i \quad , \quad i = 1, \ldots, k \end{aligned}$$

where x is the optimisation variables, $f(x)$ and $g_i(x)$ are convex functions, j is the number of inequality constraints and k is the number of equality constraints. □

The characteristic of convex optimisation problems is that if a minimum for the problem is found, then this minimum is a *global* minimum of the function $f(x)$. This conclusion cannot be made in the general case, because the solution in that case might be a *local* minimum. Consequently, it is desirable to formulate the optimisation problem describing the path tracking as a convex optimisation problem.

Solution of optimisation problems

Only in rare cases is it possible to obtain the solution of an optimisation problem with analytic solution methods. Therefore, the solution of an optimisation problem, both of a general optimisation problem and a convex optimisation problem, can be acquired with numerical software dedicated for solving optimisation problems. In this work mainly the software JModelica.org, see Section 2.6, has been examined. For convex optimisation problems, also another alternative numerical software has been examined that might be more efficient for this kind of problems.

2.2 Robotics

In this thesis, industrial manipulators, or more specificly serial kinematic industrial robots, have been utilised. This type of industrial robots consists of a number of joints and corresponding links, with each link serially connected to the next via each joint. Rotation of the links is realised by motors

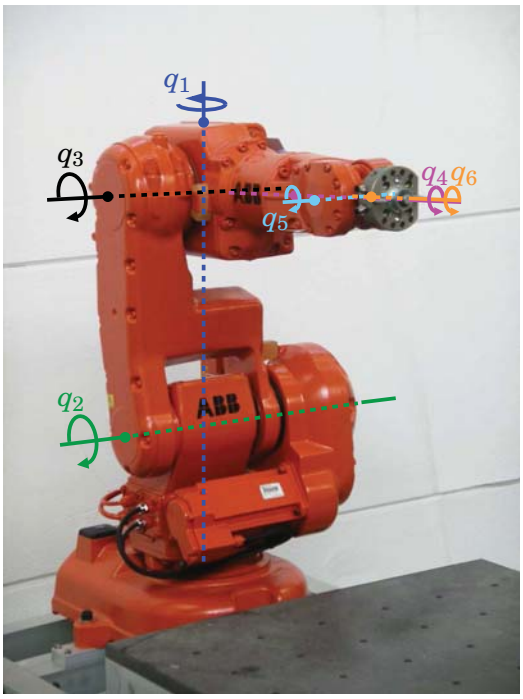

Figure 2.1 Illustration of the six joints, q_i, $i = 1, \ldots, 6$, of the ABB IRB140. In the figure the robot is in its home position, *i.e.*, the configuration where all joint positions equal to zero.

in the joints. The construction of a serial kinematic robot is often done such that six degrees of freedom is achieved. This means that an arbitrary position in the workspace of the robot can be reached from an arbitrary angle with the TCP of the robot. The robot utilised in this thesis, an ABB IRB140, can be seen in Figure 2.1, where the six joints are indicated.

Forward kinematics To describe the motion of the serial kinematic robot, a relation between the joint positions and the corresponding position and orientation of the TCP is required. This relation constitutes the forward kinematics of the robot. The forward kinematics problem is facilitated by the introduction of a number of Cartesian coordinate systems attached to each link of the robot, the base of the robot, the robot flange and the TCP [Siciliano *et al.*, 2009]. Then, the coordinates of the TCP can be expressed in for example the base coordinate system. In Figure 2.2 two of the coordinate systems, attached to the base and the flange, are shown.

Transformation matrices The connections between the Cartesian coordinate systems attached to the robot can be established by introducing transformation matrices. This is done by utilising basic linear algebra and expressing the next coordinate system as a translated and reoriented version of the former coordinate system. As an example, consider the two coordinate systems in Figure 2.3, below referred to as coordinate systems 1 and 2. Consider a vector $p_2 = [\, p_2^x \quad p_2^y \quad p_2^z \,]^T$ describing an arbitrary point P in coordinate system 2. Then introduce the vector r between the origins of the two coordinate systems, expressed in coordinate system 1. Also introduce a rotation matrix R of dimension 3×3 describing the rotation of coordinate system 2 with respect to coordinate system 1. Then the point P

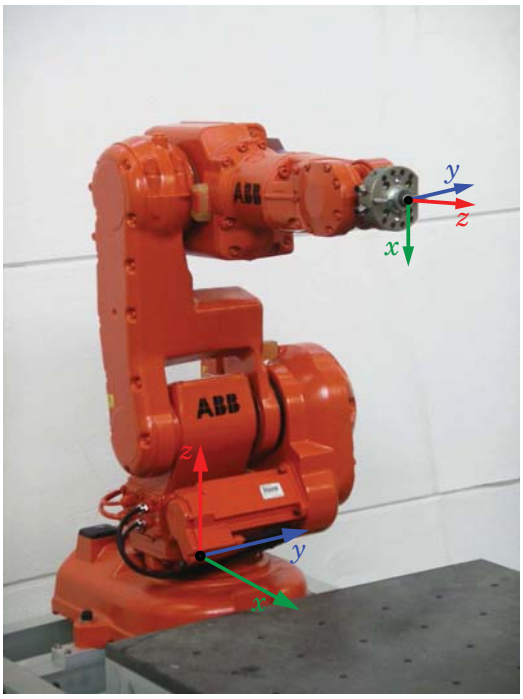

Figure 2.2 In the figure the Cartesian coordinate systems attached to the robot base and to the flange are shown.

can be expressed by a vector $p_1 = [\,p_1^x \quad p_1^y \quad p_1^z\,]^T$ in coordinate system 1 according to

$$p_1 = r + Rp_2. \tag{2.1}$$

Both the translation and the rotation can be described by a single transformation matrix of dimension 4×4, see *e.g.*, [Spong *et al.*, 2006], denoted T_{44}. In order to use the T_{44} matrix, the vector to be multiplied with the transformation matrix also has to be extended to four elements. The fourth element can be introduced as 1, without interfering with the desired transformation. The T_{44} matrix can then be established as the following block matrix

$$T_{44} = \begin{bmatrix} R & r \\ 0^T & 1 \end{bmatrix}. \tag{2.2}$$

Note that the three first elements in the fourth column in the transformation matrix express the position of the origin of the second coordinate system in the first.

Denavit-Hartenberg convention The coordinate systems are attached to the robot according to the Denavit-Hartenberg convention[1], see *e.g.*, [Spong *et al.*, 2006]. The transformation matrices between the different coordinate systems can then be decided stepwise via each joint starting from the base and ending with the robot flange or the TCP. Four coordinates are central in the Denavit-Hartenberg convention: The offset d, the length a, the twist α and the angle θ, see Figure 2.3. With these coordinates, a T_{44} matrix from the second coordinate system to the first can be

[1]Note that there are two versions of the Denavit-Hartenberg convention. The first is only called Denavit-Hartenberg while the other is called *modified* Denavit-Hartenberg.

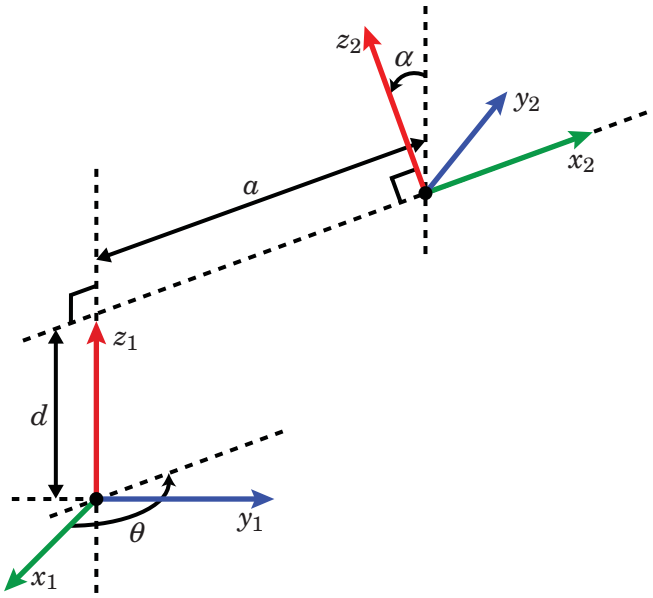

Figure 2.3 The figure illustrates the four coordinates used in the Denavit-Hartenberg convention. Firstly, a is the distance beetwen z_1 and z_2. Secondly, α is the angle beetwen z_1 and z_2. Thirdly, d is the distance from the origin o_1 to the intersection of z_1 with x_2, measured along z_1. Finally, θ is the angle between x_1 and x_2. The positive directions of the angles are also shown in the figure.

established as follows [Spong *et al.*, 2006]

$$T_{44} = \begin{bmatrix} \cos\theta & -\sin\theta\cos\alpha & \sin\theta\sin\alpha & a\cos\theta \\ \sin\theta & \cos\theta\cos\alpha & -\cos\theta\sin\alpha & a\sin\theta \\ 0 & \sin\alpha & \cos\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.3}$$

Establishment of the corresponding transformation matrices from the different coordinate systems makes it possible to translate a position or vector in one coordinate system to another coordinate system easily. As an example, a vector $p_{flange} = \begin{bmatrix} p^x_{flange} & p^y_{flange} & p^z_{flange} & 1 \end{bmatrix}^T$ expressed in the flange coordinate system can be expressed in the base coordinate system with a vector $p_{base} = \begin{bmatrix} p^x_{base} & p^y_{base} & p^z_{base} & 1 \end{bmatrix}^T$ by multiplication of the corresponding transformation matrix $T^{f\to b}_{44}$ from flange to base. Since the base is connected to the flange via the chain of robot links, this transformation matrix can be determined by stepwise multiplication of all transformation matrices from base to flange, via each joint. The final transformation can then be written according to

$$p_{base} = T^{f\to b}_{44} p_{flange}. \tag{2.4}$$

Inverse kinematics The opposite procedure of forward kinematics — *i.e.*, to determine the joint positions given the position and orientation of the TCP — is called the inverse kinematics problem. This is a much harder problem to solve, due to the existence of multiple solutions of the joint positions, given a position and orientation of the TCP [Siciliano *et al.*, 2009]. One approach for establishing the inverse kinematics is to determine the

solution pointwise and such that it in every point is chosen as the one closest to the solution in the point before. However, this strategy also fails in the singular configurations of the robot, where there are an infinite number of solutions to the inverse kinematics problem.

Jacobian The forward kinematics relation connects the joint positions to the position and orientation of for example the robot flange. In many applications it is desirable to have a corresponding relationship between the joint velocities and the velocity of the flange. This relationship is realised through the introduction of the Jacobian describing the differential kinematics of the robot [Siciliano *et al.*, 2009]. The Jacobian connects the joint velocities with the velocity of for example the flange. The flange velocity is characterised by three linear velocities $[v_x \quad v_y \quad v_z]^T$ and three angular velocities $[\omega_x \quad \omega_y \quad \omega_z]^T$, all expressed in a Cartesian coordinate system. Mathematically the relation can be written for a robot with n joints [Siciliano *et al.*, 2009]

$$v = J(q)\dot{q} \tag{2.5}$$

where $v = [v_x \quad v_y \quad v_z \quad \omega_x \quad \omega_y \quad \omega_z]^T$ is the velocity of the flange, $J(q)$ is the Jacobian of dimension $6 \times n$ and $\dot{q}$ is the $n \times 1$ vector of joint velocities. Obviously the Jacobian depends on the current configuration of the robot, *i.e.*, the joint positions q. It is also to be noted that the Jacobian loses rank in the singular configurations of the robot.

Robot model

As mentioned earlier, a demand for a good model of the robot exists when determining the path tracking. Availability of the robot model makes it possible to take the dynamics of the robot into consideration in the optimisation. This is a requirement for obtaining accurate path tracking, which uses maximum capacity of the robot. One frequently encountered robot model is the rigid body model. This model describes the relation between applied torques on the joints and the joint variables; the model can be written as [Spong *et al.*, 2006]

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + D\dot{q} + g(q) \tag{2.6}$$

where q is the vector with the joint variables, $M(q)$ is the inertia matrix, $C(q,\dot{q})$ is a matrix that describes the Coriolis and centrifugal effects, D is a matrix describing the viscous friction, $g(q)$ is a vector that describes the gravitational forces and τ is a vector with the applied torques on the robot joints. The joint variables can also be called joint positions. These two expressions will be used interchangeably in the thesis. For a revolute joint, which all the joints in the IRB140 are, each joint position $q_i,\ i = 1,\ldots,n$, is the same as the corresponding link angle θ in the Denavit-Hartenberg convention.

Robot dynamics with path tracking requirements

The robot dynamics presented above can be simplified in the case where a path is to be tracked, see *e.g.*, [Bobrow *et al.*, 1985]. Simplification in this context means that the number of states in the corresponding path tracking problem is reduced. The basis is the path, which is assumed to be expressed in the joint positions of the robot. The path is parametrised in a

so called path parameter $s(t)$. Note that the time dependency of the path parameter $s(t)$ will be implicit in the rest of the presentation for notational simplicity. The parametrised path can be written according to

$$f(s) = \begin{bmatrix} f_1(s) \\ f_2(s) \\ \vdots \\ f_n(s) \end{bmatrix} \quad , \quad s \in [s_0, s_f] \tag{2.7}$$

where n is the number of joints in the robot, s_0 is the starting point and s_f the endpoint of the parametrisation. For path tracking it is required that $q = f(s)$. With this equality the following two relations are directly obtained

$$\dot{q} = f'(s)\dot{s} \quad , \quad \ddot{q} = f'(s)\ddot{s} + f''(s)\dot{s}^2. \tag{2.8}$$

These relations are then used to rewrite the robot dynamics. Depending on the robot model used, the results are different, which means that all robot models cannot be used with all solution methods for the path tracking problem. Therefore, it is here assumed that the robot dynamics can be described by a model in accordance with the rigid body model (2.6). Then the robot dynamics can be rewritten, by using (2.8), on the following general form

$$\tau = \Gamma_1(s)\ddot{s} + \Gamma_2(s, \dot{s}) \tag{2.9}$$

stated in [Dahl, 1992].

In order to be able to formulate a convex optimisation problem that describes the path tracking problem, the even stricter assumption that the robot model used in the optimisation can be written, by using (2.8), on the following form

$$\tau = m(s)\ddot{s} + c(s)\dot{s}^2 + g(s) \tag{2.10}$$

is made in [Verscheure *et al.*, 2009]. Note that this requirement is a special case of the form (2.9). The following example shows that this reformulation is possible in a case where the robot can be described by a simple model.

Example 2.1
Assume that the robot has two joints, which can be described by a linear model. Further, the coupling between the links is assumed to be neglectable. Then the model can be written

$$\tau = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \ddot{q} = M\ddot{q}. \tag{2.11}$$

Using the path tracking requirement $q = f(s)$ *and the corresponding derivatives* (2.8) *gives*

$$\tau = M[f'(s)\ddot{s} + f''(s)\dot{s}^2]. \tag{2.12}$$

Identification of the coefficients in (2.10) *then gives*

$$m(s) = Mf'(s) \quad , \quad c(s) = Mf''(s) \quad , \quad g(s) = 0.$$

□

Remark In [Dahl, 1992] it is shown that reformulation of the robot dynamics to the form (2.9) is possible for all robot models that satisfy the rigid body model (2.6). In [Verscheure *et al.*, 2009] it is shown that reformulation to the less general form (2.10) is possible for the rigid body model (2.6) with the viscous friction term $D = 0$.

2.3 Solution methods for the path tracking problem

When the robot dynamics subjected to path tracking requirements has been formulated according to the previous section, several different methods that can be utilised to solve the path tracking problem with various criteria specified exist. In this thesis, mainly two methods have been studied. The first is the traditional phase plane method. With this method the time-optimal solution of the path tracking problem is constructed in the phase plane that is defined by the path parameter s and the corresponding path velocity $\dot{s}$. Further, methods that solve the path tracking problem as a general optimisation problem have been studied. With these methods the solution of the optimisation problem defines the solution of the path tracking problem. The phase plane method and optimisation methods will be presented in Section 2.4 and Section 2.5, respectively.

Another method for solving the path tracking problem, which can be considered as a submethod of the above mentioned methods, is to use dynamic programming in order to solve the path tracking problem in the phase plane s–$\dot{s}$ [Pfeiffer and Johanni, 1987; Shin and McKay, 1986]. This method has not been examined in this thesis, wherefore the reader is referred to the mentioned references for further details.

2.4 Time-optimal path tracking with the phase plane method

An early method to determine the time-optimal solution of the path tracking problem, with constraints on the joint torques, was to construct the solution in the phase plane that is defined by the path parameter s and its time derivative, the path velocity $\dot{s}$. Two versions of this method were presented in [Bobrow *et al.*, 1985] and [Shin and McKay, 1985]. The method has then been further developed in among others [Pfeiffer and Johanni, 1987; Shiller and Lu, 1992].

The idea of the method is to construct a solution in the phase plane — *i.e.*, determine the path velocity as a function of the path parameter — such that the path velocity is as high as possible at every point along the path. However, this must be done such that the limits on the joint torques are not violated and path tracking is achieved.

Maximisation of the path velocity is achieved by maximising or minimising the path acceleration $\ddot{s}$ at every point along the path, *e.g.*, [Bobrow *et al.*, 1985]. The points where the switches from one extreme to another take place are decided as a part of the solution algorithm. To concretise, the solution is constructed by integration in the phase plane. The integration is done with the double integrator $\ddot{s} = u$ with an input that is decided

by calculating maximum or minimum path acceleration $\ddot{s}$ at every point in the phase plane. Expressed in formulae, the system to be integrated can be written as

$$\ddot{s} = \ddot{s}_{max}(s, \dot{s}) \qquad \text{or} \qquad \ddot{s} = \ddot{s}_{min}(s, \dot{s}). \tag{2.13}$$

In the further presentation of the algorithm it is assumed that the robot model used to determine the optimal path tracking can be reformulated to the less general form (2.10). However, the phase plane algorithm can also be extended to cover the more general form (2.9).

Bounds on $\ddot{s}$ Calculations of the maximum and minimum allowed path acceleration $\ddot{s}$ at every point in the phase plane, such that the limits on the joint torques are not violated, are done by utilising the relation (2.10). Given the limits on the torques, τ_{max} and τ_{min}, and a point in the phase plane s–$\dot{s}$, the bounds can be written for each joint i, $i = 1, \ldots, n$ [Dahl, 1992]

$$\ddot{s}^i_{max} = \begin{cases} \dfrac{\tau_{min} - g_i(s) - c_i(s)\dot{s}^2}{m_i(s)} & , m_i(s) < 0 \\ \dfrac{\tau_{max} - g_i(s) - c_i(s)\dot{s}^2}{m_i(s)} & , m_i(s) > 0 \\ \infty & , m_i(s) = 0 \end{cases} \tag{2.14}$$

and

$$\ddot{s}^i_{min} = \begin{cases} \dfrac{\tau_{max} - g_i(s) - c_i(s)\dot{s}^2}{m_i(s)} & , m_i(s) < 0 \\ \dfrac{\tau_{min} - g_i(s) - c_i(s)\dot{s}^2}{m_i(s)} & , m_i(s) > 0 \\ -\infty & , m_i(s) = 0 \end{cases} \tag{2.15}$$

where subscript i for $m(s)$, $g(s)$ and $c(s)$ denotes the vector element belonging to joint i. As the constraints on the joint torques must be satisfied for all joints, the bounds $\ddot{s}_{min} = \max_i \ddot{s}^i_{min}$ and $\ddot{s}_{max} = \min_i \ddot{s}^i_{max}$ are chosen.

Maximum velocity curve Further, a curve is decided in the phase plane that determines the maximum allowed path velocity for every value of the path parameter s. All path velocities lower than the maximum are thus allowed. The allowed area in the phase plane is defined by the constraint that $\ddot{s}_{min} < \ddot{s}_{max}$ [Bobrow *et al.*, 1985]. The upper border of this area constitutes the maximum velocity curve, denoted $\dot{s}_{max}$.

Switching points When constructing the solution in the phase plane, maximum path velocity at every point along the path is strived for, as earlier mentioned. Therefore, some of the points where switches from maximum path acceleration $\ddot{s}_{max}$ to minimum path acceleration $\ddot{s}_{min}$, or vice versa, take place, will be located at the maximum velocity curve. In [Shiller and Lu, 1992] a criterion to numerically find the switching points on the maximum velocity curve during the solution with the phase plane algorithm is presented.

Algorithm With the above presented concepts and ideas, the algorithm can be constructed, which determines the time-optimal solution to the path tracking problem. In this thesis, the version that is presented in [Dahl,

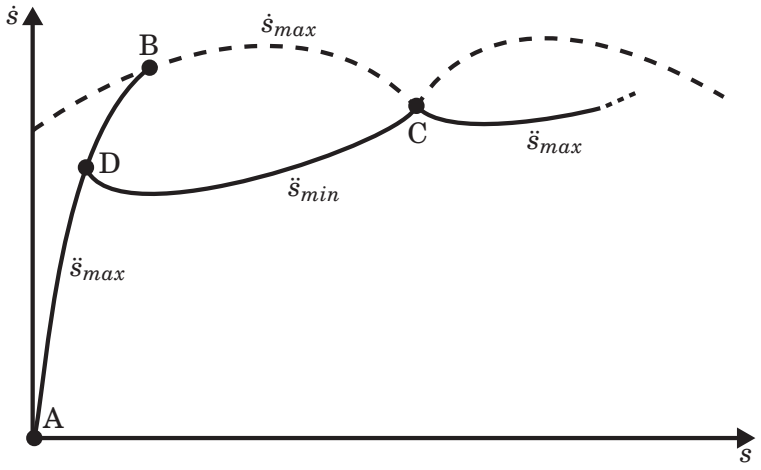

Figure 2.4 In the figure a pictorial description of the different steps in the phase plane algorithm discussed in Section 2.4 is shown.

1992] has been implemented; see examples in the next chapter. Therefore some of the main ideas from this algorithm will be presented here.

A pictorial presentation of the algorithm is seen in Figure 2.4. The letters below all refer to this figure. For simplicity, it is assumed that the path is parametrised such that $s = 0$ where the path begins and that initial rest holds. The algorithm then starts at the point $(0, 0)$ in the phase plane (A) and integrates the double integrator with maximum path acceleration $\ddot{s}_{max}(s, \dot{s})$ until the trajectory tries to leave the allowed area in the phase plane by intersecting the maximum velocity curve (B). From the intersection point (B), a search along the curve is performed for increasing values of the path parameter s until a possible switching point is found (C). From the switching point (C), the double integrator is integrated backwards in time with minimum path acceleration $\ddot{s}_{min}(s, \dot{s})$ until the former integrated trajectory is intersected (D). Thereby, another switching point is found. Then, the whole procedure is repeated, but this time starting at the switching point (C) on the maximum velocity curve. The algorithm terminates when the path parameter s reaches its final value s_f.

2.5 Formulation of an optimisation problem

With the optimisation theory and robotics that have been presented in the previous sections as foundation, an optimisation problem can be formulated, which achieves optimal path tracking for industrial robots. The advantage with a general optimisation problem compared to the above presented phase plane method is that the former is more flexible in terms of the criteria that can be taken into account when determining the optimal path tracking.

General optimisation problem When the robot dynamics has been expressed in the path parameter s and its derivatives according to Section 2.2, an optimisation problem can be formulated with these as variables. The formulation of a general optimisation problem in this thesis is done according to [Dahl, 1992] and a convex formulation is made according to [Verscheure

et al., 2009]. The cost function that is minimised consists of different components that weight different parameters. In order to obtain a time-optimal solution, the following term is used

$$t_f = \int_0^{t_f} 1 \, \mathrm{d}t = \int_{s_0}^{s_f} \frac{\mathrm{d}t}{\mathrm{d}s} \, \mathrm{d}s = \int_{s_0}^{s_f} \frac{1}{\dot{s}} \, \mathrm{d}s \tag{2.16}$$

where t_f is the time at the end of the path. Furthermore, a term that weight the derivatives of the torques is introduced in the cost function. This is motivated by the desire to reduce the wear of the robot joints. The term can be formulated similar to the formulation in [Verscheure *et al.*, 2009] according to

$$\int_{s_0}^{s_f} \sum_{i=1}^{n} |\tau_i'(s)| \, \mathrm{d}s \tag{2.17}$$

where n is the number of joints in the robot. Further, the state variable[2] $\beta(s)$ is introduced

$$\beta(s) = \dot{s}(s)^2. \tag{2.18}$$

With the cost function determined by the two terms above and the state variable $\beta(s)$ an optimisation problem can be formulated similar to [Dahl, 1992]

$$\text{minimise} \quad \int_{s_0}^{s_f} \frac{1}{\sqrt{\beta(s)}} + \eta \sum_{i=1}^{n} |\tau_i'(s)| \, \mathrm{d}s \tag{2.19}$$

$$\text{such that} \quad \tau(s) = \Gamma_1(s)\ddot{s}(s) + \Gamma_2(s, \dot{s}) \tag{2.20}$$

$$\dot{s}(s_0) = \dot{s}(s_f) = 0 \quad , \quad \beta'(s) = 2\ddot{s}(s) \quad , \quad \dot{s}(s) \geq 0 \tag{2.21}$$

$$\tau_{min} \leq \tau(s) \leq \tau_{max} \quad , \quad \beta(s) = \dot{s}(s)^2 \tag{2.22}$$

where the assumption that the robot starts and stops in rest was made. Further, the path acceleration $\ddot{s}$ serves as an input to be determined while the path parameter s serves as pseudo-time in the problem. The parameter η is chosen in order to weight the derivatives of the torques in the cost function.

Note that this optimisation formulation reduces the path tracking problem to a problem with only one state, namely the path velocity $\dot{s}$ squared, *regardless of* the number of joints in the robot model. This can be compared to an ordinary optimal control problem for a robot with six degrees of freedom that has 12 states.

Convex optimisation problem In the case where the robot model allows reformulation to the less general form (2.10), the general optimisation problem above can be reformulated according to [Verscheure *et al.*, 2009] in order to obtain a convex optimisation problem. The following two variables are considered as the optimisation variables

$$\alpha(s) = \ddot{s}(s) \ , \ \beta(s) = \dot{s}(s)^2. \tag{2.23}$$

[2]Note that the state variable is scaled by two compared to the state variable introduced in [Dahl, 1992]. This is motivated by the desire to be consistent with the convex formulation. The scaling only requires minor modifications in the optimisation formulation.

With these variables the rewritten robot dynamics can be stated as

$$\begin{aligned}\tau(s) &= m(s)\ddot{s}(s) + c(s)\dot{s}(s)^2 + g(s) \\ &= m(s)\alpha(s) + c(s)\beta(s) + g(s).\end{aligned} \tag{2.24}$$

Now a convex optimisation problem can be formulated with the same cost function as in the general case [Verscheure *et al.*, 2009]

$$\text{minimise} \quad \int_{s_0}^{s_f} \frac{1}{\sqrt{\beta(s)}} + \eta \sum_{i=1}^{n} |\tau_i'(s)| \, \mathrm{d}s \tag{2.25}$$

$$\text{such that} \quad \tau(s) = m(s)\alpha(s) + c(s)\beta(s) + g(s) \tag{2.26}$$

$$\beta(s_0) = \beta(s_f) = 0 \quad , \quad \beta'(s) = 2\alpha(s) \quad , \quad \beta(s) \geq 0 \tag{2.27}$$

$$\tau_{min} \leq \tau(s) \leq \tau_{max} \tag{2.28}$$

2.6 Solution of the optimisation problem

The preceding section introduced the method that has been used in this thesis to formulate an optimisation problem, which describes path tracking for an industrial robot. As the complexity of the optimisation problem increases, analytic solutions are not a realistic alternative. Therefore, the solution in this thesis has been determined with numerical software aimed at optimisation problems. Mainly the software JModelica.org has been utilised for solving the optimisation problem.

The optimisation software JModelica.org

A numerical software for optimisation used in the thesis is JModelica.org [Åkesson *et al.*, 2010; JModelica.org, 2010]. This software makes it possible to solve general optimisation problems, *i.e.*, the optimisation problem does not need to be convex. JModelica.org was initiated in a project at the Department of Automatic Control at Lund University [Åkesson, 2007] and is under continuous development. Hence, the current section describes its current configuration and functionality. In JModelica.org the modelling language Modelica is used to describe the dynamic properties of a system. This feature makes it possible to use a vast range of systems in JModelica.org. In Modelica models however, there is no possibility to formulate explicit optimisation problems. Therefore, optimisation problems are made possible in JModelica.org with an extension of Modelica called Optimica [Åkesson, 2008]. This extension makes it possible to define arbitrary cost functions together with arbitrary constraints. In both Modelica and Optimica, models and optimisation problems are defined in continuous time.

Model description In Modelica, and accordingly in JModelica.org, the system dynamics is described by differential algebraic equations, abbreviated DAE. A general system described by DAEs can be stated as [Åkesson, 2007]

$$F(t, x(t), \dot{x}(t), y(t), u(t)) = 0 \tag{2.29}$$

with consistent inital values for the variables given. The variable t is the independent variable, $x(t)$ is the differential variable, $y(t)$ is a pure algebraic variable since its derivative is not part of the system and $u(t)$ is the

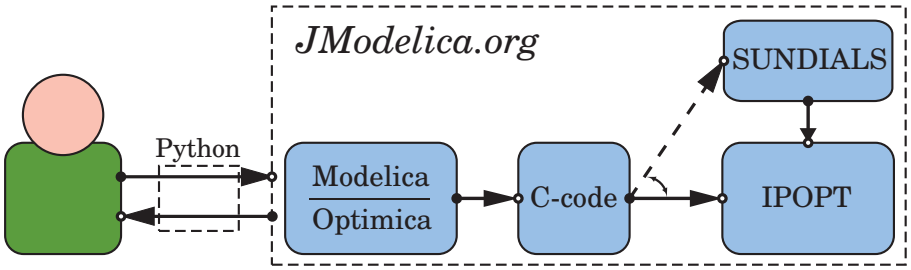

Figure 2.5 In the figure, the usage of the optimisation software JModelica.org is illustrated. The user utilises the scripting language Python in order to communicate with JModelica.org. The Modelica and Optimica models created by the user written in Modelica-code are then transformed into C-code. The C-code is compiled and used in the software IPOPT in the numerical solution. Simulation of initial values in the optimisation is possible with the software SUNDIALS.

input signal. Once the user has specified the model in Modelica and the optimisation problem in Optimica, the problem can be solved automatically by JModelica.org.

Solution of optimisation problems The solution of the optimisation problem in JModelica.org is done by transforming the original optimisation problem expressed in continuous time — *i.e.*, a problem with infinite dimension — to a problem with a large, however finite, number of optimisation variables. This process is called transcription and results in a large nonlinear program, abbreviated NLP. The specific method used in JModelica.org to make the transcription is called direct collocation. This is a simultaneous method, where all continuous variables in the optimisation problem are transformed into discrete form [Biegler *et al.*, 2002].

During the transcription a partition of the time interval in several elements is made. In every element a shape of the optimisation variables has to be assumed, *e.g.*, constant or linear. Further, constraints have to be introduced at the boundary of the elements for each differential variable in order to ensure continuous solutions. In JModelica.org, an orthogonal collocation is used, where the shapes of the variables in each element are described using Lagrange polynomials and the positions of the collocation points are chosen as the corresponding Radau points [Åkesson, 2007]. When the whole system has been discretised, the numerical solver IPOPT [Wächter and Biegler, 2006] is used in order to solve the NLP.

Technically, the solution of optimisation problems in JModelica.org is organised such that the original Modelica and Optimica models are transformed into C-code. The C-code is then compiled, whereby the result is used in the numerical solution in IPOPT [Åkesson *et al.*, 2010]. The procedure is summarised in Figure 2.5.

Interface The interface in JModelica.org for communication between the user and the software is the scripting language Python. With the interface it is possible, besides solving optimisation problems, to plot and analyse the result of the optimisation. In order to make the convergence to a solution of the optimisation problem more robust, initial values of the variables in the optimisation problem can be specified. These initial values can be obtained from simulations. Simulation in JModelica.org is possible with SUNDIALS [SUNDIALS, 2010], which is software intended for numerical integration

of systems.

Reformulation of the optimisation problem to a cone problem

In the thesis another alternative formulation of the convex optimisation problem describing the path tracking has also been studied. The reformulation is done according to [Verscheure *et al.*, 2009]. This is done such that the optimisation problem (2.25)–(2.28) presented above is rewritten in order to get constraints in the shape of second order cones. It is assumed that the stronger assumption (2.10) on the robot dynamics holds, *i.e.*, the optimisation problem is convex. The reformulation is further done such that the convexity of the optimisation problem is maintained. The reformulation also includes a transcription. This transcription has to be done manually in contrast to solving the optimisation problem in JModelica.org. The reader is referred to [Verscheure *et al.*, 2009] for the details regarding the reformulation.

In order to solve the reformulated optimisation problem, YALMIP [Löfberg, 2004] has been used in order to formulate the optimisation problem in MATLAB and then the numerical solver SDPT–3 [Toh *et al.*, 1999] has been utilised in order to solve the formulated optimisation problem.

2.7 Singular points in the path tracking problem

A problem that can arise when determining the time-optimal solution to the path tracking problem is that certain points in the phase plane s–$\dot{s}$ are singular in the optimisation [Shiller, 1994]. In order to understand this phenomenon, the parametrisation (2.10) of the torques $\tau(s)$ is studied. When one or more of the elements in the vector $m(s)$ is zero, the bounds on the torques, $\tau_{min} \leq \tau(s) \leq \tau_{max}$, for the corresponding joints only constrain the path velocity and not the path acceleration. In Section 2.4 on the phase plane method it was mentioned that during time-optimal path tracking, the path acceleration is always maximised or minimised. Therefore, a situation in a singular point can arise, where maximum or minimum path acceleration cannot be used. Instead, averaged path acceleration has to be used [Shiller, 1994]. If the algorithm determining the time-optimal path tracking does not consider the singular points, oscillations can be seen in the solution, and thereby in the joint torques. The reason is that the algorithm tries to achieve the averaged path acceleration by averaging the maximised or minimised path acceleration.

Examples of the phenomenon of singular points will be shown in a path tracking problem in Chapter 3. In that chapter, also some solutions to the problem with singular points will be discussed.

2.8 Control of the robot system

When an optimal solution has been determined with one of the methods for solving the path tracking problem, a suitable control strategy has to be decided, such that the path is traversed without violating the criteria specified. Under ideal circumstances it would be satisfying to directly apply the optimised values of the joint torques. As the model that is used in

the optimisation cannot describe all the dynamic properties of the robot system, the model is only approximate compared to the real robot system. Under mild assumptions it can be shown that in a time-optimal solution to the path tracking problem, one and only one of the joints is saturated in every time instance in terms of applied torque [Chen and Desrochers, 1989]. Thereby follows that the sensitivity to modelling errors is high and a strategy has to be introduced such that the sensitivity is reduced and that the path tracking is not violated.

Feedback Consequently, feedback is introduced in this thesis as a complement to the optimised values. The feedback is introduced according to a control structure presented in [Dahl, 1992]. The structure is called path velocity controller, abbreviated PVC, and is illustrated in Figure 2.6. In the PVC the results from the optimisation — *i.e.*, the path velocity $\dot{s}(s)$ and the path acceleration $\ddot{s}(s)$ — are used. Note that the solution obtained from the optimisation specifies these variables as a function of the path parameter s. Therefore, also in the PVC the path velocity and corresponding path acceleration are specified as a function of the path parameter s. In the PVC the notations σ, $\dot{\sigma}$ and $\ddot{\sigma}$ are used in order to describe the traverse of the path in terms of the path parameter and its time derivatives. The idea in the PVC structure is to drive this chain of integrators such that path tracking is achieved, in spite of modelling errors and disturbances, with the optimised values as a basis. The price paid in order to achieve the path tracking is that the time of the path traverse is increased.

Results from the optimisation The nominal values of $\ddot{\sigma}(\sigma)$ are obtained from the solution of the optimisation problem, namely the path acceleration $\ddot{s}$, here considered as function of σ. The nominal values, though, have to be limited due to modelling errors and other disturbances in order not to violate the hard constraints on the joint torques and consequently the path tracking. Therefore bounds on $\ddot{\sigma}(\sigma)$ are determined online during the control. When the nominal values are within the allowed interval no limitation is done, meanwhile in case of limit violation the nominal values are saturated. This saturation can violate the path tracking, because the robot system can be driven to a point where there are no allowed joint torques that retain the path tracking. Consequently, also feedback is introduced from the optimised path velocity. The control structure called PVC can be considered as built up of several different components, see Figure 2.6, which all will be presented below.

Internal controller for feedback from robot system

In the PVC a controller is used for feedback from the measured signals from the robot system. The signals consist of the joint positions q and corresponding time derivatives $\dot{q}$. The reference values to this controller are the path positions $f(\sigma)$ and the corresponding first and second derivatives, $f'(\sigma)$ and $f''(\sigma)$, with respect to the path parameter. The PVC does not specify the controller to be used in order to achieve the feedback. This gives large freedom when choosing the internal controller. The only requirement is that the control law — *i.e.*, the calculation of the joint torques τ — can be parametrised on the form [Dahl, 1992]

$$\tau = \beta_1 \ddot{\sigma} + \beta_2 \tag{2.30}$$

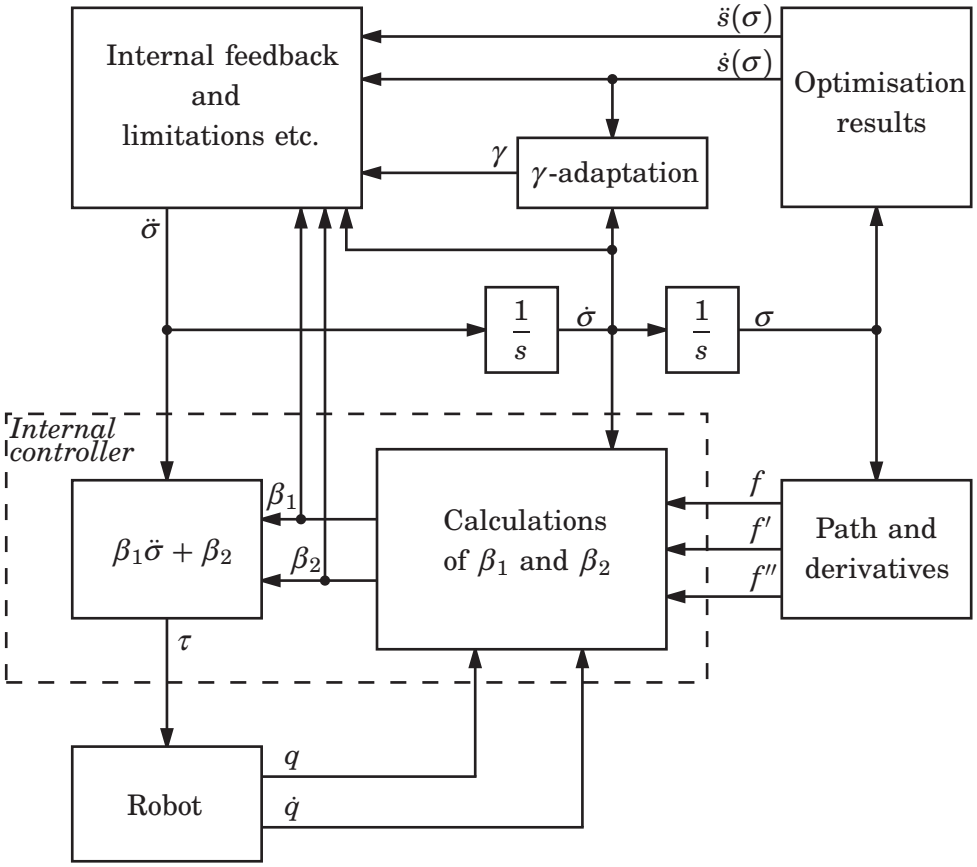

Figure 2.6 In the figure, the structure of the PVC by [Dahl, 1992] used in this thesis is illustrated.

where β_1 and β_2 do not depend on $\ddot{\sigma}$. In order to show that this parametrisation is possible, an example is given below.

EXAMPLE 2.2
Assume that the controller for the linear robot model in EXAMPLE 2.1 *is described by a control law based on passivity according to [Spong et al., 2006]*

$$\tau = M[\ddot{q}^r - \Lambda(\dot{q} - \dot{q}^r)] - K[(\dot{q} - \dot{q}^r) + \Lambda(q - q^r)] \tag{2.31}$$

where K and Λ are diagonal matrices that determine the gain and superscript r denotes the desired values of the joint positions q and their time derivatives. Insertion of the path tracking requirement $q^r = f(\sigma)$ and its time derivatives (2.8) *then gives*

$$\begin{aligned}\tau = {} & M[f'(\sigma)\ddot{\sigma} + f''(\sigma)\dot{\sigma}^2 - \Lambda(\dot{q} - f'(\sigma)\dot{\sigma})] - \\ & K[(\dot{q} - f'(\sigma)\dot{\sigma}) + \Lambda(q - f(\sigma))].\end{aligned} \tag{2.32}$$

Identification of the coefficients β_1 and β_2 thus gives

$$\beta_1 = M f'(\sigma) \tag{2.33}$$

$$\begin{aligned}\beta_2 = {} & M[f''(\sigma)\dot{\sigma}^2 - \Lambda(\dot{q} - f'(\sigma)\dot{\sigma})] - \\ & K[(\dot{q} - f'(\sigma)\dot{\sigma}) + \Lambda(q - f(\sigma))],\end{aligned} \tag{2.34}$$

which is on the specified form. □

Calculation of bounds on $\ddot{\sigma}$

From the parametrisation (2.30) of the control law for the internal controller, bounds on $\ddot{\sigma}$ can be calculated, such that each joint torque is in the allowed interval $[\tau_{min}, \tau_{max}]$. Thereby, the same idea as in the phase plane method described in Section 2.4 is used. The bounds on $\ddot{\sigma}$ can be written for each joint $i = 1, \ldots, n$ according to [Dahl, 1992]

$$\ddot{\sigma}^i_{max} = \begin{cases} \dfrac{\tau_{min} - \beta^i_2}{\beta^i_1} & , \beta^i_1 < 0 \\ \dfrac{\tau_{max} - \beta^i_2}{\beta^i_1} & , \beta^i_1 > 0 \\ \infty & , \beta^i_1 = 0 \end{cases} \tag{2.35}$$

and

$$\ddot{\sigma}^i_{min} = \begin{cases} \dfrac{\tau_{max} - \beta^i_2}{\beta^i_1} & , \beta^i_1 < 0 \\ \dfrac{\tau_{min} - \beta^i_2}{\beta^i_1} & , \beta^i_1 > 0 \\ -\infty & , \beta^i_1 = 0 \end{cases} \tag{2.36}$$

where the assumption was made that all joints have the same limitations on their torques. The calculations, though, can easily be extended to the case with different constraints on the joint torques. As the constraints on the torques τ have to be fulfilled for all joints, the limitations on $\ddot{\sigma}$ are chosen such that $\ddot{\sigma}_{min} = \max_i \ddot{\sigma}^i_{min}$ and $\ddot{\sigma}_{max} = \min_i \ddot{\sigma}^i_{max}$.

Feedback from the path velocity

From the optimisation, values of the path velocity as function of the path parameter are also obtained. These values are used in the PVC in order to increase the quality of the control of the robot system further. In [Dahl, 1992] it is shown that this feedback $\bar{v}_f(\sigma)$ can be introduced according to

$$\bar{v}_f(\sigma) = \frac{\alpha}{2}(\dot{s}(\sigma)^2 - \dot{\sigma}(\sigma)^2) \tag{2.37}$$

where α is a parameter used to choose the gain of the feedback and $\dot{s}(\sigma)$ is the optimised path velocity. This choice of feedback achieves asymptotic tracking of the optimised path velocity when the path acceleration is not saturated. The time constant in the tracking is determined by the parameter α [Dahl, 1992]. This feedback is used to adjust the nominal path acceleration $\ddot{s}(\sigma)$ such that the path tracking is not violated. More specificly, this is done by addition of $\bar{v}_f(\sigma)$ with $\ddot{s}(\sigma)$.

Scaling of the path velocity

In order to further ensure that the path tracking is achieved as well as possible, a scaling of the optimised path velocity is introduced in the PVC structure. This is motivated by the fact that the time-optimal solution in practice often have one or more points where the optimised path velocity $\dot{s}(\sigma)$ is higher than the maximum allowed path velocity due to modelling errors [Dahl, 1992], see Section 2.4 on the phase plane method. Therefore, a scaling of the optimised path velocity $\dot{s}(\sigma)$ with a parameter γ is introduced.

The scaling is updated adaptively, whereby it is updated only when the path acceleration is saturated and such that the scaling only decreases. The decrease of the parameter γ is thereby chosen such that the optimised path velocity $\dot{s}(\sigma)$ scaled with the parameter γ, converges to the current path velocity $\dot{\sigma}(\sigma)$. In [Dahl, 1992] it is shown that the scaling can be introduced with the desired properties by modifying the optimised path acceleration $\ddot{s}(\sigma)$ to

$$\ddot{s}(\sigma) \rightarrow \gamma^2 \ddot{s}(\sigma) \tag{2.38}$$

and the feedback from the optimised path velocity is modified according to

$$\bar{v}_f(\sigma) \rightarrow v_f(\sigma) = \frac{\alpha}{2}(\gamma^2 \dot{s}(\sigma)^2 - \dot{\sigma}(\sigma)^2). \tag{2.39}$$

The adaptation of the parameter γ is given by the algorithm

$$\dot{\gamma} = \begin{cases} k\dot{\sigma}(\sigma)\left[\dfrac{\dot{\sigma}(\sigma)}{\dot{s}(\sigma)} - \gamma\right] & , \gamma\dot{s}(\sigma) \geq \dot{\sigma}(\sigma) \text{ and } \ddot{\sigma} \text{ saturated} \\ 0 & , \text{otherwise} \end{cases} \tag{2.40}$$

where k is a parameter chosen to get the desired time constant in the adaptation. From the latter equation it is apparent that the parameter γ only decreases because $\dot{\gamma} \leq 0 \ \forall \ \sigma$.

Summary of the PVC structure

To summarise the section on the PVC, the control structure can be described by the following relations according to above

$$\frac{\mathrm{d}\sigma}{\mathrm{d}t} = \dot{\sigma} \tag{2.41}$$

$$\frac{\mathrm{d}\dot{\sigma}}{\mathrm{d}t} = \ddot{\sigma} \tag{2.42}$$

$$\ddot{\sigma} = \begin{cases} \ddot{\sigma}_{max} & , v(\sigma) > \ddot{\sigma}_{max} \\ v(\sigma) & , \ddot{\sigma}_{min} \leq v(\sigma) \leq \ddot{\sigma}_{max} \\ \ddot{\sigma}_{min} & , v(\sigma) < \ddot{\sigma}_{min} \end{cases} \tag{2.43}$$

$$v(\sigma) = \gamma^2 \ddot{s}(\sigma) + v_f(\sigma) = \gamma^2 \ddot{s}(\sigma) + \frac{\alpha}{2}(\gamma^2 \dot{s}(\sigma)^2 - \dot{\sigma}(\sigma)^2) \tag{2.44}$$

$$\dot{\gamma} = \begin{cases} k\dot{\sigma}(\sigma)\left[\dfrac{\dot{\sigma}(\sigma)}{\dot{s}(\sigma)} - \gamma\right] & , \gamma\dot{s}(\sigma) \geq \dot{\sigma}(\sigma) \text{ and } \ddot{\sigma} \text{ saturated} \\ 0 & , \text{otherwise.} \end{cases} \tag{2.45}$$

3. Example of a path tracking problem

In this chapter an example of a problem for optimal path tracking and the solution thereof will be presented. The solution methods discussed above in the theory chapter will all be applied on the same example in order to illustrate the use of the methods and allow comparison of the results. The result is experimentally verified on a lab process.

3.1 Path tracking problem

In this section an example of a time-optimal path tracking problem will be presented. The example is borrowed from [Dahl, 1992]. The robot has two joints, which both are modelled with a linear model and the coupling between the joints is neglected; the model can be written

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \underbrace{\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}}_{M} \ddot{q} = M\ddot{q}. \tag{3.1}$$

where m_1 and m_2 are the masses of respective link, which both are assumed to be 1. The constraints on the torques are $\tau_{min} = -1$ and $\tau_{max} = 1$. The path to be tracked is specified directly in the two joint variables according to

$$f_1(s) = 2\sin(s) \tag{3.2}$$
$$f_2(s) = 1 - \cos(s) \tag{3.3}$$

with $s \in [0, 2\pi]$. The path and the joint positions of the two robot joints can be seen in Figure 3.1. Using the result from Example 2.1, the parameters $m(s)$, $c(s)$ and $g(s)$ in the rewritten robot dynamics according to Chapter 2 are obtained as

$$m(s) = M f'(s) = \begin{bmatrix} 2m_1 \cos(s) \\ m_2 \sin(s) \end{bmatrix} \tag{3.4}$$
$$c(s) = M f''(s) = \begin{bmatrix} -2m_1 \sin(s) \\ m_2 \cos(s) \end{bmatrix} \tag{3.5}$$
$$g(s) = 0. \tag{3.6}$$

Note that this robot model satisfies the stronger assumption (2.10), which means that a convex optimisation problem can be formulated. The parameters above will be used in the solution methods in order to determine the time-optimal path tracking. The three different methods presented in Chapter 2 will be used to solve this problem.

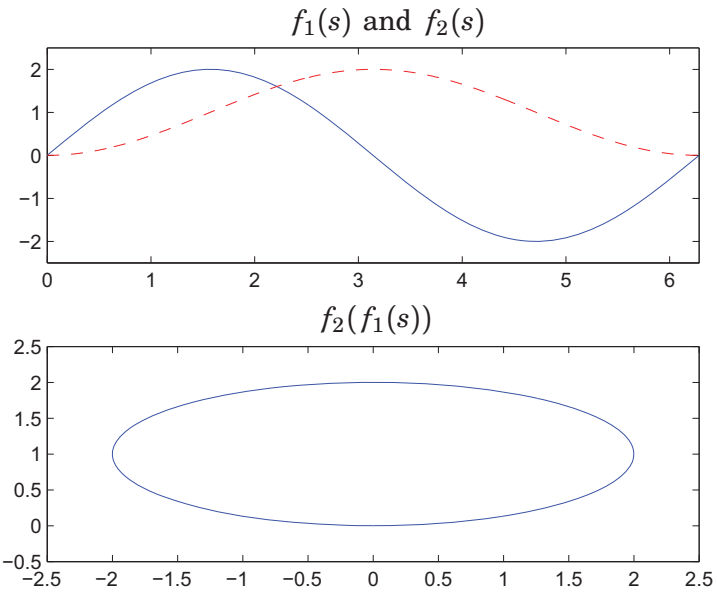

Figure 3.1 In the upper plot the joint position of joint 1 (solid blue) and joint 2 (dashed red) for the example in this chapter are shown. In the lower plot the joint position of joint 2 is plotted as a function of the joint position of joint 1.

3.2 Time-optimal path tracking with the phase plane method

In this section, the solution of the path tracking problem obtained with the phase plane method that was described in Section 2.4 will be presented. With the phase plane method, only a time-optimal solution can be decided and weighting of energy and other aspects are not possible. In the implementation the discretisation was done such that 10^5 equally spaced points are obtained in the interval $[0, 2\pi]$. In the numerical integration of the double integrator system a simple method is used, namely forward Euler. The algorithm for phase plane optimisation determines the path velocity $\dot{s}$ as function of the path parameter s, and the switching points where changes from maximum to minimum path acceleration, or vice versa, take place. Hence, the path acceleration $\ddot{s}$ is determined. As both of these quantities are determined as function of the path parameter s, the joint torques to be applied on the robot system can be calculated according to

$$\tau(s) = m(s)\ddot{s} + c(s)\dot{s}^2. \tag{3.7}$$

Splines During the solution, the two paths $f_1(s)$ and $f_2(s)$ are represented as cubic splines, *i.e.*, as piecewise third order polynomials. Even though it is not necessary to represent the current paths as splines, the problem is more general if the spline representation is chosen. This is because a realistic path is often given as a large set of data points, rather than a mathematical expression. The spline representation also makes it possible to differentiate the paths two times, which is necessary in this method. The derivatives of the path with respect to the path parameter s are decided by differentiation of the splines describing the paths.

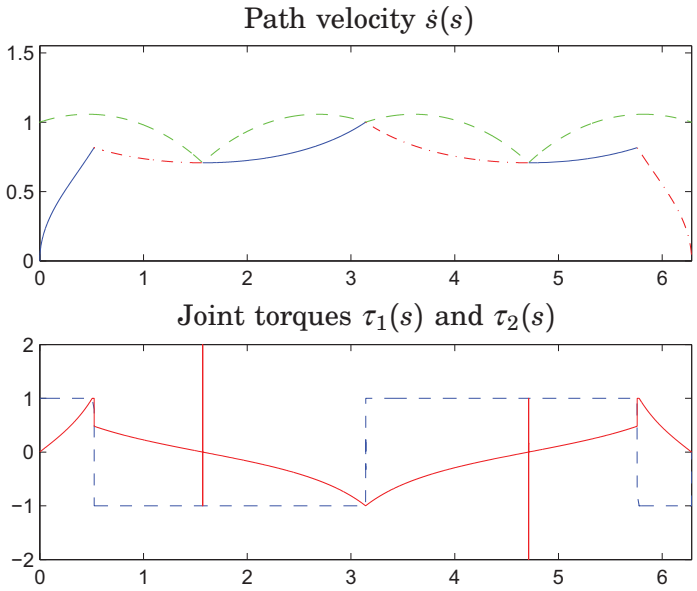

Figure 3.2 In the figure the solution obtained with the phase plane method to the path tracking problem studied in this chapter is shown. Both the path velocity and the joint torques are presented as function of the path parameter s. In the phase plane s–$\dot{s}$ in the upper plot, the trajectory parts where maximum path acceleration (solid blue) holds and the trajectory parts where minimum path acceleration (dash-dotted red) holds are indicated. Further, the maximum velocity curve (dashed green) is shown. In the lower plot, the torques τ_1 (dashed blue) and τ_2 (solid red) are shown.

Observations The solution obtained with the phase plane method is presented in Figure 3.2. In the solution a few observations can be made. At first the torques in two points are clearly outside the allowed interval and exhibit large oscillations. The reason for this behaviour is that these two points are singular in the optimisation, see Section 2.7 on singular points. In the singular points the traditional phase plane method has to be modified in order to work properly. One such modification is suggested in [Shiller and Lu, 1992]. In this thesis this modification has not been implemented. Instead, a more satisfying solution is determined with the other methods for solving the path tracking problem, which have been presented in Chapter 2.

3.3 Optimal path tracking with cone constraints

In this section the solution to the path tracking problem obtained with the optimisation problem (2.25)–(2.28) will be presented. The solution is done with the reformulation described in [Verscheure *et al.*, 2009]. Thereby a discrete optimisation problem with cone constraints is obtained. The paths are represented, in the same way as in the solution with the phase plane method, by cubic splines. In the solution 1000 points in the interval $[0, 2\pi]$ have been used in the discretisation of the optimisation problem.

Parameters In the convex optimisation problem there is a possibility to

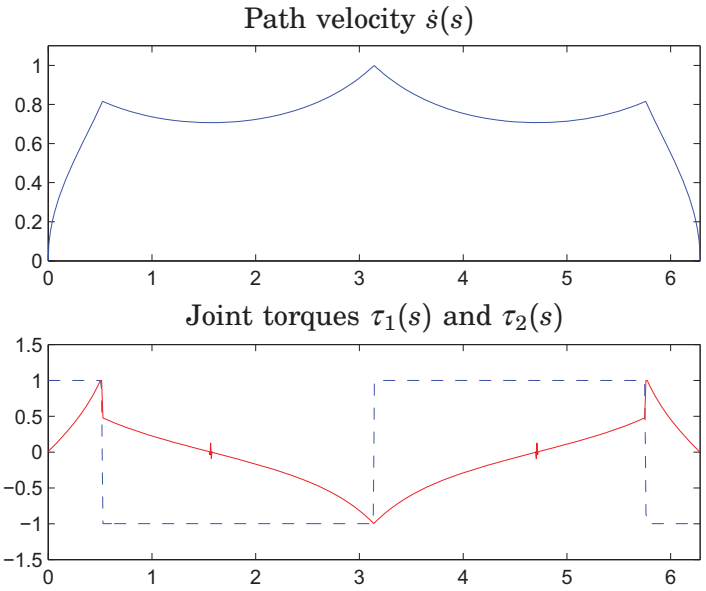

Figure 3.3 In the figure the solution obtained with cone constraints in the optimisation problem and parameter choice 1 is shown. In the lower plot, the torques τ_1 (dotted blue) and τ_2 (solid red) are shown. Note the similarity of this result to the solution obtained with the phase plane method and the behaviour at the singular points.

weight the derivatives of the torques, with the parameter η. In the solution, the following two parameter choices have been used

$$1. \quad \eta = 0 \tag{3.8}$$
$$2. \quad \eta = 10^{-6}. \tag{3.9}$$

The solution obtained when YALMIP is used to formulate the optimisation problem in MATLAB and SDPT–3 is used to solve the cone problem can be seen in Figure 3.3 for parameter choice 1 and in Figure 3.4 for parameter choice 2. The solution in both cases gives the torques τ to be applied on the joints and the variables $\alpha(s) = \ddot{s}$ and $\beta(s) = \dot{s}^2$. Hence, the path velocity and the path acceleration can be calculated from the solution.

Conclusions From the result in Figure 3.3 and Figure 3.4 certain conclusions can be drawn. The similarity of the solution obtained with the phase plane method is apparent. In the case with parameter choice 1 it is seen that problems arise in the singular points, where the joint torques τ oscillate. This is obviously not satisfactory. Therefore the derivatives of the joint torques are weighted with the parameter η in parameter choice 2 according to [Verscheure *et al.*, 2009]. It is obvious that the weighting is an efficient way to eliminate the oscillations in the singular points.

3.4 Solution of the optimisation problem with JModelica.org

In this section the solution of the optimisation problem describing the path

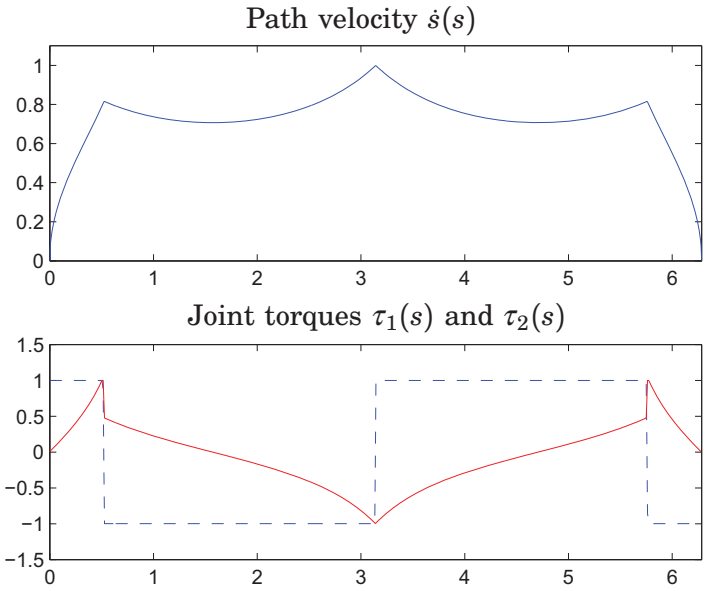

Figure 3.4 In the figure the solution obtained with cone constraints in the optimisation problem and parameter choice 2 is shown. In the plot below, the torques τ_1 (dotted blue) and τ_2 (solid red) are shown. Note in particular that the weighting of the derivatives of the torques with the parameter η makes the result also satisfactory at the singular points.

tracking obtained with JModelica.org will be presented. The optimisation problem to solve is specified, precisely as in the case with the cone constraints above, by (2.25)–(2.28). The parameter η in the cost function has been selected according to

$$\eta = 0, \tag{3.10}$$

i.e., pure time-optimality is strived for. This section also serves as an illustration of how optimisation problems are solved in JModelica.org. Therefore the code solving the path tracking problem will be presented. The optimisation problem in JModelica.org is specified in continuous variables. The transcription to discrete form is then made automatically by the software. The user, though, has to specify the number of elements that the continuous problem will be divided into and how many points in every element that will be used in the collocation. In this example 200 elements have been used in the interval $[0, 2\pi]$ and in every element three collocation points have been selected.

Spline implementation In order to represent the paths as function of the path parameter, cubic splines are used in the same way as in the other solution methods presented above. As splines are not yet supported by JModelica.org, a separate Modelica model has been created, which provides the value of a spline in a certain point. The implementation of the model is done with if- and else-clauses. The design of the model is in the shape of an exponential search tree, wherefore the search for correct spline value can be made with a binary search. The creation of the Modelica model is facilitated by a MATLAB-script modified during the thesis, but originating from the original Optimica compiler.

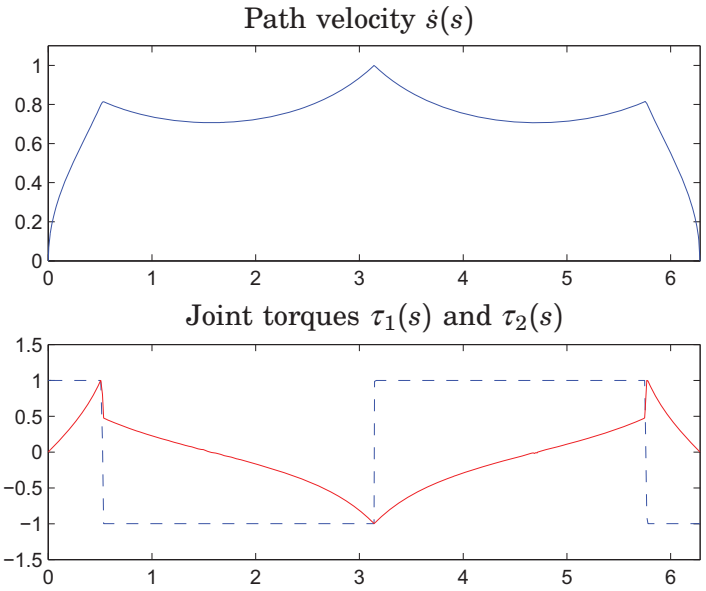

Figure 3.5 In the figure the solution obtained with JModelica.org of the optimisation problem describing the path tracking problem is shown. In the lower plot, the torques τ_1 (dashed blue) and τ_2 (solid red) are shown.

Modelica code In Appendix B.1 the code formulating the optimisation problem in Modelica and Optimica is presented in its entity. The robot dynamics subjected to path tracking requirements is described by a Modelica-model called `robDyn`, meanwhile the optimisation problem as such is described by `ellips_Opt`.

In the code in Appendix B.1 it is seen that there is a Modelica model called `ellips_Init_Opt`, that is used in order to determine initial values to the main optimisation. The convergence of the optimisation is more robust if good initial values have been found beforehand. The dilemma is, of course, to know what good initial values are when the correct solution is not known. In the current optimisation it turns out that it is enough to determine an initial guess where the path acceleration, or the variable $\alpha(s)$, is constant k in the interval $s \in [0, \pi]$ and then constant $-k$ in the interval $s \in [\pi, 2\pi]$. With this input — *i.e.*, the path acceleration — the system is simulated with SUNDIALS according to the description in Section 2.6. The obtained values of the variables are then initialised in the optimisation.

Solution The solution obtained when JModelica.org solves the optimisation problem defining the path tracking problem can be seen in Figure 3.5. It is apparent that this solution method gives the same result as the other methods for this example. Further, this solution method has, for the current example, no problems in the singular points, even though weighting of the derivatives of the torques is not introduced.

3.5 Experimental verification

In order to experimentally verify the solution of the optimal path tracking problem studied in this chapter, a suitable process that can be modelled as

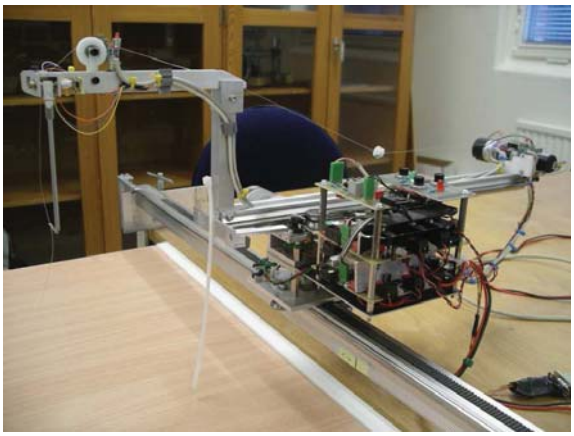

Figure 3.6 In the figure the crane process used for experimental verification of the path tracking example studied in this chapter is shown.

two double integrators from input to output in accordance with (3.1) has to be chosen. In this thesis the gantry crane process [Larsson and Braun, 2008] has been used. The crane, see Figure 3.6, consists of a cart that is used to position the pivot point of the load. The load is attached to the cart via a string. The position of the pivot point is altered by moving the cart along two perpendicular rails, aligned in the x- and y-directions in a Cartesian coordinate system respectively. Also, the height of the load can be controlled. Since only positioning of the pivot point is applicable for the purpose of path tracking in this thesis, the load was not attached to the cart during the experiments.

Hardware

The process is connected to a PC via a RS232 communication port. This communication channel can be used in a controller implemented in SIMULINK with a sampling rate of $h = 0.01$ s. The position of the cart in the two rail directions can be measured. The corresponding velocities are approximated in the SIMULINK model by applying a high-pass filter on the position signals. In order to control the position of the cart along the rails, the acceleration in each rail direction is used as control input to the system. The acceleration signals given by the user are then integrated in order to get the corresponding velocities. Two PI controllers implemented in the SIMULINK model are responsible for the velocity control of the cart in the two rail directions. The controllers send the voltages to be applied on the motors to the crane process. If the velocity control loops have fast time constants, the process can be considered as two double integrators from acceleration to the corresponding position, one in each direction. A SIMULINK block for communication with the hardware and velocity control of the cart was available for use in the thesis.

Experimental results

The results from the solution of the path tracking problem studied in this chapter can be used almost directly on the crane process by interpreting the joint positions as positions for the cart along the rails instead. However, the path must be scaled in order to fit the dimensions of the real process and the limits on the control signals have to be adapted to the real process.

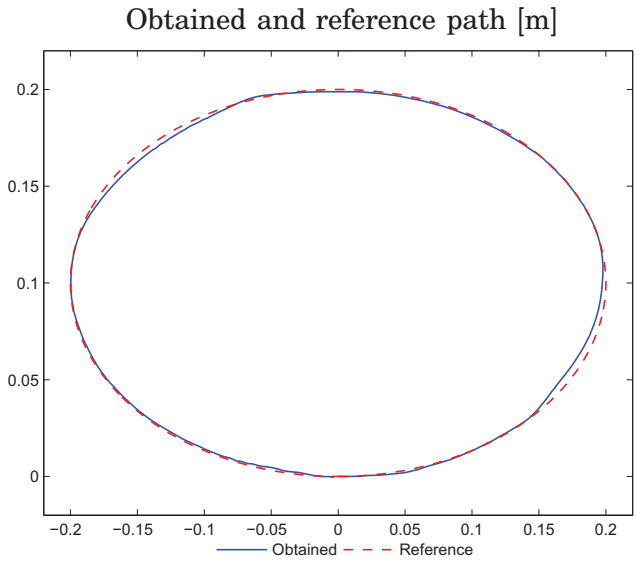

Figure 3.7 In the figure the result from the path tracking experiments on the crane process is shown. The position of the cart in the y-direction is plotted as function of the position in the x-direction.

An implementation of the PVC described in Section 2.8 made in SIMULINK was used for performing the experiments. For a more in-depth treatment of the PVC-implementation made in this thesis, the reader is referred to Section 5.2.

In the experiments on the crane process, the solution from the convex optimisation formulation of the path tracking problem was used. The limits on the acceleration signals in the optimisation were chosen to be 0.3 $\mathrm{m/s^2}$ due to the limited lengths of the rails. The result from this optimisation is only a scaled version of the results presented in this chapter, wherefore the new optimisation results are not presented.

Result The result of the path tracking on the gantry crane process is seen in Figure 3.7. In the figure, the position in the y-direction is plotted as a function of the position in the x-direction. The reference path, a scaled version of the path in Figure 3.1, is also shown for comparison. It is seen that the path tracking is working, but some parts are more difficult to track. During the experiments the limits on the accelerations have been chosen to be 0.42 $\mathrm{m/s^2}$ — *i.e.*, slightly larger than in the optimisation — in order to make the control more robust to modelling errors. This choice gives space to control the process despite modelling errors.

Improvements It is crucial for the path tracking that the model of the process is accurate. However, since the velocity control of the cart is implemented in a SIMULINK model and runs with the sampling period 0.01 s, the double integrator model might not be accurate enough. There are reasons to believe that the path tracking can be improved if the velocity control of the cart is implemented in hardware instead, *e.g.*, in a micro processor. This allows a higher sampling rate in the velocity control. Also controlling the current in the motors can be a way to improve the velocity control of the cart and consequently the path tracking.

4. Robot modelling and path identification

In this chapter robot modelling is discussed and a suitable robot model identified on the robot system is presented. Further, a contact-force control approach for path identification is described. Also the implementation of the force controller is discussed. Finally, experimental results from path identification with force control are presented.

4.1 Robot modelling

As earlier stated in Chapter 2, a good model of the robot is required for accurate path tracking. A natural way is to try to derive or experimentally identify a model on the real robot system from torques τ to joint positions q, in accordance with the rigid body model (2.6) presented above. However, such derivation or identification is not trivial to perform. Therefore, the decision was made that modelling of the separate joints as linear systems is appropriate for the needs in this thesis. Further, the couplings between the different joints in the robot were decided to be neglected.

Control structure for the independent joint control

In order to be able to identify suitable models of the robot joints, the structure in the control cabinet IRC5 chosen by the manufacturer for controlling the joints in the IRB140 robot has to be studied. Each joint in the robot is controlled by a cascaded structure of standard P and PI controllers. This is a standard way of controlling the robot joints [Siciliano *et al.*, 2009]. The cascade structure can be seen in Figure 4.1. The cascade structure makes it possible to control both the joint position and the joint velocity. The inner PI controller controls the joint velocity and the outer P controller controls the joint position. The advantage of using a cascaded structure instead of only an outer joint position loop is that the inner loop, the velocity loop, reacts fast on errors in the velocity control. Thereby, errors in the velocity control do not have to propagate to the joint position in order for the controller to react. As can be seen in Figure 4.1, it is possible to feedforward joint velocities $\dot{q}^r$. Also, torques τ^r to be applied on the joints directly can be feedforwarded to the control structure.

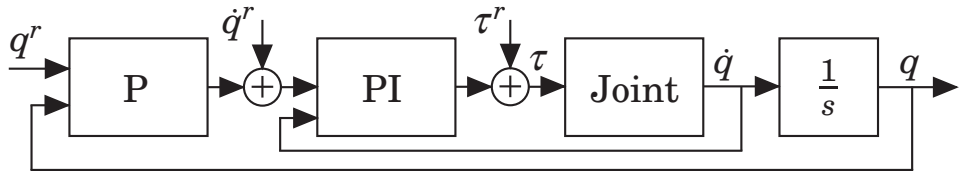

Figure 4.1 In the figure, the cascaded control structure used to control the individual joints in the robot IRB140 is illustrated. The control structure consists of one P controller for position control and one PI controller for velocity control. In the figure, superscript r denotes reference or feedforward signals.

Model suitable for optimisation and path tracking

With the cascaded control structure presented above at hand, different models can be appropriate for obtaining path tracking. The first choice when determining a model is what variable that is most appropriate as input signal. There are three options:

- torque τ
- position reference q^r
- velocity reference $\dot{q}^r$.

The first choice, the torque, can be utilised if the position and velocity control loops are turned off, see Appendix A. Experiments were made on the robot system with the torque as input signal. It was realised that nonlinear effects are apparent in the joints from input signal to the position and velocity. The nonlinear effects are mainly caused by friction in the joints. Experiments also showed that the friction is not constant, but rather dependent of the joint angle and the direction of the joint velocity. The influence of the friction makes it hard to use linear models of the robot joints. Further, choosing the torque as input signal leads to practical problems in terms of gravity compensation in order to avoid that the second and third link fall due to gravity. Accordingly, this choice was not made in this thesis.

The second choice, the position reference, may lead to problems because the input signal in a time-optimal path tracking often exhibits a so called bang-bang character where the signal switches from one extreme value to another in a short time. This is not appropriate for controlling the robot, since the safety system in the robot system does not allow the joint position reference to be too far away from the actual position. In case of too high discrepancy, as is the case when a step in the input signal occurs, the safety system locks the brakes on the robot. Further, it is not clear how the joint velocity reference should be selected when only controlling the joint position. To conclude, neither the position reference is appropriate to consider as input signal.

The third choice, to consider the velocity reference as input signal, has been tested in a previous Master thesis project [Hast, 2009]. This choice makes it possible to identify a model from the joint velocity reference to the joint velocity. Also, this choice of input signal does not suffer from the disadvantages that the other above choices have. In order to only use the joint velocity reference as input signal to the joint control system, the position loop has to be turned off, see Appendix A. To summarise, this is the most convenient input signal. Hence, this input signal was chosen in this thesis.

Linearity and model order The second question is if a linear model can be utilised. Certainly, the robot joints themselves are not linear from motor input to joint position as the experiments with torque control described above showed. However, by using the velocity reference as input to the system it is reasonable to believe that the system from input to the joint position can be modelled with a linear model. This is because there already is the PI controller for the joint velocity available in the robot system.

The third question is the order to choose for the models. Recalling the theory chapter, where an optimisation problem was presented with a robot model of order two, it is desirable to identify a model of second order from the joint velocity reference to the joint position for each joint. As the joint velocity is the derivative with respect to time of the joint position, a suitable transfer function model to identify for each joint i, $i = 1, \ldots, 6$, is on the form

$$\dot{q}_i = \frac{A_i}{B_i p + 1} \dot{q}_i^r \quad \text{or} \quad q_i = \frac{A_i}{B_i p + 1} \frac{1}{p} \dot{q}_i^r \tag{4.1}$$

where p is the differential operator, A_i is the gain and B_i is the time constant. These relations can be reformulated to the form

$$A^{-1} B \ddot{q} + A^{-1} \dot{q} = \dot{q}^r \tag{4.2}$$

where A is a diagonal matrix with the gains A_i and B is a corresponding diagonal matrix with the time constants B_i. From this relation it is apparent that a second order model from joint velocity reference to joint position has the same structure as the rigid body model (2.6). Hence, the methods for optimal path tracking presented in the theory chapter can be used without modifications by utilising this model.

Identification on the robot system Identification of models for the individual joints in accordance with the model (4.1) can be made with a step response experiment on each joint. These experiments are performed such that the velocity reference for the joint changes abruptly from zero to 0.2 rad/s. The selection of the step size was made to ensure that the joint operates in the linear range during the step.

During the step response experiments the velocity of the joint can be logged. The logged joint velocity values can then be plotted in a diagram. The parameters A_i and B_i in the first order models from joint velocity reference to joint velocity can be read from the diagram as follows: A_i is the ratio between the joint velocity in steady state and the chosen step size. Further, B_i is the time from when the step occurs until the fraction $1 - 1/e \approx 0.63$ of the final step value is achieved.

Experimental results

Step response experiments were performed on the robot system for each joint i, $i = 1, \ldots, 6$. During initial step response experiments on the robot system it was discovered that there is a rather long delay in the robot system from joint velocity reference to the response in the measured joint velocity. Measurements were made and the delay was estimated to approximately 4–5 samples, or 16–20 ms, see Figure 4.2.

Since the delay is long relative to the rise time, a strategy has to be introduced for reducing the influence of the delay. As a workaround for the delay, it was decided to introduce lowpass filters on the inputs, *i.e.*, on the joint velocity references. A first order Butterworth discrete filter was introduced on each input. The lowpass filters were then considered as part of the robot system. The step responses for the six joints with lowpass filters on the inputs can be seen in Figure 4.3.

From the step responses in Figure 4.3, the parameters A_i and B_i in the model (4.1) can be estimated for respective joint. Since the joint velocity loop in the robot system contains an integrator, no stationary errors are

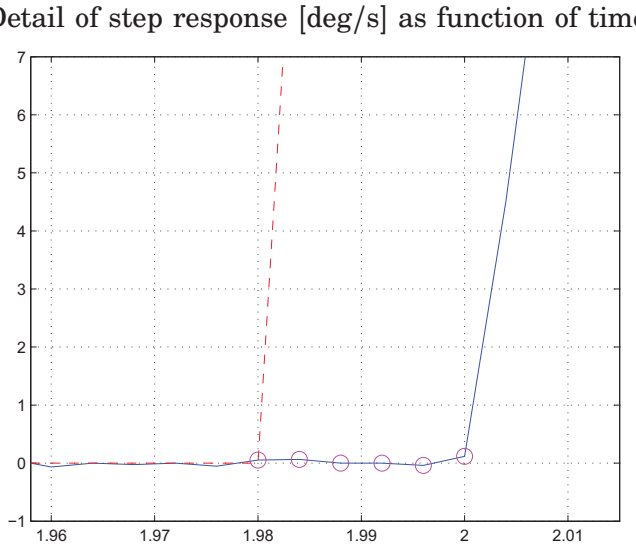

Figure 4.2 The figure shows a detail from a step response in the joint velocity reference for joint 2. A delay of 4-5 samples can be seen from joint velocity reference (dashed red) to the measured joint velocity (solid blue). The sample instants are marked with circles.

Table 4.1 The table displays the time constants B_i, $i = 1,\ldots,6$, in the models identified on the six joints of the IRB140 robot.

Joint	Time constant [s]
1	0.1466
2	0.1483
3	0.1513
4	0.1482
5	0.1509
6	0.1459

present in the output signal. Hence, all gains A_i are selected as 1. The time constants were measured in the step responses and the results are shown in Table 4.1.

4.2 Path to be tracked

When the path to be tracked is defined by a motion of a tool along a contour of an object, experimental methods are required in order to determine the corresponding motion of the robot as earlier mentioned. Both the position and the orientation of the tool have to be determined during the identification procedure. An interesting way of determining the geometric robot motion is by letting the robot identify the path along the object by itself. Then, interaction between the robot and its environment is vital. In this thesis, interaction is achieved by using a force sensor attached to the robot flange, see Appendix A. The force sensor measures forces and torques exerted on it in three different orthogonal directions. With the force sensor,

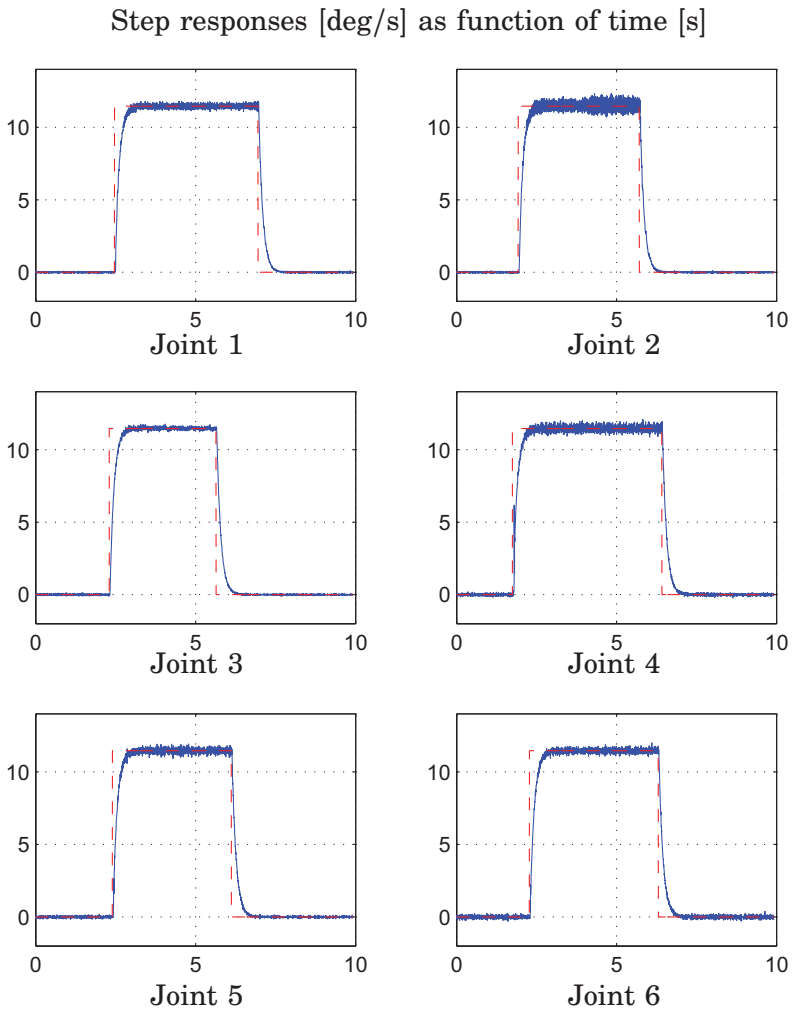

Figure 4.3 Step responses obtained for the six joints of the IRB140 robot, when applying a lowpass filter on the input to the robot system. The measured joint velocities (solid blue) and the reference values (dashed red) are shown. The lowpass filters reduce the problem with the delay in the robot system.

all contact-forces acting on the TCP can be measured, given that the sensor is calibrated. First, two different approaches to determine the geometric motion of the robot along the path are described below, where the latter is used in this thesis.

Lead through concept

One approach for determination of the robot motion is that a human teach the robot the path to be tracked by moving the TCP of the robot along the desired path. This approach, obviously, requires interaction between the robot and its environment. During the demonstration of the path the joint positions are recorded and later used in the determination of an optimal path tracking. This approach is developed in the lead through concept, where a human leads the robot along the desired path. Lead through is possible due to a force sensor measuring the force exerted on the robot TCP by the human. A controller moves the robot such that the contact-

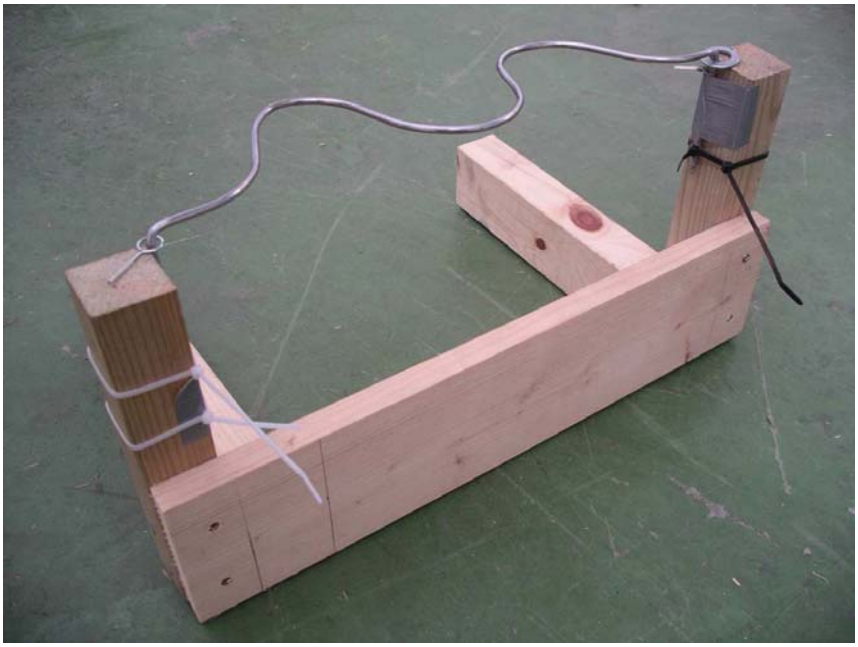

Figure 4.4 In the figure the path to be identified and later tracked is shown. The path consists of a metal bar with circular cross section.

force at the TCP is kept as close to zero as possible. Consequently, the robot follows the human demonstrator.

Force control identification

Another approach, chosen in this thesis, is to use contact-force control to automatically identify an unknown path to be tracked by using a technique called contour following, see *e.g.*, [De Schutter and Van Brussel, 1988a; De Schutter and Van Brussel, 1988b]. In the current case, the path that the tool is to track is a metal bar with a circular cross section with a diameter of 5 mm, see Figure 4.4. By attaching a small metal stick to the force sensor, the path can be identified with contact-force control. The metal stick and the path are shown in Figure 4.5. Using the fact that the force on the stick attached to the force sensor is perpendicular to the path once contact is achieved, the robot can be moved in the correct direction following the path by calculating the tangential direction of the path. However, also the friction forces arising between the path and the stick are measured by the force sensor. This issue will be revisited later. In order to keep the contact-force constant a force controller can be utilised.

4.3 Calibration of TCP and force sensor

Before force control experiments can be made on the real robot system, the force sensor and the tool — *i.e.*, the metal stick — have to be calibrated. For the sensor, this means that the axes along which the force and torque are measured are determined. Also, the distance from the force sensor origin to the robot flange has to be determined. For the tool, the calibration procedure means that the position and orientation of the TCP are determined relative to the robot flange.

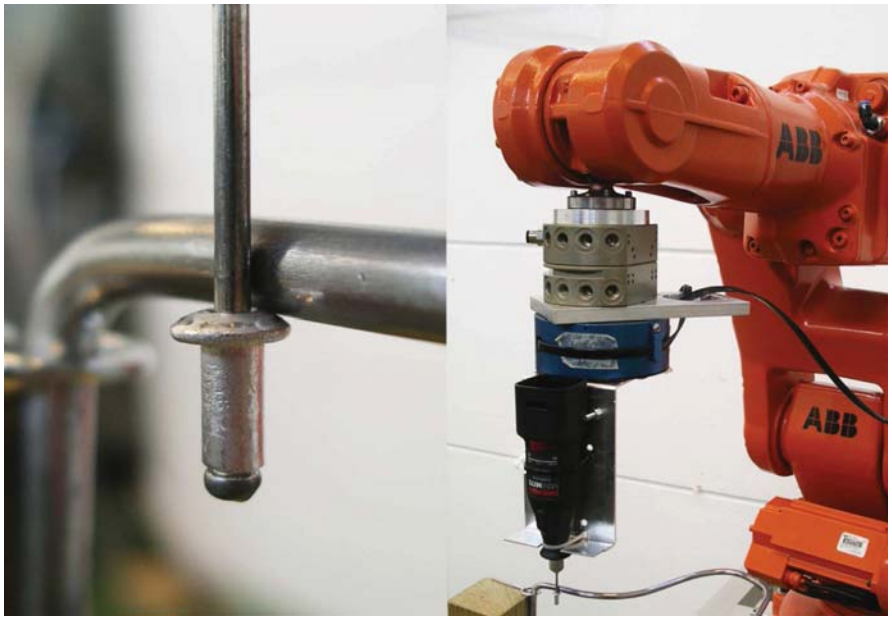

Figure 4.5 The metal stick used for path identification with contact-force control is shown to the left. Notice that the stick is in contact with the path both on the side and underneath it. To the right, the metal stick attached to the force sensor via a multitool is shown. The force sensor is the blue disk attached to the robot flange.

Transformation matrices

The result of the calibration is two transformation matrices, a T_{44}–matrix describing the transformation from the sensor coordinate system to the flange coordinate system and a corresponding matrix describing the transformation from the TCP coordinate system to the flange coordinate system.

Tool calibration The calibration of the TCP can be made with standard routines incorporated in the robot system from ABB. The procedure is called a four point calibration. This procedure determines the position of the TCP relative to the flange. More specificly, the robot is placed in four different configurations, but such that the TCP in the four configurations have the same location in the space $\mathbb{R}^3$. Then, an equation system is solved to determine the position of the TCP relative to the flange. Further, it was decided that the orientation of the TCP coordinate system can be the same as the orientation of the flange coordinate system. The result from the calibration is the transformation matrix

$$T_{44} = \begin{bmatrix} 1 & 0 & 0 & -68.1 \\ 0 & 1 & 0 & 4.7 \\ 0 & 0 & 1 & 275.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

which describes the transformation from the TCP coordinate system to the flange coordinate system. Note that the three first elements in the fourth column represent the translation of the TCP coordinate system relative to the flange coordinate system. The unit chosen in the transformation matrix is millimeter.

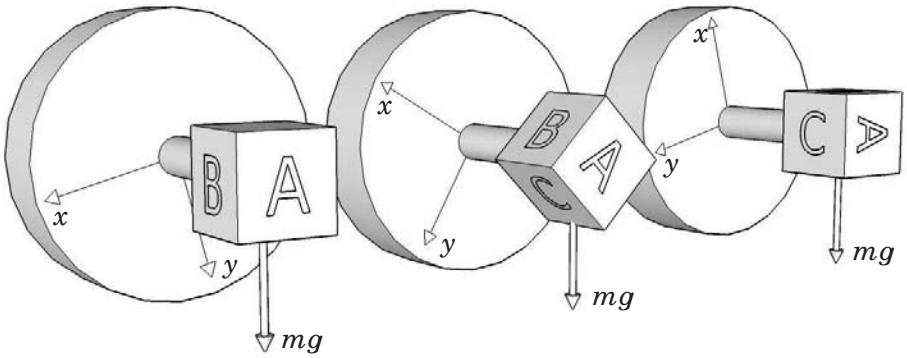

Figure 4.6 In the figure a schematic illustration of the sensor calibration procedure is shown. As joint six changes its position, the sensor and the tool will rotate. At the same time, the x- and y-components of the gravity force in the sensor coordinate system change as a sinusoid.

Sensor calibration Calibration of the force sensor can be made by positioning the robot in its home position, *i.e.*, the position where all joint positions equal to zero. Then, the gravity acting on the tool attached to the force sensor results in a force that is measured by the force sensor. It is known that the force sensor z-axis is oriented in the same direction as the flange coordinate system z-axis. Hence, the sensor coordinate system is only a rotated version of the flange coordinate system in the xy-plane.

In the home position of the robot, the gravity force on the tool only gives components along the sensor x- and y-axes. By rotating the sixth joint with a constant angular velocity the orientation of the force sensor x- and y-axes can be changed, see Figure 4.6. During the rotation of the sixth joint, the components of the gravity force in the sensor x- and y-axes change as sinusoids. The measured force components can then be fitted with a nonlinear least-squares approximation to a sinusoid. With the fitted sinusoid, the rotation angle between the flange coordinate system and the sensor coordinate system can be determined. This procedure is facilitated by a script in MATLAB available in the Robotics Lab that makes the least-squares approximation and determines the rotation angle.

According to the manufacturer of the force sensor, the origin of the sensor coordinate system is located at the centre of the sensor. Accordingly, the distance between the flange of the robot and the sensor origin can be measured with a vernier caliper. Measurements show that the sensor origin is located along the z-axis of the flange at a distance of 86.4 mm from the origin of the flange. Utilising this information and the rotation angle of the sensor coordinate system, the following transformation matrix can be established

$$T_{44} = \begin{bmatrix} 0.7456 & -0.6664 & 0 & 0 \\ 0.6664 & 0.7456 & 0 & 0 \\ 0 & 0 & 1 & 86.4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

which describes the transformation from the sensor coordinate system to the flange coordinate system. The unit chosen in the transformation matrix is millimeter.

Gravity compensation for the force sensor

When the robot is reoriented, the gravity force components resulting from the tool attached to the sensor and measured in the different axes of the sensor frame change since the force sensor coordinate system is fixed relative the flange coordinate system. Hence, in order to only measure the real contact-forces acting on the tool, the gravity force has to be compensated. The gravity force also gives raise to a measured torque that has to be compensated.

The compensation can be done by estimating the gravity force acting on the tool attached to the sensor and the position of the centre of gravity for the tool. This information is also extracted from the data that is used to determine the sensor frame with the method described above. The information is calculated with the same script in MATLAB that determines the sensor frame information.

With the centre of gravity and the gravity force vector known, an online gravity compensation for the force and the torque can be implemented in the SIMULINK controller. The compensation is recalculated in every sample to mirror the reorientation of the robot.

4.4 Control structure for path identification

In order to use force control for path identification, a suitable force control strategy has to be utilised. First, it is assumed that the path is located in a two dimensional plane with a given normal vector. Then the identification procedure can be performed by moving the TCP of the robot only in this plane in order to follow the contour of the path. Further, this assumption implies that the normal force on the metal stick in Figure 4.5 in every point along the path is located in the same plane. In the presentation that follows, it is assumed that the robot is oriented such that the z-axis of the sensor frame is parallel to the normal of the plane in which the path is located. The procedure of the identification will be described in the corresponding section below and is illustrated in Figures 4.7–4.8.

Force control

The main part of the force control structure is a PI controller working on the error of the normal force, commonly referred to as a direct force controller [Siciliano *et al.*, 2009]. The normal force can be calculated by projecting the measured force vector $f = [\, f_x \quad f_y \quad f_z \,]^T$ on the plane, in which the path is located. Since it is assumed that the path is located in a plane parallel to the xy-plane of the sensor frame, the normal force f_N can be established as follows

$$f_N = [\, f_x \quad f_y \quad 0 \,]^T . \tag{4.5}$$

The control signal $\tilde{f}_c$ is calculated by the control law

$$\tilde{f}_c = K_p(||f_N^r|| - ||f_N||) + \frac{1}{K_i}\int_0^T (||f_N^r|| - ||f_N||)\, \mathrm{d}t \tag{4.6}$$

where $||\cdot||$ denotes the 2-norm, f_N^r is the desired normal force and K_p and K_i are tuneable controller parameters. The direction of the control signal

is decided as the opposite direction of the normal force signal, which means that the control signal vector f_c can be written

$$f_c = -\tilde{f}_c \frac{f_N}{||f_N||}. \tag{4.7}$$

The control signal is interpreted as the velocity vector in the xy-plane of the TCP, *i.e.*, the TCP is moved in the direction and with the velocity given by the control signal.

Vertical position controller During initial experiments on the robot system, it was noticed that if the path is not oriented exactly in the assumed plane due to differences in the vertical position along the path, the contact point between the stick and the path varies during the path identification. Consequently, it is desirable to compensate for the slight differences in the vertical position along the path such that the contact point is fixed. This can be achieved by introducing a force controller in the z-direction of the force vector and usage of a two dimensional metal stick, see Figure 4.5 where the stick is in contact with the path both on the side and underneath it. The two dimensional stick allows measurements of both the normal force and the force in the z-direction.

The control law for the vertical position controller is exactly the same, except for controller parameters, as for the normal force controller, *i.e.*, a PI controller. A reference value has to be selected corresponding to the desired contact-force between the stick and the path. The control signal of the controller is interpreted as a velocity of the TCP in the z-direction.

Tangential direction To follow the unknown path a new tangential direction along the path has to be calculated in every sample. The tangential direction is calculated such that it is perpendicular both to the measured normal force vector f_N and the normal vector n of the plane in which the path is located. Mathematically, the tangential direction v_t is calculated as the vector product between these two vectors, *i.e.*, as

$$v_t = n \times f_N. \tag{4.8}$$

The calculated vector is normalised such that its length corresponds to a movement of the TCP along the path with 1 mm/s.

Reference value The contact-force reference value in the PI controller can be chosen arbitrarily. Though, it is advantageous to choose a value as low as possible. Firstly, the friction force arising in the contact point between the stick and the path is reduced if the force reference value is reduced. Further, the impact on the environment that the stick causes is reduced if the force reference is as low as possible. This is important in applications where the path to be investigated is made of a weak or flexible material.

Torque control

During traverse of the path it is advantageous to reorient the tool such that the contact point between the tool and the path does not change. For this purpose the torque measurements from the force sensor can be

utilised. During the path identification, the force sensor is oriented such that the force sensor z-axis is perpendicular to the plane in which the path is located. Consequently, the normal force acting on the tool results in a torque in the force sensor z-axis direction.

Reorientation In order to reorient the tool correctly during the identification of the path, the torque in the z-direction can be controlled in order to be held constant. Then, the contact point will not change. The control structure chosen is a PI controller that acts on the difference between the torque in the z-direction M_z and a corresponding reference value M_z^r. The control signal M_c can be written as

$$M_c = K_p^t(M_z^r - M_z) + \frac{1}{K_i^t}\int_0^T (M_z^r - M_z)\ \mathrm{d}t \tag{4.9}$$

where K_p^t and K_i^t are tuneable controller parameters.

The control signal is then interpreted as an angular velocity around the z-axis of the TCP coordinate system in the opposite direction of M_c, *i.e.*, as an angular velocity ω expressed in the TCP coordinate system according to

$$\omega = [\,0 \quad 0 \quad -M_c\,]^T\,. \tag{4.10}$$

This is important since the reorientation has to be done around the point of contact between the stick and the path, otherwise contact with the path will be lost. This angular velocity is then transformed into a corresponding velocity of the flange of the robot. Note that an angular velocity around the z-axis of the TCP corresponds both to linear and angular velocities of the flange.

Reference value The reference value for the PI controller can be chosen arbitrarily as long as the torque is realisable with respect to the chosen force control reference value and the corresponding arm length for the force vector. An especially easy choice of torque reference is zero. This corresponds to a configuration where the normal force vector intersects the sensor z-axis all the time and consequently the torque around the sensor z-axis is equal to zero. During experiments on the robot system it was found that it is advantageous to choose the reference value slightly larger than zero in order to compensate for the small friction forces arising when the metal stick is moved along the path. The reason for this is that the friction forces also give contributions to the torque around the sensor z-axis.

Procedure

The procedure of identifying the path with contact-force control can be described as follows. With reference to Figure 4.7, at (A), the stick is positioned in front of the path and starts to move in a prespecified direction towards the path. At (B), the stick will come into physical contact with the path. When the normal force magnitude $||f_N||$ exceeds a certain threshold value the controller for the normal force is switched on. This controller will now make sure that $||f_N||$ reaches the desired force $||f_N^r||$. Then the stick will begin to move in a direction v_z, which is parallel to the normal of the plane in which the path is located. At (C), when the stick comes into contact underneath the path the vertical position controller is activated

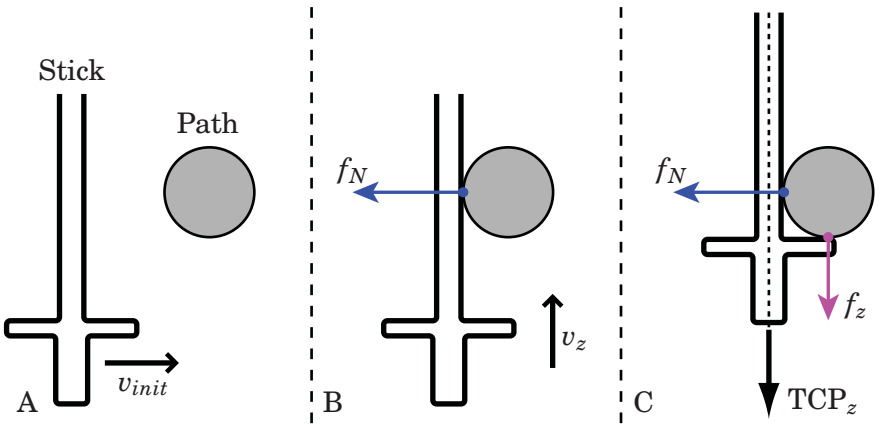

Figure 4.7 The figure illustrates the first steps in the procedure of the path identification. In the figure, the stick and the cross section of the path can be seen. (A) is the same instance here as in Figure 4.8. In (C) the z-axis of the TCP coordinate system is also illustrated.

and will repeat the same procedure but this time for the force f_z. When the desired forces are reached in both directions the torque controller is activated.

With reference to Figure 4.8, at (B) the force sensor will measure a torque M_z. The torque controller will make sure the torque reaches the desired value by rotating the tool around the point of contact, and hence change the distance r. At (C), once the desired torque is reached the stick will begin to move along the path in the tangential direction v_t, which is calculated according to (4.8). At (D), when the stick reaches a curve in the path, the tangential direction will follow the arc, all according to (4.8) since the force f_N will change its direction. During the whole traverse the force controllers are active in order to ensure that the contact-forces are kept as close to the reference values as possible.

Implementation of force controller in Simulink

In order to test the force control structure before performing experiments on the real robot system, a simulation model was implemented in SIMULINK. This model allows pre-tuning of controller parameters before performing experiments on the real robot system. During simulations, the robot system and the forces and torques arising when the stick comes into contact with the path are simulated.

In order to structure the simulation model, subsystems have been used as far as possible. The main simulation model in SIMULINK can be seen in Figure C.1 in the appendix. Below some of the main points of the implementation made in this thesis are discussed.

Library extctrl In the Robotics Lab a library called `extctrl` containing SIMULINK blocks suitable for robot control has been developed. This library was available for use in the thesis. Several of these blocks were utilised during the implementation of the force controller. Among the most important blocks, the block calculating the transformation matrix from the robot flange to the base coordinate system and the block calculating the Jacobian for the robot flange in the base coordinate system can be mentioned. Note

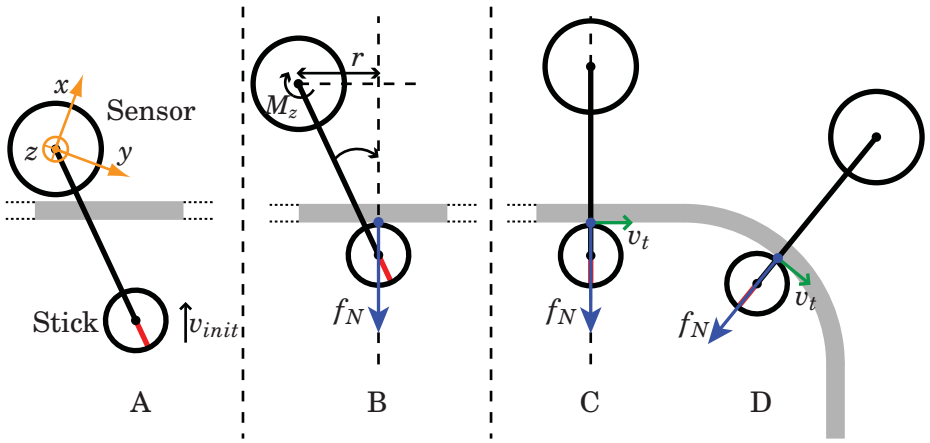

Figure 4.8 The figure illustrates the reorientation of the tool during the path identification, seen from above. Note how the rotation of the tool (B) aligns the stick such that its orientation is the same with respect to the path during the traverse, see (C) and (D) respectively. (A) is the same instance here as in Figure 4.7.

that both the transformation matrix and the Jacobian have to be recalculated in every sample, because they depend on the current configuration of the robot.

Also, blocks for resetting the force sensor, multiplying two transformation matrices and blocks for transforming velocities and forces from one coordinate system to another are available in the library. Further, two blocks are available for transforming the joint angles to motor angles and vice versa. These blocks take the gear ratios of the joint motors and the cross couplings of joint 4, 5 and 6 into account.

Simulation of forces and torques In order to simulate forces and torques arising when the stick comes into contact with the path, a suitable model describing the contact-forces has to be introduced. One basic model that has been utilised is to use a linear deformation law, often referred to as Hooke's law, where the force f_N is modelled as

$$f_N = kx \tag{4.11}$$

where k is the so called spring constant and x is the linear deformation of the path. Obviously, a large spring constant k corresponds to a stiff material that is hard to deform and a small spring constant to a soft material. The direction of the force is perpendicular to the tangent of the path and can thus easily be simulated.

With the force and its direction known, the torque in the sensor frame can be calculated since the vector r from the sensor origin to the contact point is known. Hence, the torque M can be calculated with the vector product

$$M = r \times f_N. \tag{4.12}$$

In order to make the simulations as close to the real experiments as possible, the simulated force and torque are transformed to the sensor frame before transmitting the signal to the force controller. The transformation is made with the transformation matrices described in a previous section. Simulating the forces and torques in the sensor frame also allows easy

change between simulations and real experiments, since the block simulating the forces can be replaced by the signal from the real force sensor.

Simulation of robot system The simulation model implemented in SIMULINK can also be used for executions on the real robot system after code generation, see Appendix A. During simulations, the robot system has to be modelled. Since no explicit feedback is used from the joint positions and joint velocities, the robot model utilised is not that important. In this application it is assumed that the reference values for the joint velocities and joint positions given to the robot system are tracked virtually instantaneously. Hence, the model identified on the robot system and used for optimisation purposes has been used during the simulations of the force control application. A discretised version of the robot model is utilised. Consequently, a fixed step discrete solver can be used to perform the simulation. The robot model block is replaced by a block communicating with the robot system when performing experiments on the real robot system.

Force controller The force controller is implemented according to the description in a previous section. The I-parts in the different PI controllers are discretised using forward Euler. Prior to using the force sensor signals in the force controller, the signals are filtered with a Butterworth lowpass filter. The reason for this approach is that the signals from the force sensor have a notable noise component.

The control signals from the different parts of the force controller are interpreted as velocities of the TCP. The velocities from the different parts of the force controller are added with equal weight. The block calculating the Jacobian from the `extctrl` library gives the Jacobian connecting the velocities of the flange in the base coordinate system and the joint velocities. Hence, the velocity of the TCP has to be transformed into a corresponding velocity of the flange expressed in the base coordinate system. Since the TCP and the robot flange forms a rigid body, the linear velocities can be transformed to the base coordinate system by rotating the velocity vector with the corresponding transformation matrix from the TCP coordinate system to the base coordinate system. Note that the angular velocities of the TCP have to be transformed with the velocity transmission block from the `extctrl` library.

Jacobian The velocities calculated in the force controller — *i.e.*, three linear velocities and three angular velocities — of the flange have to be transformed into corresponding joint velocities of the six joints of the robot. For this purpose, the Jacobian block from the `extctrl` block can be utilised. The Jacobian describes the transformation from joint velocities into the corresponding velocity of the flange. By inverting the Jacobian, a given velocity of the flange can be transformed into joint velocities. These joint velocities can then be applied on the robot system. Since only joint velocities are utilised and not the joint positions, the position loops in the robot system were decided to be turned off, see Appendix A.

Safety features In order to allow a safe start procedure of the force controller, a switch called `f_switch` was introduced in the model. This variable was also appointed as an inline parameter, which makes it possible to alter

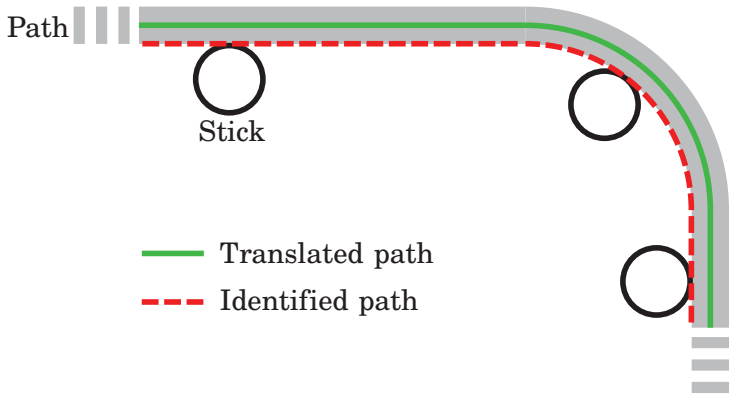

Figure 4.9 In the figure a schematic picture of the identified path and the corresponding translated path is shown.

the value of the parameter from the robot system Opcom, see Appendix A. When `f_switch` is zero, the control signals sent to the robot system are set to zero. On the contrary, when the switch is set to one, the path identification procedure is started.

In order to stop the control of the robot in a case where the contact with the path is lost during the path identification, a safety feature was implemented that sets the control signals to the robot system to zero if contact with the path is lost. The reason for this feature is that it is plausible that an error has occurred if the contact with the path is lost.

Translate the path When the path is identified, the stick is moved along the outer edge of the path. In some cases it is more interesting to know the centre of the path. This can be achieved by translating the identified path the corresponding offset from the contact point to the centre of the path. This distance is oriented in the direction of the normal force. Hence, the thickness of the path can be measured and the path translated in the direction of the measured normal force a distance of half the thickness of the path, see Figure 4.9.

In order to perform time-optimal path tracking, the joint positions along the translated path also have to be determined. This is done by the procedure described in Section 2.2 called inverse kinematics. Given a position and orientation of the robot flange, a block in the `extctrl` library calculates the corresponding joint positions. Since the inverse kinematics problem has multiple solutions, a strategy for selecting the most appropriate solution has to be introduced. In the inverse kinematics block, the solution closest to the previous solution is chosen. Hence, the previous solution also has to be fed to the inverse kinematics block.

The translation procedure is made automatically during the path identification. The corresponding joint positions are logged during the entire path identification and can accordingly be used in the path tracking optimisation. Note that the correct offset for the current path has to be set in the SIMULINK model for correct path identification.

Table 4.2 The table shows the controller parameters used during the identification of the path with the force controller.

	K_p	K_i
Normal force controller	0.55	10
Vertical position controller	0.25	20
Torque controller	0.95	7

4.5 Experimental results from path identification

A force control identification according to the strategy presented above was made on the real robot system with the path seen in Figure 4.4. Thereby, the SIMULINK model previously described was used. Before performing experiments on the real robot system, controller parameters were pre-tuned in the simulation model. Then, during robot experiments the controller parameters were adapted to the new real environment. The controller parameters used in the robot experiment can be seen in Table 4.2. Note the relative small values of the gains K_p, motivated by the desire not to amplify the remaining noise component in the measured and filtered force and torque signals from the force sensor.

Both the identified path along the outer edge of the metal bar and the translated path can be seen in Figure 4.10. The identified path has been translated a distance of 1 mm in the opposite direction of the normal force. The reason for not translating half the thickness of the path is that the tool used for the path identification is quite flexible and accordingly contributes to the translation.

In Figure 4.10, the measured force vector in some points along the path is also seen. By comparing the direction of the measured force to the tangent of the path, it is apparent that the friction forces make the measured force not completely corresponding to the assumed normal force. However, the deflection was not that significant because of the low reference values chosen in the force controllers. Therefore, the friction forces were decided to be neglected during the path traverse.

Further, the joint positions q and the corresponding position of the TCP in the base coordinate system can be seen in Figure 4.11 and Figure 4.12, respectively. These joint- and TCP-positions correspond to the translated path. The joint positions seen in the figure are later used in the optimisation for determining of the optimal path tracking.

Finally, the controlled forces and torque during the path identification can be seen in Figure 4.13. The reference value for the normal force controller is 5 N and the reference value for the vertical position controller is 4 N. A value of 0.075 Nm is chosen as the reference value for the torque controller. This value has been tuned in order to compensate for the friction forces arising between the metal stick and the path, which also contribute to the measured torque.

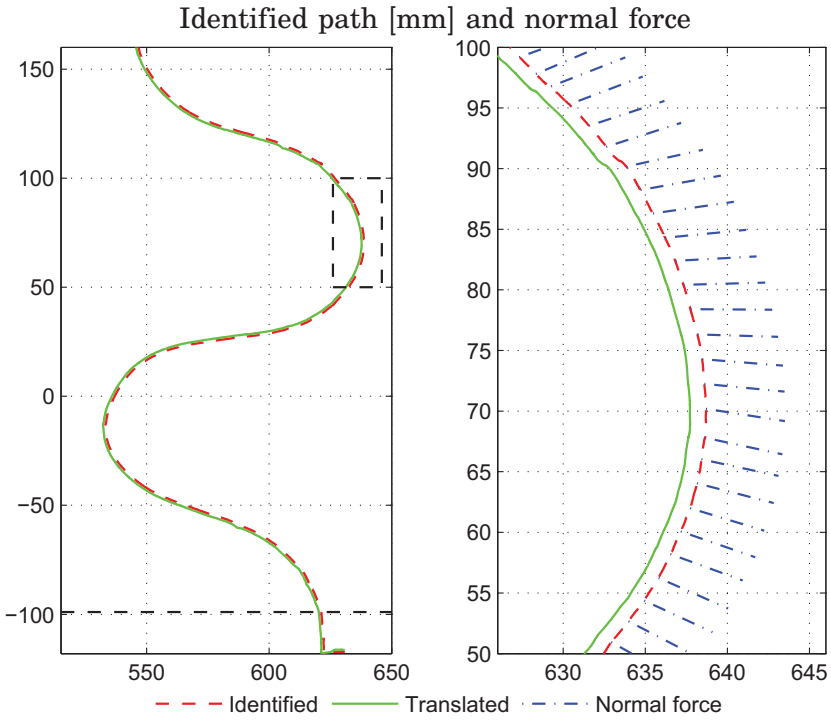

Figure 4.10 In the plot to the left, the identified path can be seen in its whole. The TCP y-position is plotted as function of the TCP x-position. Both positions are given in the base coordinate system. In the plot, also the translated path is shown. The horizontal dashed line marks the start of the path used in the optimisation. In the plot to the right, a detail of the path can be seen. The corresponding normal force vector in some of the points along the path is also shown. Note that the friction forces make the measured force not completely perpendicular to the path.

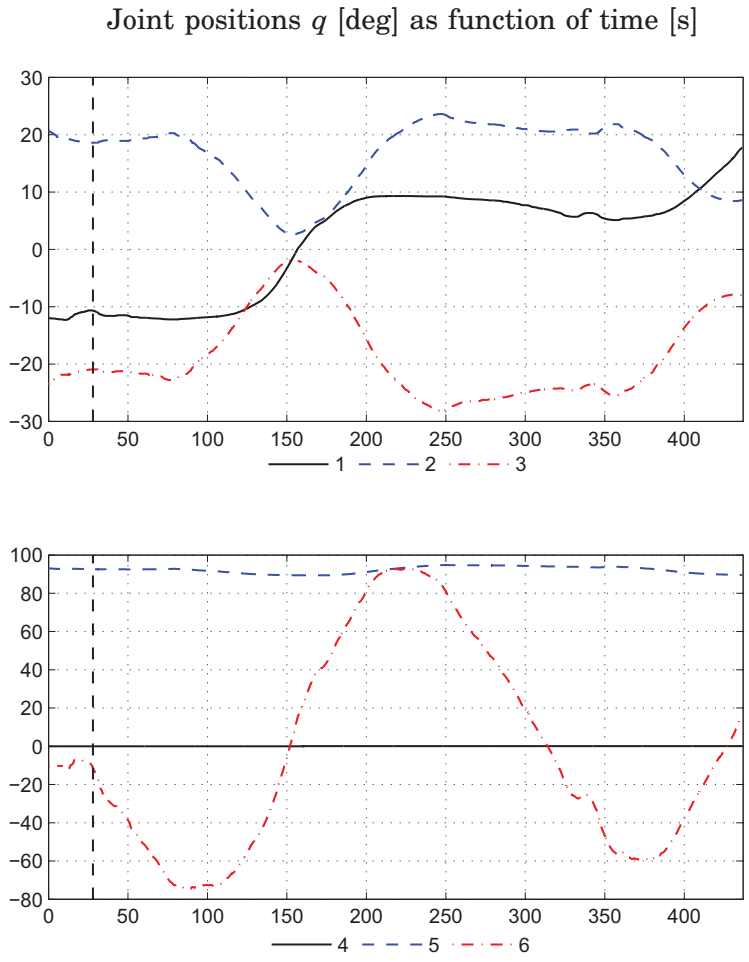

Figure 4.11 The figure shows the identified joint positions q_i, $i = 1,\ldots,6$, of the robot along the path. The dashed line marks the start of the path used in the optimisation.

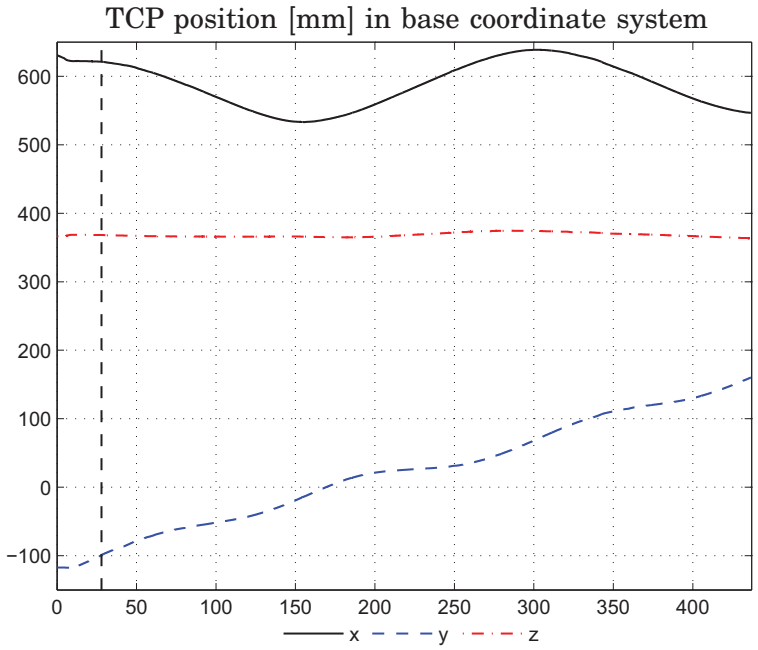

Figure 4.12 The figure shows the position of the TCP expressed in the x-, y- and z-directions of the base coordinate system. Note the slight differences in the vertical position in the z-axis direction along the path. The dashed line marks the start of the path used in the optimisation.

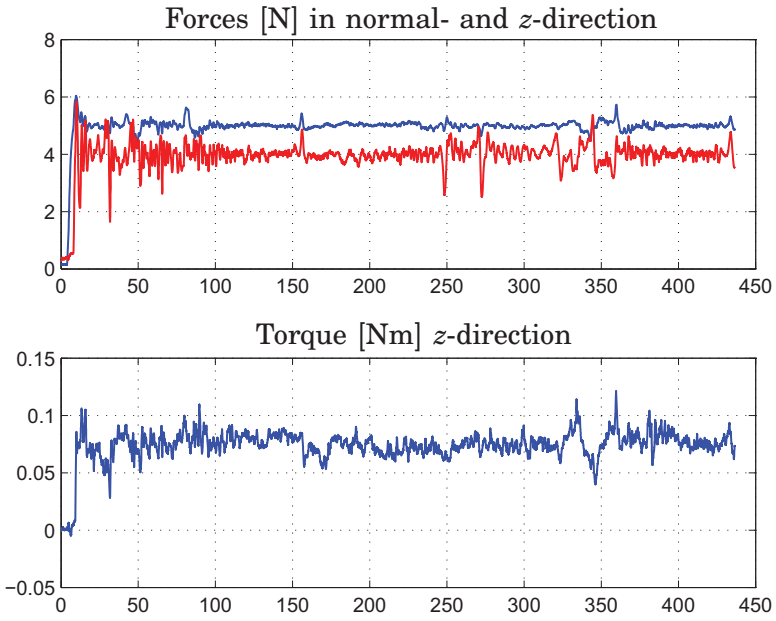

Figure 4.13 In the upper plot, the normal force (blue) and the force in the z-axis direction (red) during the path identification are shown. The reference value for the normal force is 5 N and the reference value for the force in the z-direction is 4 N. In the lower plot the torque in the z-direction is shown. The reference value is 0.075 Nm.

5. Path optimisation and experimental results

This chapter presents results obtained from optimal path tracking. The results presented include optimisation results, simulations and experimental results obtained from executions on a real robot system. The model used to execute the simulations and experiments is also presented and discussed.

5.1 Optimisation problem in JModelica.org

Once the force identification of the path has been performed and the joint positions of the robot along the path have been determined, the optimisation problem determining the optimal path tracking can be solved in JModelica.org. The optimisation problem is defined by (2.19)–(2.22). Note that the convex optimisation formulation of the path tracking problem cannot be used since a viscous friction term is present in the identified robot model (4.2). This term makes that the rewritten robot dynamics cannot be expressed affine in the path acceleration and the path velocity squared. Also note that the input signals for the robot system are not the joint torques τ in this case. In the current model, the joint velocity reference $\dot{q}_i^r$ for each joint i is used as input signal. However, the optimisation problem as such is exactly the same.

Rewritten robot dynamics

The robot dynamics subjected to path tracking requirements for the identified robot model can be decided analogous to EXAMPLE 2.1. The rewritten robot dynamics for the model (4.1) can be written, for each joint i, $i = 1, \ldots, 6$, according to

$$A_i \dot{q}_i^r = B_i(f_i'(s)\ddot{s} + f_i''(s)\dot{s}^2) + f_i'(s)\dot{s}. \tag{5.1}$$

Implementation

With the rewritten robot dynamics, it is straightforward to implement the optimisation problem in JModelica.org. The implementation is analogue to the implementation made in the example in Chapter 3. The paths are represented with cubic splines. Transformation of the joint positions obtained during the path identification to a Modelica model describing the splines is facilitated by a script written in MATLAB called `robot2jmod`. When creating the splines it is neither possible nor necessary to use all the data collected during the path identification. Since the path identification with force control is performed with a rather low speed, a lot of data values are collected. When making the splines, a uniform decimation is selected such that the experimentally determined paths are approximated good enough by the splines. The number of data used for making the splines is also limited by the capacity of the compiler in JModelica.org, as the spline implementation requires several lines of code.

It has been found advantageous to solve the optimisation problem in two steps when introducing derivative weighting in the cost function with

the parameter η. First, the pure time-optimal solution is determined as a reference solution. Then, derivative weighting is introduced in a second step where the result from the first optimisation is used as initial values.

Issues when solving the optimisation problem

Two main issues were encountered when solving the optimisation problem in JModelica.org. Firstly, when derivative weighting is used in the cost function, the absolute value is not appropriate in the numerical optimisation in JModelica.org. The reason is that the absolute value function cannot be continuously differentiated, which is required in the numerical solution. A workaround for this problem is to use the square function instead of the absolute value, *i.e.*, change the cost function in the optimisation problem to

$$\int_{s_0}^{s_f} \frac{1}{\sqrt{\beta(s)}} + \eta \sum_{i=1}^{n} (\dot{q}_i^r(s)')^2 \, \mathrm{d}s. \tag{5.2}$$

The square function also has the property that the function value always is non-negative. However, the square function amplifies high values of $\dot{q}_i^r(s)'$ and reduces low values. This has to be taken into account when tuning the weighting parameter η in the cost function, since the square function is more aggressive than the absolute value.

Differentiation of the paths The second issue concerns the individual paths for the six joints. In the solution of the optimisation problem, the first and second order derivatives of the paths are required. One option that was tested was to let JModelica.org differentiate the paths during the solution. This option led to problems when simulating initial values for the optimisation in SUNDIALS. The simulation tool Dymola can also be used for simulating the initial values for the Modelica models in the optimisation. Dymola was tested with satisfactory result and the results can be imported to JModelica.org. Since it is desirable to use the simulation facility in JModelica.org for a straight workflow, it was decided to reformulate the implementation of the optimisation problem slightly.

The reason for the problems in the simulation in SUNDIALS is that the DAE system in the optimisation problem when JModelica.org differentiates the paths is of an index higher than one. The index of a DAE system is defined as the number of times parts of it have to be differentiated in order to get a system, from which the derivatives of the variables can be solved for [Mattsson and Söderlind, 1992].

If the index is higher than one, a procedure called index reduction can be introduced. One approach for index reduction presented in [Mattsson and Söderlind, 1992] is to differentiate some of the equations in the DAE system and introducing a number of dummy variables. The differentiated equations are then incorporated in the DAE system.

The index reduction can be executed automatically by simulation software, as is the case with Dymola. It is advantageous to reduce the index, because high-index DAE systems are hard to solve, see for example [Mattsson and Söderlind, 1992]. Since index reduction is not yet implemented in JModelica.org, a manual effort for reducing the index was made. The reason for the high index in the optimisation problem is the differentiation of the paths. Accordingly, the paths were differentiated manually and included in the optimisation problem. This method solved the simulation

Table 5.1 The table shows the chosen upper and lower limits on the input signals — *i.e.*, the joint velocity references $\dot{q}^r$ — to the robot system. The limits are given in rad/s and deg/s.

Joint	Limit [rad/s]	Limit [deg/s]
1	± 0.6981	± 40.0
2	± 0.6981	± 40.0
3	± 0.9076	± 52.0
4	± 1.2566	± 72.0
5	± 1.2566	± 72.0
6	± 1.5708	± 90.0

problems in SUNDIALS. The drawback with the method is that more code is required in the Modelica models, and hence the compilation time when solving the optimisation problem is increased.

Implementation in JModelica.org

The Modelica and Optimica code describing the optimisation problem can be seen in the listing in Appendix B.2. A separate Modelica model which is called `path_tracking_Init_Opt` is used for simulation in SUNDIALS of initial values for the optimisation. The limits on the input signals have been chosen as 20 % of the maximum realisable joint velocity according to the manufacturer of the robot. The limits on each joint can be seen in Table 5.1. Further, the weighting parameter η in the cost function has been chosen as $1.0 \cdot 10^{-4}$. The paths are parametrised in the path parameter s, where $s \in [0, 1]$.

When solving the optimisation problem in JModelica.org a suitable number of elements has to be chosen for the transcription process. In the current problem 235 elements were chosen and in every element 3 collocation points were selected. The choice of number of elements is a trade-off between the accuracy in the solution, solution time and convergence properties.

When implementing the splines with if- and else-clauses, a function called `noEvent()` has to be used. This function tells JModelica.org that the included code can be calculated immediately, and no switches are required when moving from one region of the spline to another.

Optimisation results

The result from the optimisation can be seen in Figure 5.1 and Figure 5.2. The first figure displays the path velocity $\dot{s}(s)$ and the path acceleration $\ddot{s}(s)$. The second figure displays the input signals, *i.e.*, the joint velocity references along the path. During the optimisation, the time-optimal traverse time is also calculated. The theoretically time-optimal solution has a traverse time of 6.30 s for the current path. When derivative weighting is used with $\eta = 1.0 \cdot 10^{-4}$, the traverse time is increased to 6.49 s, *i.e.*, an increase of approximately 3 % compared to the time-optimal solution.

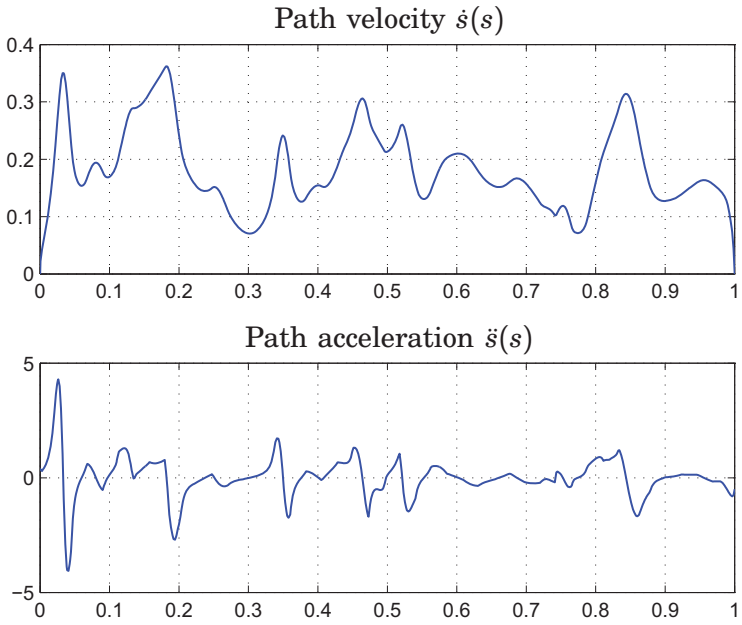

Figure 5.1 In the figure, the results from the optimisation in JModelica.org are shown. The upper plot displays the path velocity $\dot{s}(s)$ and the lower plot displays the path acceleration $\ddot{s}(s)$.

5.2 Implementation of PVC in Simulink

In order to test the different parts of the control structure obtaining the path tracking, simulations were made prior testing on the real robot system. The following section describes the simulation model used to make the simulations.

Simulation model for path tracking

In a previous Master thesis project at the Department of Automatic Control at Lund University, the PVC structure described in Section 2.8 was implemented in SIMULINK [Hast, 2009]. Though, the structure was not implemented on the real robot system with satisfactory result. The new implementation made in this thesis mainly follows the outline of the earlier made implementation.

In order to test different configurations of the PVC structure described in Chapter 2 with different robot models, a flexible implementation of a model for simulations of optimal path tracking has been constructed in SIMULINK. This makes it possible to test different robot models and different parameter choices in the PVC structure. Further, the current path and the results from the optimisation — *i.e.*, the path velocity and the path acceleration — are stored in a `.mat`-file, which is the format of MATLAB for saving data on disk. This data structure makes it possible to change between different paths and optimisation results from different optimisation methods in a straightforward manner. Below the main points of the implementation made in SIMULINK will be discussed.

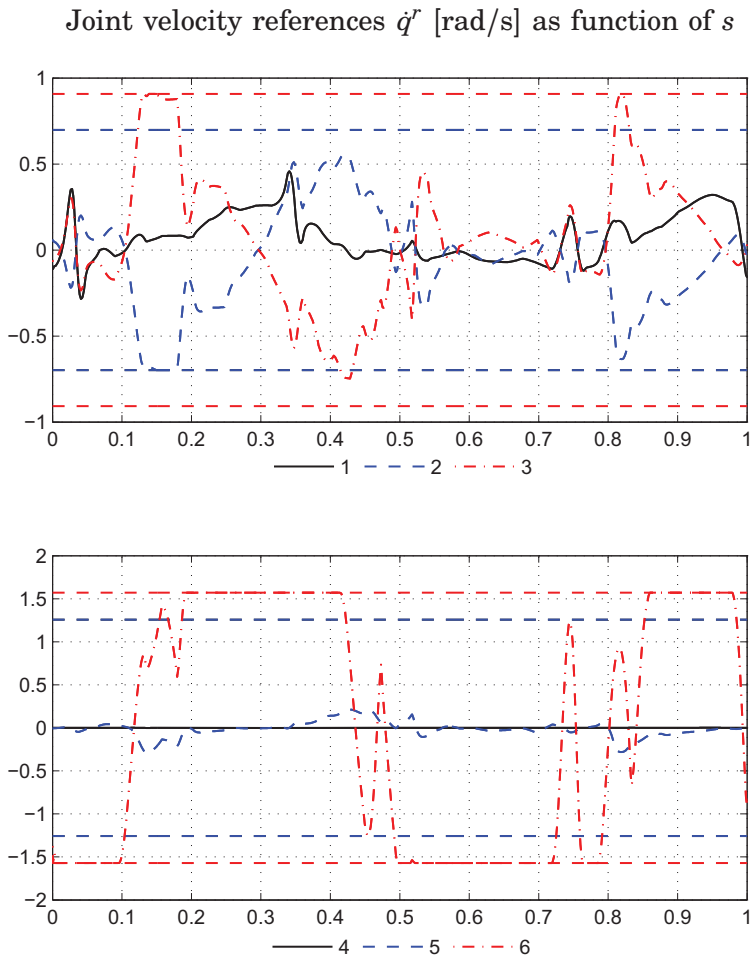

Figure 5.2 In the figure, the result from the optimisation in JModelica.org is shown. The plot displays the input signals to the robot system, *i.e.*, the six joint velocity reference signals $\dot{q}_i^r$, $i = 1, \ldots, 6$. Note that one of the joints is saturated at its limits a large part of the path. This indicates near time-optimality. The limits on the input signals are given in Table 5.1.

Implementation A simulation model was implemented in SIMULINK for the PVC according to Section 2.8. The model consists of several subsystems in order to structure the model. The main model can be seen in Figure C.2 in the appendix. When more specialised functions not available in the standard library in SIMULINK are required, the embedded code blocks have been used.

Tables In order to make the results from the optimisation, the paths and their derivatives available in the model, look-up tables have been used. These tables each use a variable saved in the MATLAB workspace. The variable must have two columns, one with the independent variable, here the path parameter s, and one with the dependent variable. During execution of the model, these tables perform linear interpolation between the data points in the variable in order to get data values also for intermediate points.

To facilitate the procedure of transforming an optimisation result from

JModelica.org to corresponding MATLAB variables suitable for the look-up tables, a script in MATLAB called `jmod2mat` was developed. It is of interest to be able to use an arbitrary number of points in the PVC, independent of the number of collocation points in the optimisation. Therefore spline approximations are made of the results from the optimisation. With this approach, an arbitrary number of points can be used in the PVC. In the current implementation 2000 equally spaced points were chosen to be used in the PVC.

Robot model The simulation model implemented in SIMULINK can also be used for executions on the real robot system after code generation, see Appendix A. In order to allow easy change between simulations and real executions on the robot system, two blocks were built. One implements the robot model identified on the robot system and the other communicates with the real robot. The first block is used in the simulations. When the PVC is to be executed on the real robot system, the identified robot model block is replaced by the real robot block communicating with the robot system. This allows easy change between simulations and real executions on the robot system. Also the inputs and outputs from the robot system described in Appendix A are incorporated in the model for communication with the robot system.

Starting switch In order to allow a safe start procedure when running the controller on the real robot system, a starting switch called `f_switch` was introduced in the model. This switch is also defined as an inline parameter and can accordingly be changed from the robot system Opcom, see Appendix A. When the switch is zero, the control signals are set to zero and the integrators in the controller receive zero as input value in order not to build up large values before starting. When the robot is ready to perform the path tracking, the switch is set to one and the controller starts and hence the path tracking starts.

Angle deviation As an extra safety feature in the PVC, the control of the robot system is stopped if at least one of the current joint positions of the robot deviates more than a certain predefined limit value from the specified path. The reason for this feature is that if the deviation of the joint position is larger than the limit value, it is reasonable to believe that there is an error that has occurred during the path tracking.

Simulation configurations In order to perform the simulations in SIMULINK, a suitable solver has to be chosen. Since the code generation is made with a fixed step discrete solver, the same solver was chosen for the simulations. In this way, the simulations are as close as possible to the real experiments on the robot system. The fixed step size is determined by the sampling period in the robot system, *i.e.*, a sampling period $h = 0.004$ s. During the simulations, the continuous time robot model has to be discretised. This is straightforward to do using a zero-order-hold method with the sampling period h.

5.3 Simulation results of optimal path tracking

The simulation model for the PVC has been used for tuning of controller parameters but also for obtaining a suitable trade-off between time-optimality and rate of changes in the input signals prior testing on the real robot system. This section discusses some observations made when simulating the PVC with various optimisation results.

Similarity of simulation with real experiments

Comparison of the results from the simulations and the real experiments on the robot system shows that the obtained results are similar. This is plausible since a fixed step solver with a step size equal to the sampling time in the robot system is used in the simulations. However, as expected the tracking errors are more evident in the real executions due to the limited validity of the identified robot model. This is particularly the case when the input signals — *i.e.*, the joint velocity references — exhibit fast changes since the actuators in the robot joints cannot realise the desired rate of changes in the velocities.

Issues when simulating the PVC

An issue that arose when simulating the PVC in SIMULINK was that the online calculated bounds on the path acceleration $\ddot{\sigma}$ in the PVC are inverted such that $\ddot{\sigma}_{max} < \ddot{\sigma}_{min}$. This means that there is no path acceleration that can be chosen such that the constraints on the input signals are satisfied. This problem is already recognised in both [Dahl, 1992] and [Hast, 2009]. The first suggests a solution also used in this thesis, where the nominal path acceleration $\ddot{s}(\sigma)$ from the optimisation is used without saturation temporarily as long as $\ddot{\sigma}_{max} < \ddot{\sigma}_{min}$. However, this solution is not optimal since if the limits are inverted more than a few samples, the path tracking can be severely violated because the desired input signals are not achievable. This might result in that the actuators cannot keep the robot following the desired path.

Numerical issues From the expressions for calculating the bounds on the path acceleration $\ddot{\sigma}(\sigma)$ in (2.35) and (2.36) online in the PVC, it is apparent that the limit calculations are very sensitive when β_1 is close to zero because this results in small numerators in the expressions. This might result in jumps in the limits and consequently in the saturated path acceleration $\ddot{\sigma}(\sigma)$, which of course negatively influence the path traverse.

Further, the constant sample time $h = 4$ ms in the robot system sets a limit of what is possible to achieve with the PVC. Since the integrators in the implementation of the PVC are discretised using forward Euler, the stability properties and accuracy of the numerical integrations are highly dependent on the sampling rate.

Weighting of derivatives of input signals The problems in the PVC arise as expected when the input signals exhibit fast changes since modelling errors and numerical issues are most evident at these occasions. During simulations it has been noted that weighting of the derivatives of the input signals in the cost function in the optimisation with the parameter η is an efficient way of eliminating the problems in the PVC. A

weighting parameter greater than zero means sub-optimality, *i.e.*, the traverse time of the path is increased compared to the time-optimal. However, satisfactory results in the PVC when performing the path tracking can be achieved by choosing a weighting parameter such that the increase of the traverse time is just a few percent. This shows that the pure time-optimal solution can be used as a reference solution of what is possible to achieve in the robot system, but is hard to realise in practise both in simulations and consequently on the real robot system.

Tuning of weighting parameter in cost function

Finding a suitable trade-off between time-optimality and rate of changes for the input signals requires tuning of the weighting parameter η in the cost function in the optimisation. This parameter determines implicitly how fast the input signals to the robot system are allowed to change. The simulation model offers the possibility to test various choices of the parameter η. Consequently, the procedure of finding a good value is an iterative process from optimisation to simulations in SIMULINK and backwards.

Tuning of parameters in PVC

The simulation model in SIMULINK has also been used for finding suitable values of the parameters k and α in the PVC. The parameter α determines the gain of the internal feedback from the optimised path velocity and the parameter k determines the gain of the adaptation of the path velocity scaling.

Selection of these values has of course to be done such that the path tracking is performed as well as possible and the problems in the PVC described above are avoided. However, if the parameters are selected to be too aggressive, the time-optimality of the path traverse may be compromised. This is especially the case with the selection of the scaling parameter k. During this thesis it has been found advantageous to use not too aggressive tuning in the PVC, but instead use derivative weighting in the cost function in the optimisation. The reason is that the derivative weighting results in that only the fast changes in the input signals are reduced, whereas the scaling in the PVC results in a general scaling of the whole path traverse. Then, it is plausible to believe that the derivative weighting results in more precise elimination of the problems that can occur in the PVC during the optimal path traverse.

5.4 Experimental results from optimal path tracking

Experiments were performed on the robot system with the PVC. Thereby, the force control identification of the path in Figure 4.4 and the results from the optimisation in JModelica.org were used. For visual evaluation of the performance of the path tracking, a ring was attached to the tool, see Figure 5.3.

In the PVC, an internal tracking controller is used to control the robot system based on feedback. This controller can be arbitrarily chosen as long as it can be rewritten to the form (2.30) subjected to path tracking requirements. In the current experiment, the controller was chosen as a combi-

Figure 5.3 In the figure the ring used during experiments with the PVC is shown. The diameter of the ring is 10 mm and the path has a diameter of 5 mm. While traversing the path the ring must not come into contact with the path.

nation of a feedforward controller and a simple feedback controller, similar to [Dahl, 1992]. The feedforward part was determined by inverting the robot system dynamics and is accordingly completely determined by the identified robot model (4.2). Further, the feedback part was chosen as a PD controller. The combined feedforward and feedback control law can be written

$$\dot{q}^r = \underbrace{A^{-1}B\ddot{q}_{ir} + A^{-1}\dot{q}_{ir}}_{\text{feedforward}} + \underbrace{\hat{K}_p(q_{ir} - q) + \hat{K}_d(\dot{q}_{ir} - \dot{q})}_{\text{feedback}} \tag{5.3}$$

where $\dot{q}^r$ is the input signal to the robot system, $\hat{K}_p$ and $\hat{K}_d$ are tuneable controller parameter matrices and subscript ir denotes internal reference values for the controller. Rewriting this control law subjected to path tracking requirements analogous to EXAMPLE 2.2 gives the expression

$$\begin{aligned}\dot{q}^r = {} & A^{-1}B[f''(\sigma)\dot{\sigma}^2 + f'(\sigma)\ddot{\sigma}] + A^{-1}f'(\sigma)\dot{\sigma} + \\ & \hat{K}_p(f(\sigma) - q) + \hat{K}_d(f'(\sigma)\dot{\sigma} - \dot{q})\end{aligned} \tag{5.4}$$

from which the parameters β_1 and β_2 in the parametrisation $\dot{q}^r = \beta_1\ddot{\sigma} + \beta_2$ can be identified. The controller parameters in the PVC used in the experiments on the robot system can be seen in Table 5.2.

Experimental data

The experimental data collected during execution on the IRB140 robot can be seen in Figures 5.4–5.11. In all figures, the independent variable is chosen as the parameter σ, corresponding to the path parameter s in the optimisation. With this choice, the connections between the plots and the corresponding path is clear. From the figures it is also clear that the path tracking is working well. Below, the experimental results are evaluated in more detail in certain aspects.

Table 5.2 Table of parameter values used in the PVC during path tracking experiments on the robot system.

Parameter	Value
$\hat{K}_p$	$\text{diag}(10, 10, 10, 10, 10, 20)$
$\hat{K}_d$	$\text{diag}(1, 1, 1, 1, 1, 1)$
α	15
k	30

Evaluation of the PVC algorithm

The joint positions and corresponding joint velocities of the six joints can be seen in Figure 5.4 and Figure 5.5, respectively. The input signals — *i.e.*, the joint velocity references — to the robot system can be seen in Figure 5.6. This figure allows direct comparison with the corresponding figure, Figure 5.2, obtained in the optimisation with JModelica.org. It can be seen that the time-optimality is preserved by the PVC, because one of the joints is saturated most of the path traverse. It can also be noted that the sixth joint is saturated most of the path traverse, which is a result of the reorientation of the tool along the path.

Path acceleration and path velocity Another measure of the performance of the PVC is the correspondence between the obtained path acceleration $\ddot{\sigma}(\sigma)$ and path velocity $\dot{\sigma}(\sigma)$, see Figure 5.7, with the same variables from the optimisation seen in Figure 5.1. This is a measure of how well the PVC is able to integrate the path acceleration in order to obtain the path velocity $\dot{\sigma}(\sigma)$ and the path parameter σ. Consequently, the integration of the path acceleration is vital in order to traverse the path in an optimal way. The coorespondence in the current case is apparent, even though numerical issues in the PVC can be noted in the results in Figure 5.7. This is especially the case when the path parameter $\sigma = 0.5$ and $\sigma = 0.76$. These numerical issues are further discussed below.

PVC algorithm The numerical issues seen in Figure 5.7 can be derived to the online limit calculations in the PVC. The calculated limits, $\ddot{\sigma}_{max}$ and $\ddot{\sigma}_{min}$, on the path acceleration and the actual path acceleration $\ddot{\sigma}(\sigma)$ can be seen in Figure 5.8. From this figure it is seen that the observed numerical issues in the integration of the path acceleration can be linked to the limit calculations, since the calculated limits are not smooth when $\sigma = 0.5$ and $\sigma = 0.76$. Otherwise, the limit calculations are working well and the calculated limits are not inverted such that $\ddot{\sigma}_{max} < \ddot{\sigma}_{min}$ during the path traverse. The closeness of the actual path acceleration to the limits can also be seen as a measure of the time-optimality of the path tracking. This is because if the path acceleration is saturated, one of the joints is also saturated. It is seen in Figure 5.8 that most of the path traverse, the path acceleration is indeed saturated. This indicates near time-optimality. It can also be noted that the path acceleration is *not* saturated during the fast changes of the input signals. This is a result of the weighting of the derivatives of the input signals in the optimisation formulation.

Internal feedback and path velocity scaling Further, in Figure 5.9 the result of the internal feedback from the path velocity and the path

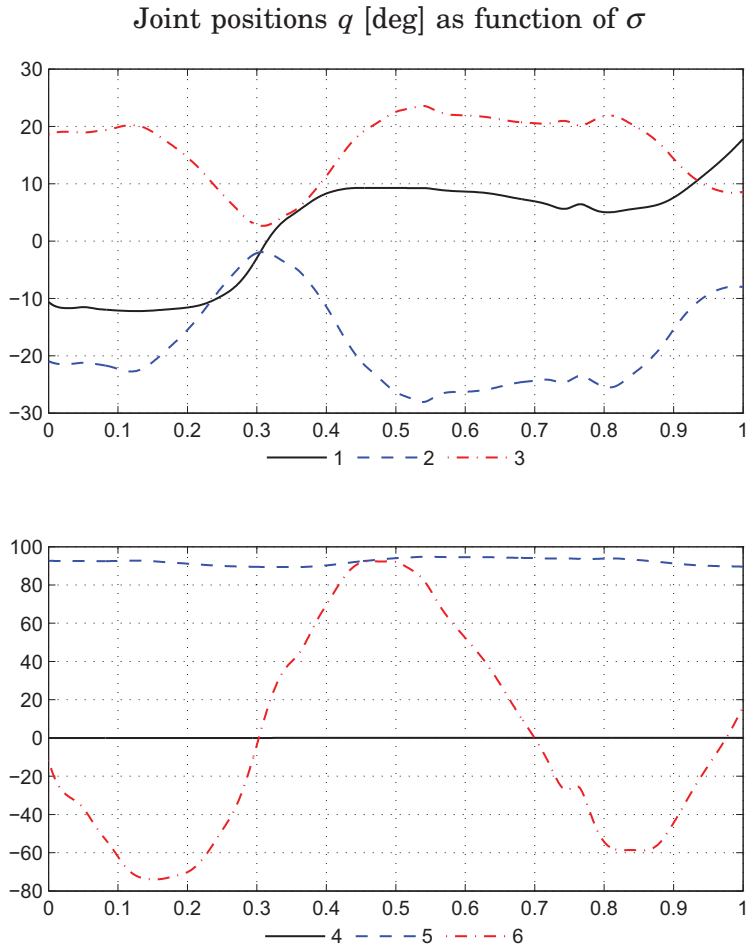

Figure 5.4 The figure displays experimental results obtained from executions on the robot system with the PVC. The figure displays the joint positions q_i, $i = 1, \ldots, 6$, during the path traverse.

velocity scaling is seen. Since the result of the internal feedback is added to the optimised path acceleration, the influence of the internal feedback can be estimated by comparing the magnitude of the internal feedback signal and the magnitude of the path acceleration. This comparison shows that the influence of the internal feedback is reasonable. This can also be seen in Figure 5.7 where the current path velocity $\dot{\sigma}(\sigma)$ tracks the optimised path velocity $\dot{s}(\sigma)$. The parameter α determining the gain of the internal feedback has been tuned in order to get a suitable balance between time-optimality and tracking performance.

The path velocity scaling seen in Figure 5.9 is determined by the parameter k. It is seen that the path velocity scaling algorithm in the PVC makes the scaling parameter γ finally approaching the value 0.985. This indicates that the optimal solution from the optimisation is preserved without too aggressive scaling in the algorithm, since the parameter γ is close to one when the whole path is traversed. Also notice that the scaling parameter γ only decreases, as is expected from the theory chapter, Chapter 2.

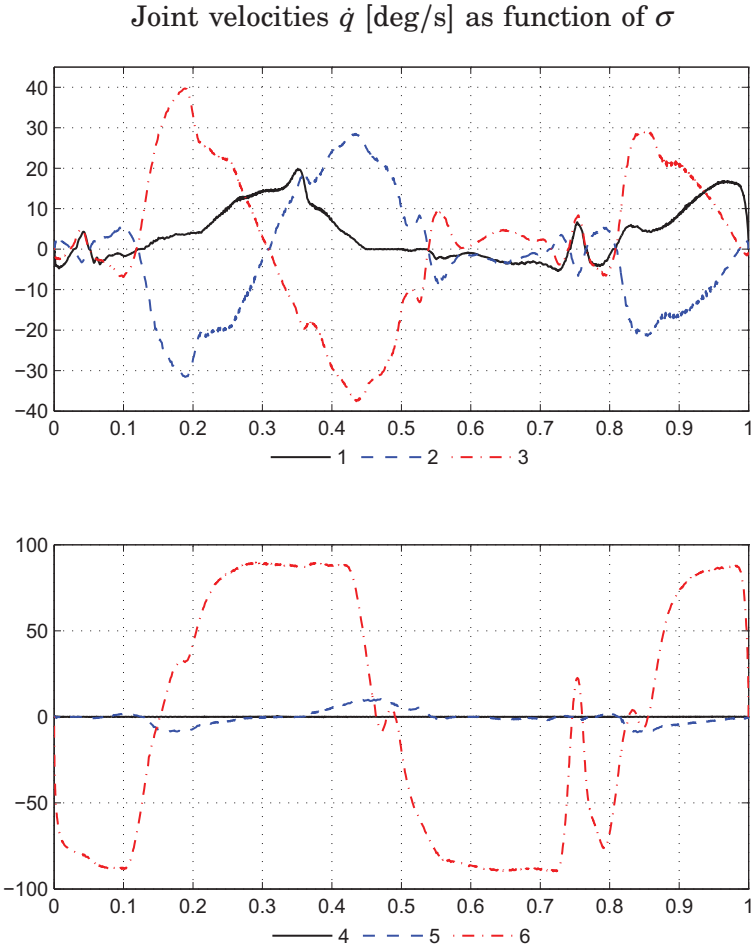

Figure 5.5 The figure displays experimental results obtained from executions on the robot system with the PVC. The figure displays the joint velocities $\dot{q}_i$, $i = 1,\ldots,6$, during the path traverse.

Time-optimality

One of the important questions when performing time-optimal path tracking is the obtained traverse time of the path when performing an experiment on the robot system. In the current experiment, the traverse time is measured to be 6.62 s. This could be compared to the traverse time obtained in the optimisation which is 6.49 s. This means that the traverse time is increased with approximately 2 % in the experiment on the robot system, which has to be considered as reasonable. The deviation in the traverse time can be explained by modelling errors of the robot, numerical issues in the implementation and the path velocity scaling made in the PVC.

Pure time-optimal solution Further, it is interesting to compare the obtained path traverse time with the theoretically pure time-optimal traverse time. Even though the pure time-optimal solution was not used in the PVC on the real robot system, this solution serves as a reference solution of what is possible to achieve in the robot system with the current path and the current limits on the input signals. For the current path, the

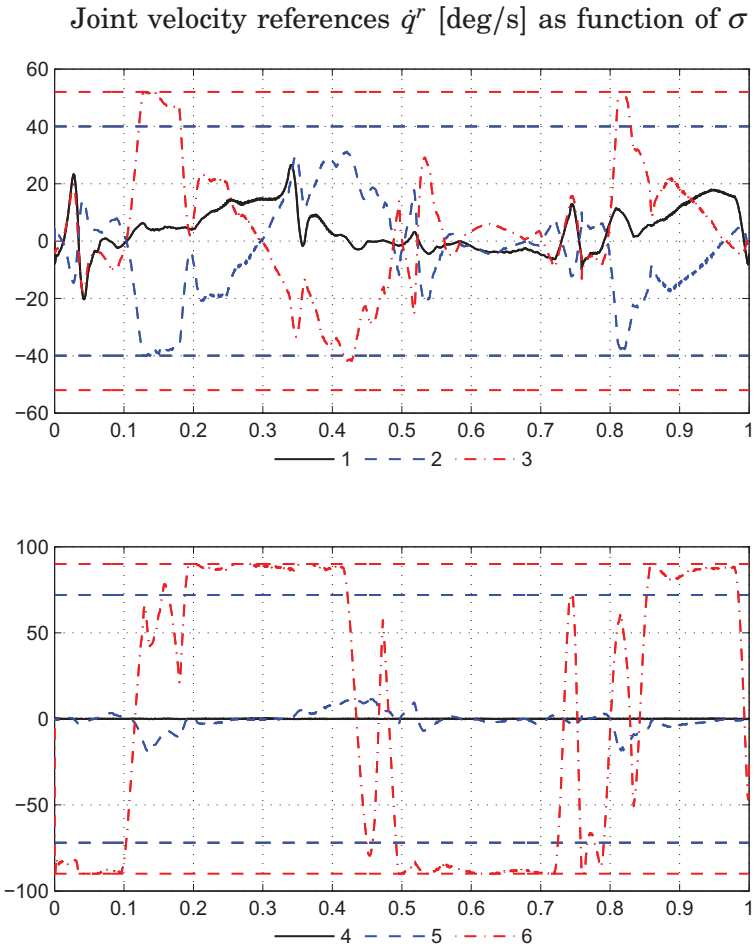

Figure 5.6 The figure displays experimental results obtained from executions on the robot system with the PVC implementation. The plot displays the input signal — *i.e.*, the joint velocity references $\dot{q}_i^r$, $i = 1,\ldots,6$ — as function of σ. The saturation limits, see Table 5.1, for the different joints are also shown in the figure.

pure time-optimal traverse time is 6.30 s. This means that the obtained traverse time in the experiment is approximately 5 % longer than the pure time-optimal.

Tracking performance

In the PVC, an internal controller for the robot system is used to obtain the tracking of the path. The controller used in the current experiment is stated in (5.3). This controller is tuned by changing the gains in the position loop and in the velocity loop. The tracking performance is best described by the plots in Figure 5.10. This plot shows the error for each of the six joints of the robot, measured as the difference between the reference position calculated in the PVC and actual joint position. As expected, the joints with the highest velocities and fast changes have the least accurate tracking performances. This is especially the case with the sixth joint. In an attempt to reduce the tracking error of this joint, the gain of this position loop was increased, compared to the gains in the other five position loops, see Table 5.2. This attempt turned out to reduce the tracking error for the

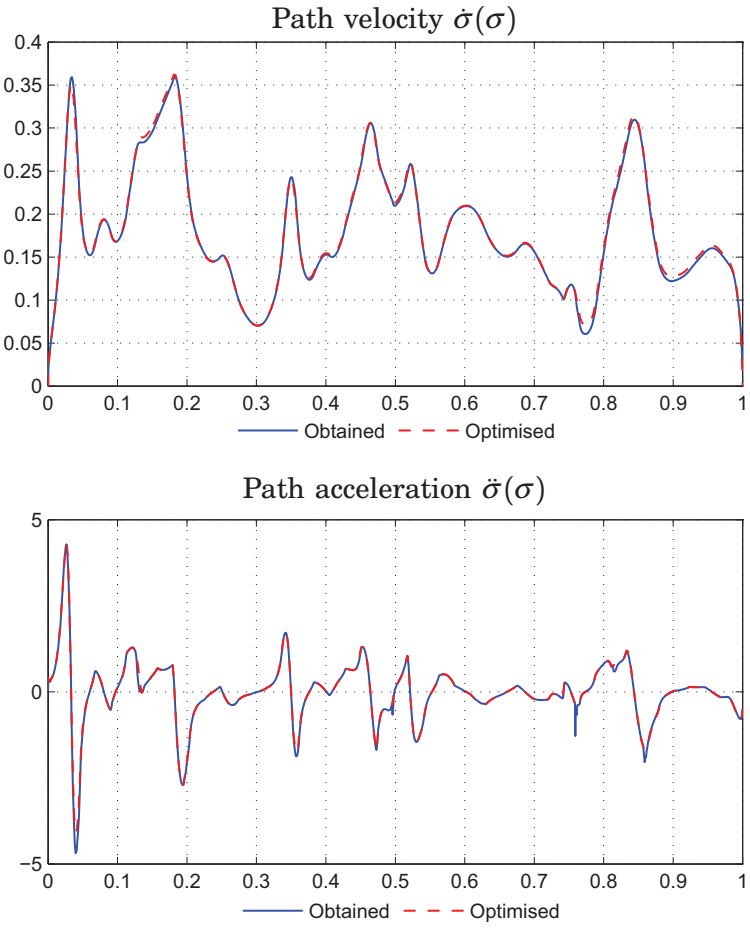

Figure 5.7 The figure displays experimental results obtained from executions on the robot system with the PVC implementation. The upper plot displays $\dot{\sigma}(\sigma)$ and the lower plot displays $\ddot{\sigma}(\sigma)$. The corresponding results from the optimisation can also be seen in the figure.

sixth joint notably.

Velocity loop Further, the velocity loop in the tracking controller is important for the tracking error. This loop can be seen as the prediction part of the controller. Unfortunately, the observed long time delay in the robot system from joint velocity reference to the measured joint velocity limits the possibility to increase the gain of the velocity loop. The joint velocity signal from the robot system also has a notable noise component even though it is filtered with a lowpass filter. This is also an obstacle for increasing the tracking accuracy.

Path tracking The overall behaviour of the optimal path tracking can be seen in Figure 5.11. In this figure, the force control identified path and the path traversed by using the PVC are shown. Figure 5.11 is the most significant figure for evaluation of the path tracking, since it includes all steps and approximations that have been performed from the force control identification of the path to the final path traverse with the PVC. It is seen

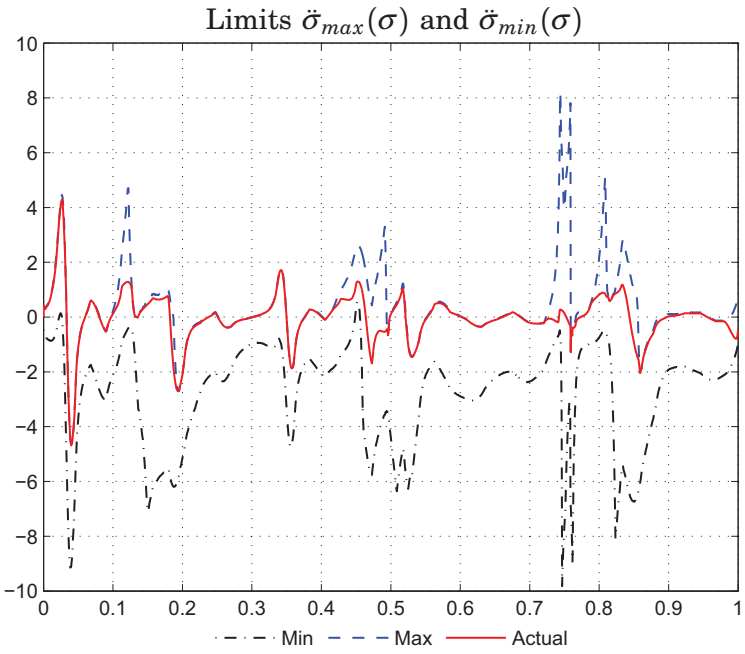

Figure 5.8 The figure displays experimental results obtained from executions on the robot system with the PVC implementation. The plot displays the online calculated limits $\ddot{\sigma}_{min}$ and $\ddot{\sigma}_{max}$. The actual value of $\ddot{\sigma}$ is also shown. Note that the path acceleration is saturated most parts of the path traverse.

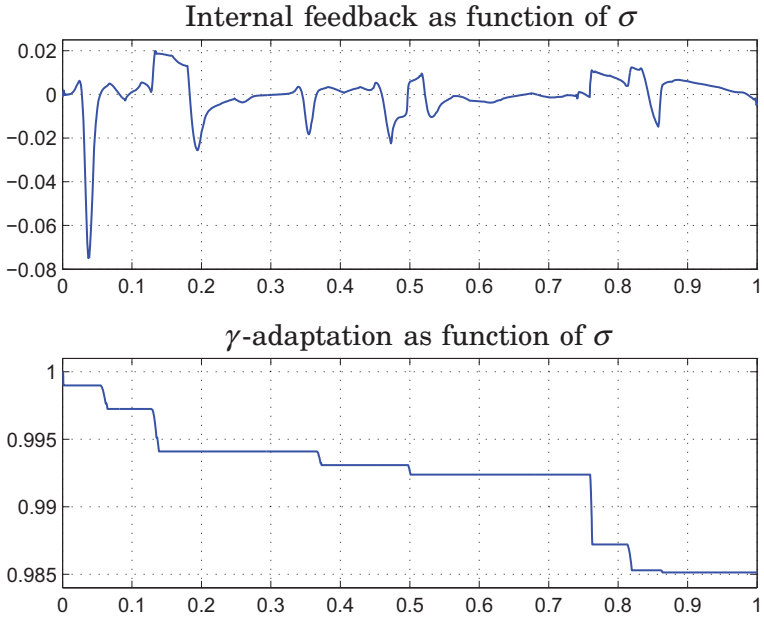

Figure 5.9 The figure displays experimental results obtained from executions on the robot system with the PVC implementation. The upper plot displays the internal feedback from the optimised path velocity in the PVC and the lower plot displays the adaptation of the path velocity scaling with the parameter γ.

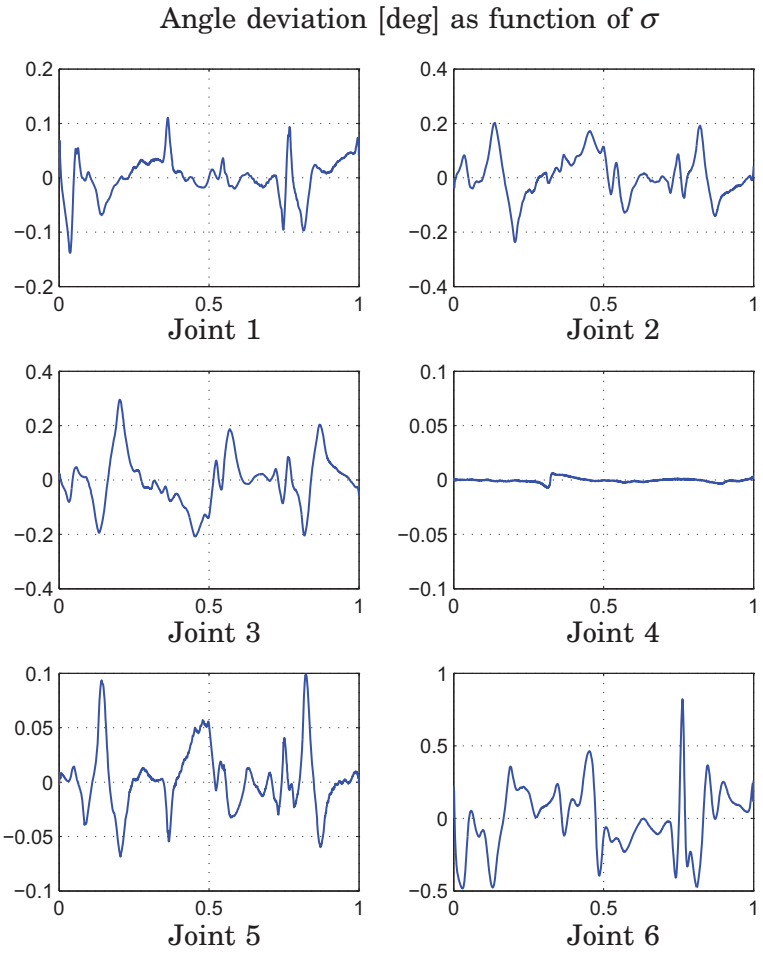

Figure 5.10 The figure displays experimental results obtained from executions on the robot system with the PVC implementation. The plots display the error in the joint positions during the path traverse.

that the tracking is working well, but certain parts of the path are harder to track accurately. A detailed analysis of some of these parts shows that the spline approximations made in the optimisation of the identified path is less accurate in these parts. Further, the detailed analysis showed that the error, at least partially, is rather because of the spline approximations than the tracking performance of the robot.

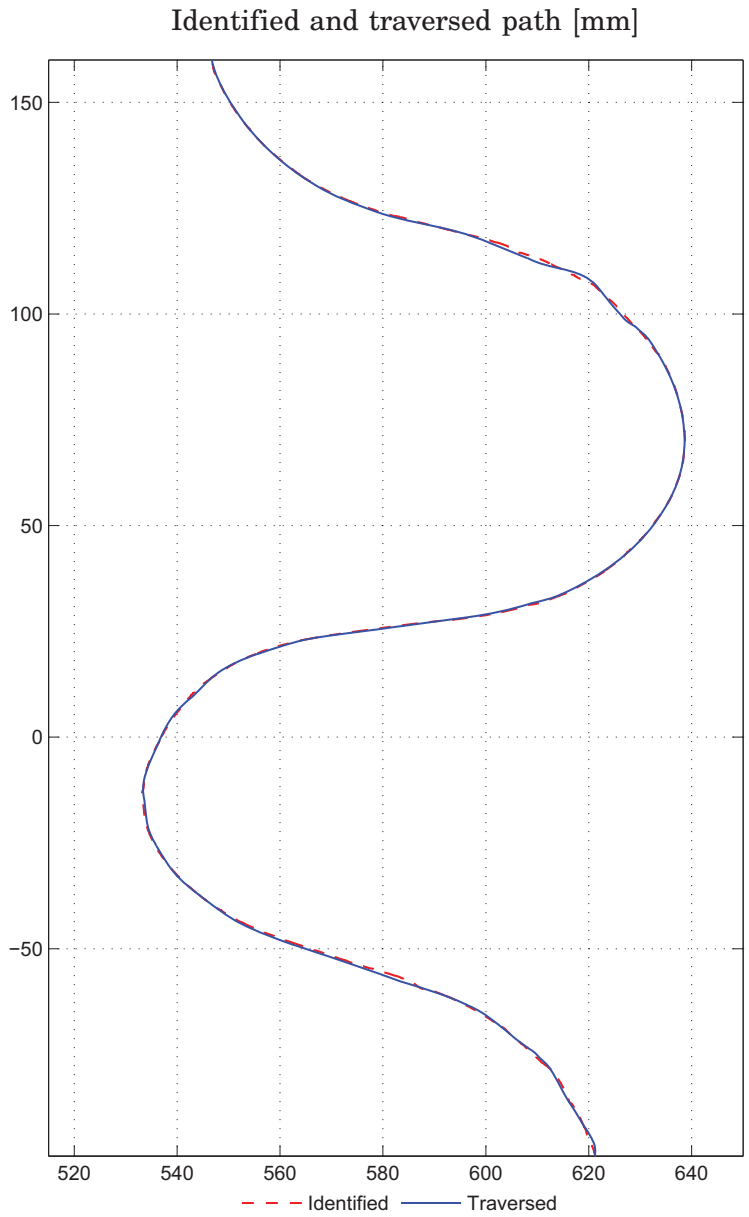

Figure 5.11 The figure displays a comparison of the force control identified path and the optimal path traverse obtained with the PVC. The y-coordinate of the TCP-position is plotted as function of the x-coordinate. The positions are given in the base coordinate system of the robot.

6. Conclusions and future work

This chapter summarises the work that has been performed in this thesis. Also, an outline of possible improvements and future work on the subject of optimal path tracking is given in this chapter. This includes improvements in both the optimisation and in the implementation of the robot control system.

6.1 Conclusions

In this thesis, time-optimal path tracking for industrial robots has been studied. The subject can be considered to consist of three main parts. Firstly, the robot motion has to be decided, such that the tool obtains the correct position and orientation along the path to be tracked. In this thesis this has been done with a contact-force control approach. Secondly, optimisation is made off-line for determining of the input signals to the robot system in order to achieve the path tracking. Finally, a suitable control strategy was implemented, such that the robot system tracks the path online in the face of modelling errors and disturbances.

Path identification

In order to determine the robot motion along the path to be tracked, a contact-force control approach has been utilised. This strategy has proven to be sucessful for the task of path identification since it allows interaction between the robot and the environment, in this case the path itself. Experimental results show that the path is identified with a high accuracy and the orientation of the tool along the path is automatically obtained during the path identification.

Path optimisation

The data obtained from the path identification with force control is used in optimisation software in order to determine suitable control signals to send to the robot system along the path traverse. Mainly, the optimisation software JModelica.org has been used for optimisation purposes. JModelica.org has proved to be well suited for the task. First, it is straightforward to change parameters in the optimisation and second, it is straightforward to change path when a new path identification has been performed. This allows an iterative procedure for obtaining a good path tracking result.

Control of the robot system

In order to obtain a robust time-optimal path tracking online, an earlier developed control structure called path velocity controller has been implemented and experimentally tested in the robot system. By using this structure, the hard constraints on the input signals to the robot system are satisfied during the whole path traverse. Even though the pure time-optimal solution to the path tracking problem has not been able to realise

satisfactory in the PVC, a near time-optimal solution can be used. The sub-optimality measured in the traverse time is only a few percent above the time-optimal path traverse time. Perhaps the pure time-optimal solution is not even desirable since the wear of the robot joints is high in that case.

6.2 Future work

In the future there are certain aspects of the subject that can be developed in order to obtain even more accurate path tracking. The improvements can be made both in the optimisation phase and in the robot control phase. Some of the possible improvements in both of these areas are discussed below.

Robot model and optimisation

Due to the chosen robot model, the convex formulation of the path tracking problem cannot be used. The reason is the viscous friction term in the robot model. Hence, the risk of obtaining a locally optimal solution is possible. This risk is eliminated if the convex formulation can be used, since every optimal solution in that case is also globally optimal. One strategy that can be used in order to be able to use the convex formulation is to identify a robot model in accordance with the rigid body model with the joint torques as input signals. Probably the viscous friction term can be neglected if a more accurate rigid body model is identified on the real robot system.

Another issue is the implementation of the optimisation problem in JModelica.org. To be able to increase the accuracy of the path representation in the optimisation, more intervals in the spline approximations are required. This is difficult in the current implementation since the splines and their derivatives are implemented as a Modelica model and require a lot of code. This calls for a more efficient spline implementation in JModelica.org, where the splines and their derivatives are calculated directly in the software for highest possible efficiency.

Robot control

In order to be able to use the pure time-optimal solution in the PVC without violating the hard constraints on the input signals, perhaps prediction can be introduced as part of the PVC. With prediction it is possible to look ahead and slow down the path traverse before the problems in the PVC occur. Of course, this issue is also related to the quality of the identified robot model, since accurate time-optimal path tracking calls for a good robot model.

Another issue is the delay in the robot system from joint velocity reference to the measured joint velocity. If this delay is reduced it is possible to tune the tracking controller in the PVC more aggressive in order to achieve even better tracking properties. Even though the tracking error is reasonable for the paths studied in this thesis, it can be improved. Improved tracking capacity is also required if the path has for example sharp corners along its way because these are known to be difficult to track without smoothing effects.

7. Bibliography

ABB Robotics (2009): "ABB IRB140 Industrial Robot Data sheet." Data sheet nr. PR10031 EN_R8.

Åkesson, J. (2007): *Languages and Tools for Optimization of Large-Scale Systems*. PhD thesis ISRN LUTFD2/TFRT--1081--SE, Department of Automatic Control, Lund University, Sweden.

Åkesson, J. (2008): "Optimica—an extension of Modelica supporting dynamic optimization." In *6th International Modelica Conference 2008*. Modelica Association.

Åkesson, J., K.-E. Årzén, M. Gäfvert, T. Bergdahl, and H. Tummescheit (2010): "Modeling and optimization with Optimica and JModelica.org—Languages and tools for solving large-scale dynamic optimization problem." *Computers and Chemical Engineering*, January.

Biegler, L. T., A. M. Cervantes, and A. Wächter (2002): "Advances in simultaneous strategies for dynamic process optimization." *Chemical Engineering Science*, **57**, pp. 575–593.

Blomdell, A., I. Dressler, K. Nilsson, and A. Robertsson (2010): "Flexible application development and high-performance motion control based on external sensing and reconfiguration of ABB industrial robot controllers." In *Proceedings of the workshop of "Innovative Robot Control Architectures for Demanding (Research) Applications — How to Modify and Enhance Commercial Controllers", the 2010 IEEE International Conference on Robotics and Automation (ICRA2010)*, pp. 62–66. Anchorage, Alaska, USA.

Bobrow, J. E., S. Dubowsky, and J. S. Gibson (1985): "Time-optimal control of robotic manipulators along specified paths." *The International Journal of Robotics Research*, **4:3**, pp. 3–17.

Boyd, S. and L. Vandenberghe (2004): *Convex Optimization*. Cambridge University Press, Cambridge, UK.

Chen, Y. and A. A. Desrochers (1989): "Structure of minimum-time control law for robotic manipulators with constrained paths." In *Proceedings of 1989 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 971–976. IEEE.

Dahl, O. (1992): *Path Constrained Robot Control*. PhD thesis ISRN LUTFD2/TFRT--1038--SE, Department of Automatic Control, Lund University, Sweden.

De Schutter, J. and H. Van Brussel (1988a): "Compliant robot motion I. A formalism för specifying compliant motion tasks." *The International Journal of Robotics Research*, **7:4**, pp. 3–17.

De Schutter, J. and H. Van Brussel (1988b): "Compliant robot motion II. A control approach based on external control loops." *The International Journal of Robotics Research*, **7:4**, pp. 18–33.

Dressler, I. (2009): "Force control interface for ABB S4/IRC5." Technical Report. Department of Automatic Control, Lund University.

Hast, M. (2009): "Optimal control and path following for industrial robots." Master's Thesis ISRN LUTFD2/TFRT--5842--SE. Department of Automatic Control, Lund University, Sweden.

JModelica.org (2010): http://www.jmodelica.org.

Larsson, P.-O. and R. Braun (2008): "Construction and control of an educational lab process — the gantry crane." In *Reglermöte 2008, Luleå*.

LaValle, S. M. (2006): *Planning Algorithms*. Cambridge University Press. URL: http://planning.cs.uiuc.edu/.

Löfberg, J. (2004): "YALMIP : A toolbox for modeling and optimization in MATLAB." In *Proceedings of the CACSD Conference*. Taipei, Taiwan. URL: http://users.isy.liu.se/johanl/yalmip.

Mattsson, S. E. and G. Söderlind (1992): "A new technique for solving high-index differential-algebraic equations using dummy derivatives." In *Proceedings of the 1992 IEEE Symposium on Computer-Aided Control System Design, CACSD '92*, pp. 218–224. Napa, California.

Nilsson, K. and R. Johansson (1999): "Integrated architecture for industrial robot programming and control." *J. Robotics and Autonomous Systems*, **29:4**, pp. 205–226.

Pfeiffer, F. and R. Johanni (1987): "A concept for manipulator trajectory planning." *IEEE Journal of Robotics and Automation*, **RA-3:2**, pp. 115–123.

Shiller, Z. (1994): "On singular time-optimal control along specified paths." *IEEE Transactions on Robotics and Automation*, **10:4**, pp. 561–566.

Shiller, Z. and H.-H. Lu (1992): "Computation of path constrained time optimal motions with dynamic singularities." *Journal of Dynamic Systems, Measurement, and Control*, **114**, March, pp. 34–40.

Shin, K. G. and N. D. McKay (1985): "Minimum-time control of robotic manipulators with geometric path constraints." *IEEE Transactions on Automatic Control*, **AC-30:6**, pp. 531–541.

Shin, K. G. and N. D. McKay (1986): "A dynamic programming approach to trajectory planning of robotic manipulators." *IEEE Transactions on Automatic Control*, **AC-31:6**, pp. 491–500.

Siciliano, B., L. Sciavicco, L. Villani, and G. Oriolo (2009): *Robotics: Modelling, Planning and Control*. Springer-Verlag, London.

Spong, M. W., S. Hutchinson, and M. Vidyasagar (2006): *Robot Modeling and Control*. John Wiley & Sons.

SUNDIALS (2010): (SUite of Nonlinear and DIfferential/ALgebraic equation Solvers), https://computation.llnl.gov/casc/sundials/main.html.

Toh, K. C., M. J. Todd, and R. H. Tütüncü (1999): "SDPT3 — a Matlab software package for semidefinite programming." *Optimization Methods and Software*, **11**, pp. 545–581. URL: http://www.math.nus.edu.sg/~mattohkc/sdpt3.html.

Verscheure, D. (2009): *Contributions to contact modeling and identification and optimal robot motion planning*. PhD thesis ISBN 978-94-6018-041-5/UDC 681.3*I29, Katholieke Universiteit Leuven, Belgium.

Verscheure, D., B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl (2009): "Time-optimal path tracking for robots: A convex optimization approach." *IEEE Transactions on Automatic Control*, **54:10**, pp. 2318–2327.

Wächter, A. and L. T. Biegler (2006): "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming." *Mathematical Programming*, **106:1**, pp. 25–57.

A. Robot system in Robotics Lab

This appendix presents practical details concerning the robot system available in the Robotics Lab at the Department of Automatic Control, Lund University. As earlier mentioned, the control cabinet used in this thesis, an ABB IRC5, has a redesigned interface that allows execution of a control system implemented in SIMULINK. The procedure of executing the model implemented in SIMULINK on the real robot system is described in this appendix.

A.1 Communication with robot system

Once the model in SIMULINK implementing the control structure has been tested in simulations with satisfactory result, the model can be used to control the robot. In order to control the robot system, there is one group of signals that can be read from the robot, called `irb2ext` and one group of signals that can be sent to the robot, called `ext2irb`. The most important signals in the first group are the joint positions and the corresponding joint velocities. These signals make it possible to use feedback from the robot system. In the second group — *i.e.*, the signals that can be sent to the robot — the most important are the joint position reference values and joint velocity reference values. However, the joint position and joint velocity control can be turned off. Consequently, control with the joint torques as control signals can be made. For a complete list of the signals in `ext2irb` and `irb2ext`, the reader is referred to [Dressler, 2009].

Code generation

In order to execute the SIMULINK controller on the real robot system, the model has to be transformed into C-code and then to an executable file. This is done with the toolbox *Real-Time Workshop* in MATLAB. The robot interface requires that the transformation is done with certain targets and parameters described in [Dressler, 2009]. The targets and parameters aim at creating a program compatible with the architecture in the robot system. The current architecture in use was developed at Lund University and is called Open Robot Control Architecture, abbreviated ORCA [Nilsson and Johansson, 1999; Blomdell *et al.*, 2010].

Graphical User Interface

The SIMULINK model implementing the controller is built on a personal computer running Fedora 11. Communication between the PC and the main computer in the control cabinet of the robot is facilitated by a graphical user interface called Opcom, see Figure A.1. Opcom has two command windows; one is for communication with the main computer in the control cabinet, a G4 Power PC, and the second is for communication with the control cabinet IRC5 itself. The user loads the controller, in the form of the executable C-code, with Opcom and then the controller is installed in the control cabinet.

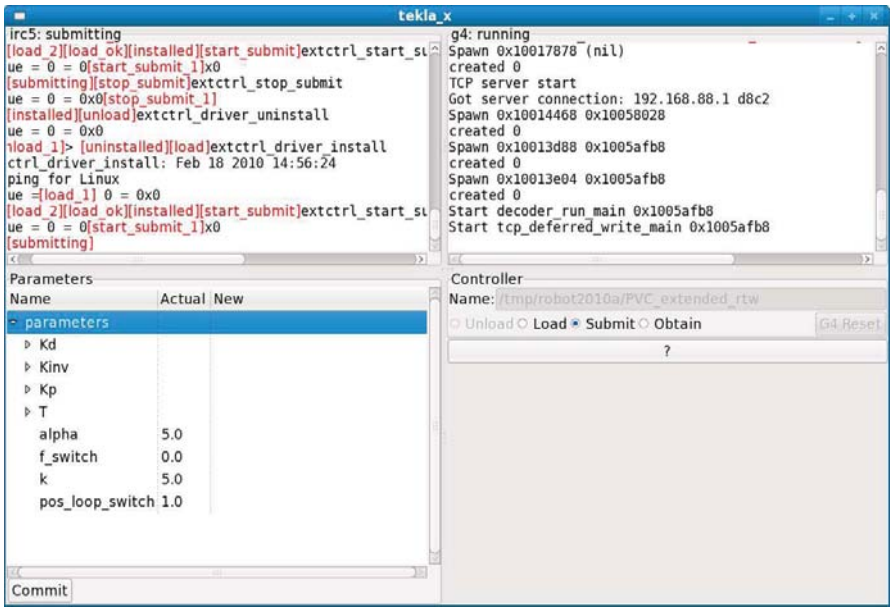

Figure A.1 In the figure the graphical user interface for communication with the robot system is shown.

In the SIMULINK model, parameters can be classified as inline parameters. By defining a parameter as an inline parameter, the parameter can be modified in Opcom before or during execution of the controller on the robot system.

As an intermediate step, the loaded controller can be tested in a state called *submit*, where the `irb2ext` signals are read, but the `ext2irb` signals are not sent to the robot. This makes it possible to test the controller before sending signals to the robot. Finally, the controller is fully executed by switching Opcom to the state *obtain*. Then, experimental results can be collected during execution of the control system.

A.2 Robot system

The control cabinet for the robot system, ABB IRC5, mainly consists of one main computer and a so called axis computer. The main computer in the control cabinet is run with a frequency of 250 Hz, which equals a sampling period of $h = 4$ ms. The axis computer controls six axis controllers, where each controller is responsible for one robot joint. The axis computer runs at the higher frequency 2 kHz, which equals a sampling period of $h = 0.5$ ms.

Logging of signals

In order to log and save signals during execution of a controller on the real robot system, a certain program for logging is available in the Robotics Lab. The signals to be logged have to be specified in the SIMULINK model. More specificly, this is done by marking the signals as test points in SIMULINK. The logging is started by executing the logging program. Thereby, the number of seconds to be logged and the number of data that should be saved must be specified. The log file created when the logging is ready is then transformed into a format that can be read and plotted in MATLAB.

Velocity control

During the path tracking implemented in this thesis, the joint position control loops in all six joints in the robot are turned off. The reason is that the joint velocity reference is considered as the input signal. Hence, the position loops are turned off by setting the gains of the P controllers in the cascaded joint control structure in Figure 4.1 to zero. The corresponding signal to use in the SIMULINK model for setting the gains to zero is called `parKp`.

Even though the gains in the P controllers are set to zero, the safety system in the robot system does not allow that the deviations of the current joint positions from the position references are too large. In the case of a too large deviation, the robot system locks the brakes of the robot and further movement is not possible. Hence, in the current application the measured joint positions are sent as position references to the robot system along the path tracking. With this approach, the deviation of the reference signal from the current position is kept small for all joints.

Torque control

In order to use the joint torques as control signals for the robot, the position and velocity loops have to be turned off. This can be achieved in the same manner as described above for obtaining velocity control. The position loops are turned off by setting the gains of the P controllers to zero. Likewise, the velocity loops are turned off by setting the gains of the P–parts and the gains of the I–parts to zero. The parameters in the velocity loop can be altered with the signals `parKv` and `parKi` in the SIMULINK model. Then, these controllers do not give any contribution to the torque applied on the robot. Instead, the joint torques can be controlled directly by applying a feedforward torque signal. The corresponding output signal to be used in the SIMULINK model is called `trqFfw`. For the same reasons as described above for velocity control, the current joint position has to be fed to the position reference signal for each joint.

A.3 Force sensor

In order to use force control of the robot, a force sensor measuring the force exerted on it has to be attached to the robot flange. The force sensor measures forces acting on it in three perpendicular directions and also the corresponding torques acting on it. In the Robotics Lab a force sensor from the American company JR3 is available. The raw measurements from the force sensor are processed by a separate computer in the Robotics Lab. The same computer links the measurements from the sensor to the loaded controller in the control cabinet. Hence, the force measurements can be utilised in the control system implemented in SIMULINK and force control is thus possible. In order to use the force measurements in the SIMULINK controller the input `jr3_comedi` is used.

B. Code listings

B.1 Modelica code for example in Chapter 3

Listing B.1 Modelica code solving the path tracking problem studied in Chapter 3.

```
1  // Robot dynamics expressed in the modelling language
2  // Modelica
3
4  model robDyn
5      // Masses
6      parameter Real m1 = 1;
7      parameter Real m2 = 1;
8      // Torques
9      Real tau1(min=-1,max=1);
10     Real tau2(min=-1,max=1);
11     // Spline describing the path and its derivatives
12     spline spl;
13     // Variable a
14     input Real a;
15     // Variable b, start in rest
16     Real b(start=0,fixed=true);
17 equation
18     tau1 = m1*spl.dtraj1*a + m1*spl.ddtraj1*b;
19     tau2 = m2*spl.dtraj2*a + m2*spl.ddtraj2*b;
20     der(b) = 2*a;
21 end robDyn;
22
23 // Formulation of the optimisation problem in Optimica
24 // Path parameter s serves as pseudo time in the
25 // optimisation problem
26
27 optimization ellips_Opt (objective=cost(finalTime),
28         startTime=0,finalTime=2*Modelica.Constants.pi)
29     // Cost function
30     Real cost(start=0,fixed=true);
31     // Variable a
32     input Real a_opt;
33     // Object of the type robDyn
34     robDyn rd(a(free=true));
35 equation
36     rd.spl.t__ = time;
37     // Cost function specified by its derivative
38     der(cost) = 1/(sqrt(rd.b + 1e-4));
39     a_opt = rd.a;
40 constraint
41     // Ends in rest
42     rd.b(finalTime)=0;
43     // Positive path velocity
44     rd.b >= 0;
45 end ellips_Opt;
```

```
46
47 // Model for creating intial values to the optimisation
48
49 model ellips_Init_Opt
50     // Cost function
51     Real cost(start=0,fixed=true);
52     input Real a_opt;
53     // Object of the type robDyn
54     robDyn rd(a(free=true));
55 equation
56     rd.spl.t__ = time;
57     // Cost function specified by its derivative
58     der(cost) = 1/(sqrt(rd.b + 1e-4));
59     a_opt = rd.a;
60 end ellips_Init_Opt;
61
62 // Model representing the path and its derivatives
63 // with splines
64
65 model spline
66     Real t__;
67     Real traj1;
68     Real traj2;
69     Real dtraj1;
70     Real dtraj2;
71     Real ddtraj1;
72     Real ddtraj2;
73 equation
74     // If- and else clauses implementing the splines
75 end spline;
```

B.2 Modelica code for path tracking problem in Chapter 5

Listing B.2 The Modelica code implementing the optimisation problem used for optimal path tracking on the robot system in Chapter 5.

```
1  // A contact-force identified path with identified
2  // robot model
3
4  model robDyn
5      // 20 % of maximum velocity according to manufacturer
6      parameter Real velScal = 0.2;
7      Real velRef[6](min = velScal*{-3.4907,-3.4907,-4.5379,
8                     -6.2832,-6.2832,-7.8540},
9                  max = velScal*{3.4907,3.4907,4.5379,
10                    6.2832,6.2832,7.8540});
11     robot_spline rspl;
12     // Robot dynamics
13     parameter Real A[6] = {1,1,1,1,1,1};
14     parameter Real B[6] = {0.1466,0.1483,0.1513,
15             0.1482,0.1509,0.1459};
16     // Path acceleration and path velocity
17     Real dds;
```

```
18     Real ds(start=0,fixed=true,min=0);
19     Real b(min=0);
20
21  equation
22     for i in 1:6 loop
23        velRef[i] = 1/A[i] * (B[i]*(rspl.dtraj[i]*dds +
24              rspl.ddtraj[i]*ds^2) + rspl.dtraj[i]*ds);
25     end for;
26     der(b) = 2*dds;
27     b = ds^2;
28  end robDyn;
29
30
31
32  optimization path_tracking_Opt (objective=
33              cost(finalTime),startTime=0,finalTime=1)
34     robDyn rd;
35     Real cost(start=0,fixed=true);
36     input Real dds_opt;
37     Real trav_time(start=0,fixed=true);
38     parameter Real eta = 1.0e-4;
39  equation
40     dds_opt = rd.dds;
41     rd.rspl.t__ = time;
42     // Cost function with derivative weighting
43     der(cost) = 1/sqrt(rd.b + 1e-4) + eta * (
44           der(rd.velRef[1])^2 + der(rd.velRef[2])^2 +
45           der(rd.velRef[3])^2 + der(rd.velRef[4])^2 +
46           der(rd.velRef[5])^2 + der(rd.velRef[6])^2 );
47     der(trav_time) = 1/sqrt(rd.b + 1e-4);
48  constraint
49     rd.ds(finalTime)=0;
50  end path_tracking_Opt;
51
52
53
54  optimization path_tracking_first_Opt (objective=
55                cost(finalTime),startTime=0,finalTime=1)
56     robDyn rd;
57     Real cost(start=0,fixed=true);
58     input Real dds_opt;
59  equation
60     dds_opt = rd.dds;
61     rd.rspl.t__ = time;
62     // Pure time-optimal cost function
63     der(cost) = 1/sqrt(rd.b + 1e-4);
64  constraint
65     rd.ds(finalTime)=0;
66  end path_tracking_first_Opt;
67
68  // Model for creating intial values to the optimisation
69
70  model path_tracking_Init_Opt
71     robDyn rd;
72     Real cost(start=0,fixed=true);
73     input Real dds_opt;
```

```
74 equation
75     dds_opt = rd.dds;
76     rd.rspl.t__ = time;
77     der(cost) = 1/sqrt(rd.b + 1e-4);
78 end path_tracking_Init_Opt;
79
80 // Model representing the path and its derivatives
81 // with splines
82
83 model robot_spline
84     Real t__;
85     Real traj[6];
86     Real dtraj[6];
87     Real ddtraj[6];
88 equation
89     // If- and else clauses implementing the splines
90 end robot_spline;
```

C. Simulink implementations

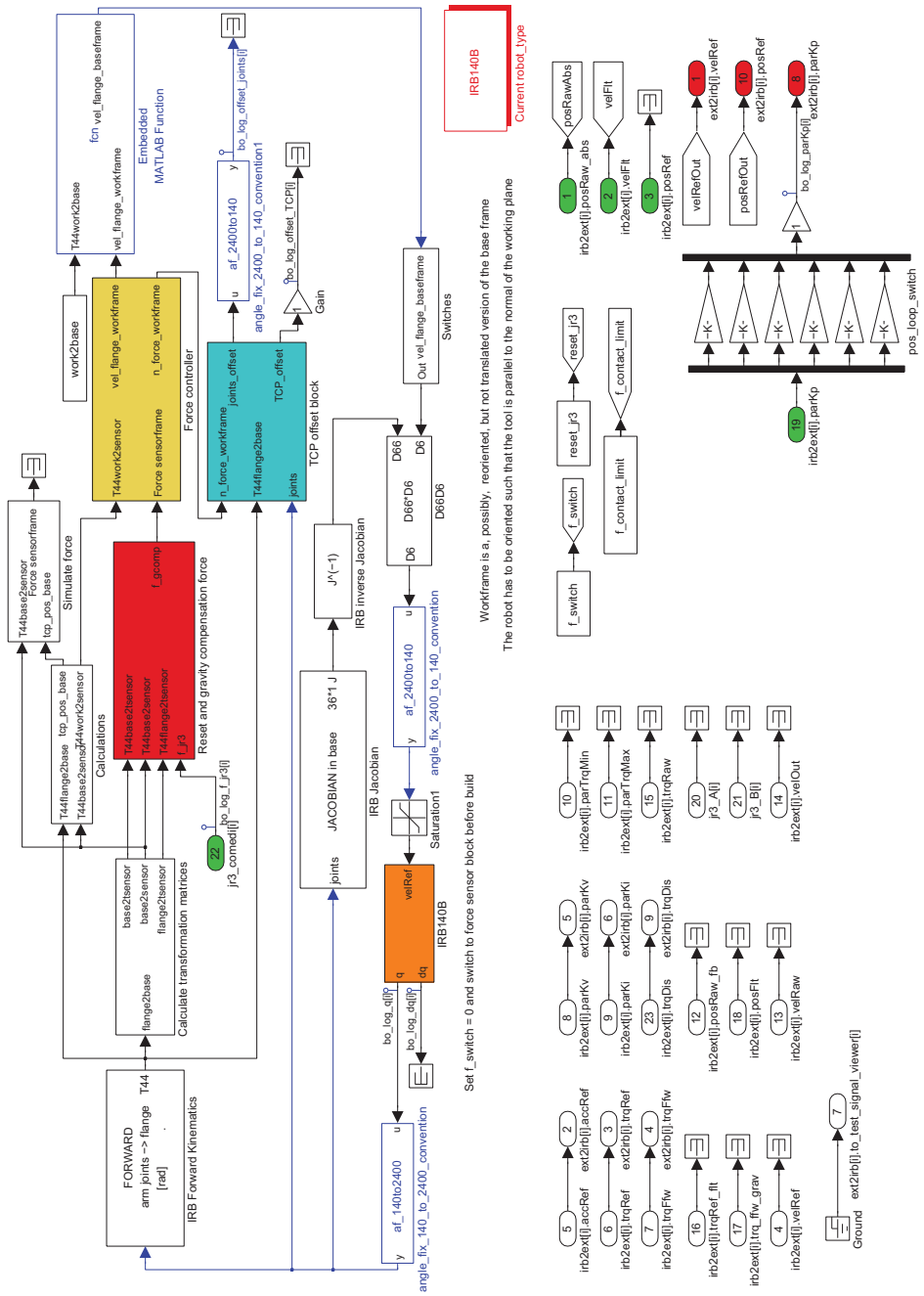

Figure C.1 Implementation of force controller in SIMULINK.

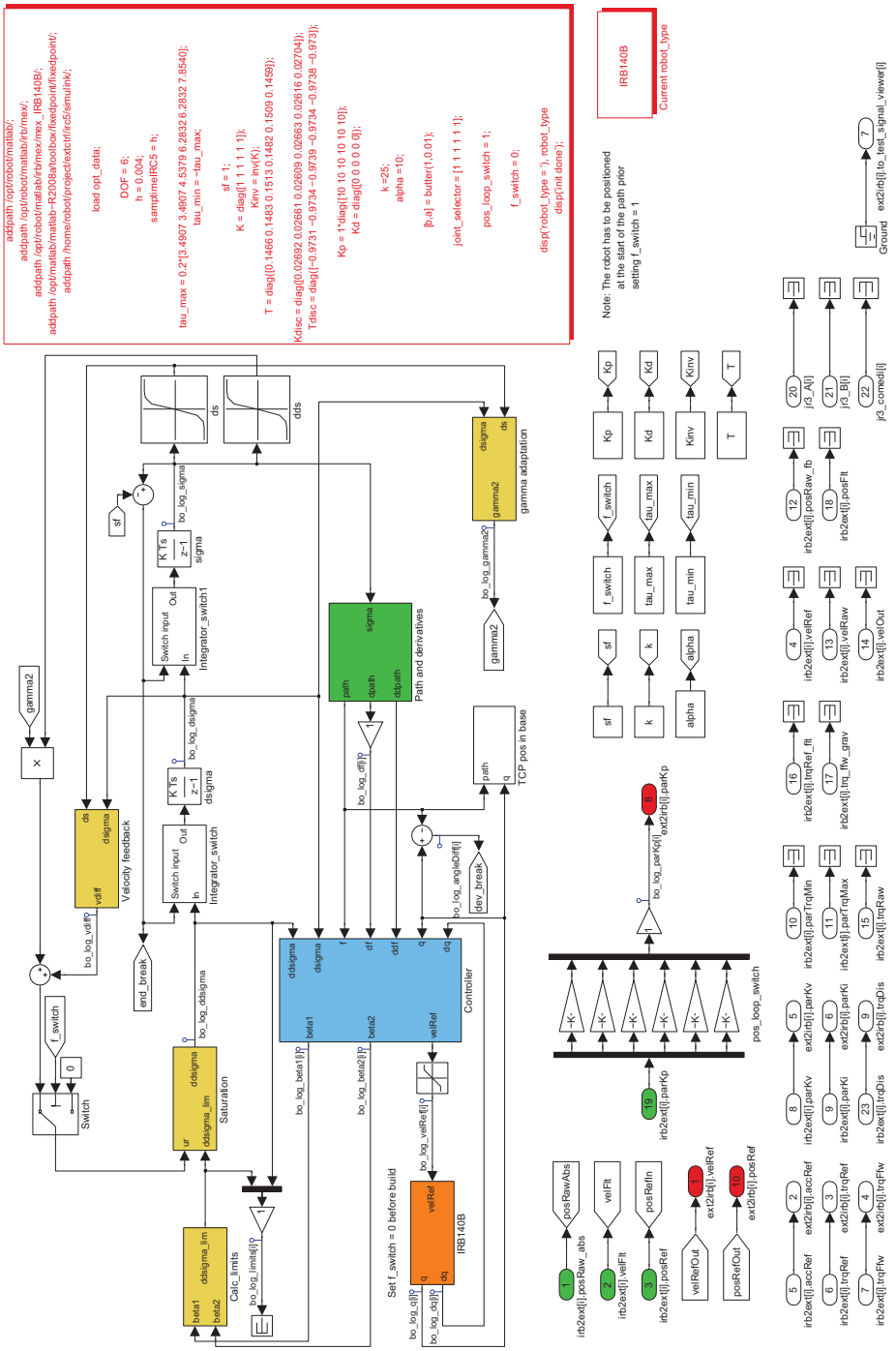

Figure C.2 Implementation of path velocity controller in SIMULINK.