Path Constrained Robot Control

Ola Dahl

Department of Automatic Control, Lund Institute of Technology

(Lejon)
(Ø 35 mm)

ISSN 0280–5316
ISRN LUTFD2/TFRT--1038--SE

Path Constrained Robot Control

Ola Dahl

Lund 1992

To Emma

Department of Automatic Control
Lund Institute of Technology
Box 118
S-221 00 LUND
Sweden

ISSN 0280–5316
ISRN LUTFD2/TFRT--1038--SE

Printed in Sweden by KF-Sigma
Lund 1992

Department of Automatic Control **Lund Institute of Technology** P.O. Box 118 S-221 00 Lund Sweden	*Document name* DOCTORAL DISSERTATION *Date of issue* April 1992 *Document Number* ISRN LUTFD2/TFRT--1038--SE
Author(s) Ola Dahl	*Supervisor* Lars Nielsen, Karl Johan Åström *Sponsoring organisation* Swedish National Board for Technical Development (NUTEK), contract 87-01384P

Title and subtitle

Path Constrained Robot Control

Abstract

Fast motion along a predefined path is important in many robot applications, and requires utilization of the maximum allowable torque range. If the torque is at the limit, there is no margin to cope with disturbances or modeling errors, which may result in deviation from the path. A path velocity controller outside the ordinary robot controller modifies a nominal velocity profile, which is specified in advance, e.g. by minimum time optimization or by a robot operator/programmer. Existing minimum time optimization methods for rigid robots give the nominal velocity profile and also insight into the constraints. Flexible joint robots are treated by a polynomial approximation of the robot velocity along the path, and the specific approximation is chosen such that the boundary conditions on zero velocity and nonzero torque are satisfied.

The path velocity controller modifies a scalar path parameter, giving computational efficiency and coordination of joint motions. A basic algorithm limits the acceleration, which may be inadmissible due to modeling errors. An extended algorithm handles the added problem of inadmissible velocity. The path velocity controller is verified by experiments on an industrial robot. The experiments are done such that it is possible to separate the performance of the path velocity controller from the performance of the robot controller. Path deviation and torque utilization are discussed and evaluated. The experiments show that the path velocity controller can adjust a nominal minimum time velocity profile, such that the result is good path following and good utilization of the available torque range.

The use of path velocity control makes it possible to have a nominal velocity profile which exceeds the robot capability. This is not possible if the reference trajectory is fixed. The experimental results also show how path velocity control makes it possible to use minimum time optimization in a nonideal situation.

Key words
Robot control, path following, torque limits, trajectory planning, minimum time optimization, velocity profile, feedback, reference trajectory generation, path velocity control, experimental evaluation.

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280–5316		*ISBN*
Language English	*Number of pages* 177	*Recipient's notes*
Security classification		

The report may be ordered from the Department of Automatic Control or borrowed through the University Library 2, Box 1010, S-221 03 Lund, Sweden, Telex: 33248 lubbis lund.

Contents

Preface

This work started in 1987, and has since then been financially supported by the Swedish National Board for Technical Development (NUTEK), under contract 87-01384P. Professor Lars Nielsen has been my supervisor. Lars has provided continuous guidance and support during the work. He has been of great help in the production of the thesis, both in the structuring of the manuscript, and in the consideration of small details. It is a great pleasure to work with Lars, and I want to express my gratitude to him. I also want to thank Professor Karl Johan Åström and Docent Per Hagander for their help. Karl Johan has been a source of inspiration in valuable discussions, and his comments and suggestions have clearly improved the presentation of the material. Per has provided constructive criticism on both the thesis manuscript and on the predecessor, the Licentiate thesis. Rewarding discussions with Per have led to substantial improvements in Chapter 3. I am also grateful to Ulf Holmberg for his comments on the manuscript.

The efforts of Klas Nilsson and Rolf Braun have made the experimental work possible. In addition, I want to thank Klas for many valuable discussions on different aspects of industrial robots. I also want to thank Eva Dagnegård for helping me with the preparation of the final manuscript. Kjell Gustafsson, always being one step-ahead in the thesis preparations, has given me important pieces of information at the right moments. The excellent computer facilities, administrated by Leif Andersson and Anders Blomdell, have also been important for the thesis work. Finally, I want to thank all my colleagues for making the Department of Automatic Control a stimulating and friendly place to work in.

O.D.

1

Introduction

Motion along a predefined path is common in robotics. Typical examples are gluing, arc welding, and laser cutting. There are several occasions in these applications, where the robot performance is limiting the production speed, and it is natural to look for a time optimal solution along the path. The path is given from the application and a first step is to obtain a nominal motion specification along the path. This can e.g. be done by a robot operator/programmer. However, if a rigid body model is available, minimum time optimization can be used [Bobrow *et al.*, 1985; Shin and N.D.McKay, 1985; Pfeiffer and R.Johanni, 1986]. The optimization algorithm computes a velocity profile, i.e. the velocity is expressed as a function of the position along the path. Using such a velocity profile implies utilization of the maximum allowable torque range. However, if the torque is at the limit, there is no margin to cope with disturbances or modeling errors. An approach to solve this robot path following problem is presented, where a path velocity controller is used for modification of a nominal velocity profile. The path velocity controller is used outside the ordinary robot controller, which is assumed available. The path velocity controller modifies the reference trajectory for the robot controller. The path velocity control concept was first presented in [Dahl and Nielsen, 1989; Dahl and Nielsen, 1990b], where it is called trajectory time scaling. The time scaling view is advantageous when designing the feedback laws and for tuning of parameters. It is also described in [Dahl and Nielsen, 1990a] and [Dahl, 1989].

A path following example is presented in Chapter 2, where also notations and terminology are introduced. Especially, the representation of a trajectory as a time function and as a velocity profile, is introduced and exemplified. An example of a minimum time optimization is given, and the sensitivity to modeling errors is demonstrated. The use of path velocity control is discussed, and illustrated in a simulation.

Chapter 3 presents methods for minimum time optimization of the nominal velocity profile. A review of minimum time optimization for rigid robots is given. The presentation includes the specific optimization algorithm, and a description and interpretation of the underlying constraints on velocity and acceleration. The constraints, which are obtained by combining the torque constraints, the robot dynamics, and the path constraint, provide useful information for the design of the path velocity controller, and are also useful when interpreting the result of using path velocity control. The minimum time solution for rigid robots is bang-bang in the sense that at least one torque is always at the limit. An optimality result shows that this property carries over to the practically important case of joint flexibility. The flexible joint model used is the simplified model given in [Spong, 1987].

The optimization algorithm for rigid robots [Bobrow *et al.*, 1985; Shin and N.D.McKay, 1985; Pfeiffer and R.Johanni, 1986] is a tractable way to obtain the minimum time solution. The algorithm can however not be extended to the case of joint flexibility. Another approach, which is approximate, but can be used both for rigid and flexible joint robots, is to use polynomial approximation to obtain a parametric optimization problem. A method based on this idea is presented. A function of the robot velocity along the path is approximated. The choice of function to approximate is considered, and a choice which allows zero initial and final velocity, and initial and final torques different from zero is given. The method is illustrated by numerical examples. The examples demonstrate the properties of the solution, showing the bang-bang structure, which is present also in the rigid case, but also showing how the velocity profile of the flexible solution differs from the rigid velocity profile.

Chapter 4 presents algorithms for path velocity control. A parametrization of the robot controller in the path parameter is presented. The parametrization allows computation of limits on the path acceleration. These limits are used for adjustment of the nominal path acceleration, which may be inadmissible due to modeling errors. A basic algorithm is obtained by combining the acceleration limits with feedback using the nominal velocity profile. An extended algorithm, designed to handle the added problem of inadmissible path velocity, is also given. The algorithm uses a scaling of the velocity profile, and the scaling factor is modified by feedback. The path velocity controller is evaluated by analysis and simulations. The nominal velocity profiles used in the simulations are computed using minimum time optimization. The path velocity control algorithm is not dependent on this choice. The choice of minimum time gives however an especially demanding velocity profile, and is therefore considered throughout the thesis.

An experimental evaluation on an industrial robot is presented in Chapter 5. The experiments are done such that it is possible to separate the performance of the path velocity controller from the performance of the robot controller. Evaluation measures are proposed. Path deviation is evaluated using visual inspection, and a quantitative measure of the torque utilization is suggested. A decoupled rigid body model is obtained from system identification, and minimum time optimization is used to compute the nominal velocity profile. The properties of the minimum time solution, described in Chapter 3, are used for interpretation of the result of using path velocity control. Experimental tests show how the path velocity controller can adjust a nominal minimum time velocity profile such that the modified velocity profile gives good path following and good utilization of the available torque range. It is also demonstrated how the path velocity controller is able to compensate for time variations in the robot dynamics.

2

An Example

An example is used as an introduction to the detailed presentation given in the following chapters. The example also introduces notations and terminology.

Robot Model

The robot model used is a decoupled linear model, given by

$$M\ddot{q} = \begin{pmatrix} m_1 & 0 \\ 0 & m_2 \end{pmatrix} \begin{pmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{pmatrix} = \begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix} = \tau \tag{2.1}$$

The parameters m_1 and m_2 represent the mass of the robot, q_1 and q_2 are the joint positions, and τ_1 and τ_2 are the input torques.

Torque Constraints

The torque constraints are constant upper and lower limits for each torque. The constraints are expressed by the inequalities

$$\tau_i^{min} \leq \tau_i \leq \tau_i^{max}, \quad i = 1, 2 \tag{2.2}$$

This type of torque constraints are used throughout the thesis.

Path Description

The path is represented as a parametrized curve $f(s) \in \mathbb{R}^2$ where s is a scalar path parameter. In this example, the path is a straight line segment, described by

$$f(s) = \begin{pmatrix} f_1(s) \\ f_2(s) \end{pmatrix} = \begin{pmatrix} \alpha_1 s \\ \alpha_2 s \end{pmatrix}, \quad s_0 \leq s \leq s_f \tag{2.3}$$

where s_0 and s_f represent the beginning and the end of the path.

Trajectories

A trajectory is obtained from the path by specifying the path parameter s as a function of time. The function $s(t)$ is defined on the interval $[t_0, t_f]$ where $s(t_0) = s_0$ and $s(t_f) = s_f$. Since we assume that the path is fixed, i.e. the function $f(s)$ is given, the trajectory $f(s(t))$ can be represented by the path parameter $s(t)$. We assume that the path parameter $s(t)$ is piecewise twice differentiable with respect to time. The first derivative $\dot{s}(t)$ is called the path velocity and the second derivative $\ddot{s}(t)$ is called the path acceleration. Further, we assume that $\dot{s}(t) > 0$ for $t_0 < t < t_f$. This means that we only consider trajectories which represent forward motion along the path.

The assumption $\dot{s}(t) > 0$ implies that $\dot{s}(t)$ can be expressed as a function of $s(t)$. This function is called the velocity profile and is denoted $v_1(s)$. The function $v_1(s)$ is defined on the interval $s_0 \leq s \leq s_f$ and is given from $\dot{s}(t)$ and $s(t)$ as

$$\dot{s}(t) = v_1(s(t)) \tag{2.4}$$

Similarly, the path acceleration $\ddot{s}(t)$ can be expressed as a function of $s(t)$. This gives the acceleration profile $v_2(s)$, defined by

$$\ddot{s}(t) = v_2(s(t)) \tag{2.5}$$

A relation between the profiles $v_1(s)$ and $v_2(s)$ is found by writing $\ddot{s}(t)$ as

$$\ddot{s}(t) = \frac{d}{dt} v_1(s(t)) = v_1'(s(t))\dot{s}(t) = v_1'(s(t))v_1(s(t))$$

which shows that

$$v_2(s) = v_1'(s)v_1(s) = \frac{d}{ds}\frac{1}{2}v_1(s)^2 \tag{2.6}$$

The velocity profile v_1 and the acceleration profile v_2 are used in Chapter 4 to represent the nominal trajectory in the path velocity controller.

2.1 Path Velocity Planning

Path velocity planning is the computation of the path parameter s as a function of time. Optimization can be used to compute $s(t)$ and a typical criterion is minimum time. A minimum time optimization for the chosen example is formulated as: minimize the traversal time

$$\int_{t_0}^{t_f} dt \tag{2.7}$$

subject to the path constraint $q = f(s)$, where $f(s)$ is defined in (2.3), the torque constraints (2.2), and the dynamic constraints (2.1). This is an optimal control problem for a fourth order system (2.1). Two of the state variables, q_1 and q_2, are constrained by equality constraints $q = f(s)$. The inputs are constrained by (2.2). The optimal control problem can be reformulated as a second order problem by substituting the path constraint $q = f(s)$ into the robot dynamics (2.1). Using the chain rule for differentiation gives

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

Substituting into the robot dynamics (2.1) now gives

$$Mf'(s)\ddot{s} + Mf''(s)\dot{s}^2 = \tau \tag{2.9}$$

For the path (2.3), we get

$$Mf'(s) = \begin{pmatrix} m_1\alpha_1 \\ m_2\alpha_2 \end{pmatrix}, \quad Mf''(s)\dot{s}^2 = 0 \tag{2.10}$$

which, substituted into (2.9), gives the torque vector

$$\tau = Mf'(s)\ddot{s} = \begin{pmatrix} m_1\alpha_1 \\ m_2\alpha_2 \end{pmatrix} \ddot{s} \tag{2.11}$$

The minimum time optimization of (2.1) along the path (2.3) is now formulated as an optimal control problem for a double integrator with states s and $\dot{s}$, and input $\ddot{s}$. The loss function is the traversal time (2.7). The constraints are the dynamic constraints

$$\begin{aligned} \frac{ds}{dt} &= \dot{s} \\ \frac{d\dot{s}}{dt} &= \ddot{s} \end{aligned} \tag{2.12}$$

and constraints on the input $\ddot{s}$. The constraints on $\ddot{s}$ are obtained by combining (2.2) and (2.11), which gives

$$\tau_i^{min} \leq \tau_i = m_i \alpha_i \ddot{s} \leq \tau_i^{max}, \quad i = 1,2 \tag{2.13}$$

We also have boundary conditions on s and $\dot{s}$, given by

$$\begin{aligned} s(t_0) &= s_0, \quad s(t_f) = s_f \\ \dot{s}(t_0) &= 0, \quad \dot{s}(t_f) = 0 \end{aligned} \tag{2.14}$$

where the boundary conditions on $\dot{s}$ require that the robot should start and finish the motion with zero velocity.

Numerical solutions to this optimal control problem are presented below. For the chosen example, it is also possible to derive an analytical solution, which is presented for comparison.

Analytical Solution

The path acceleration $\ddot{s}$ is constrained by (2.13). These constraints are in this example especially simple, since they can be replaced by constant limits on $\ddot{s}$. The limits are obtained by requiring that $\ddot{s}$ should be chosen so that both torques in (2.13) are inside the torque limits. Assume that α_1 and α_2 in (2.13) are both positive. The limits on $\ddot{s}$ then become

$$\ddot{s}_{max} = \min_{i=1,2} \frac{\tau_i^{max}}{m_i \alpha_i} \tag{2.15a}$$

$$\ddot{s}_{min} = \max_{i=1,2} \frac{\tau_i^{min}}{m_i \alpha_i} \tag{2.15b}$$

The minimum time optimization for the robot (2.1) along the path (2.3) thus becomes a minimum time problem for the double integrator (2.12) with constant input limits. The solution to this optimal control problem is bang-bang with one switch, e.g. [Bryson and Ho, 1975]. The solution has the form

$$\ddot{s} = \begin{cases} \ddot{s}_{max}, & t_0 \leq t \leq t_1 \\ \ddot{s}_{min}, & t_1 \leq t \leq t_f \end{cases} \tag{2.16}$$

where the switching time is denoted t_1. This gives the path parameter

$$s(t) = \begin{cases} \ddot{s}_{max}\frac{t^2}{2} + s_0, & 0 \leq t \leq t_1 \\ \ddot{s}_{min}\frac{(t-t_1)^2}{2} + \dot{s}_1(t - t_1) + s_1, & t_1 \leq t \leq t_f \end{cases} \tag{2.17}$$

where the velocity at the switching time t_1 is denoted $\dot{s}_1$. The torque limits in (2.2) and the parameters in (2.1) and (2.3) are chosen as

$$\begin{aligned} &\tau_i^{min} = -1, \quad \tau_i^{max} = 1, \quad m_i = 1, \quad i = 1,2 \\ &\alpha_1 = 2, \quad \alpha_2 = 1, \quad s_0 = 0, \quad s_f = 1 \end{aligned} \tag{2.18}$$

This gives, using (2.15a) and (2.15b),

$$\ddot{s}_{max} = \min(\frac{1}{2}, 1) = \frac{1}{2}, \quad \ddot{s}_{min} = \max(-\frac{1}{2}, -1) = -\frac{1}{2} \tag{2.19}$$

Using the boundary conditions (2.14), the variables t_1,s_1, $\dot{s}_1$, and t_f in (2.17) are determined as

$$t_1 = \sqrt{2}, \quad s_1 = \frac{1}{2}, \quad \dot{s}_1 = \frac{1}{\sqrt{2}}, \quad t_f = 2\sqrt{2}$$

This gives the time optimal path parameter

$$s(t) = \begin{cases} \frac{1}{4}t^2, & 0 \le t \le \sqrt{2} \\ -\frac{1}{4}(t-\sqrt{2})^2 + \frac{1}{\sqrt{2}}(t-\sqrt{2}) + \frac{1}{2}, & \sqrt{2} \le t \le 2\sqrt{2} \end{cases} \tag{2.20}$$

Velocity and acceleration profiles The above solution (2.20) can of course be represented by the corresponding velocity and acceleration profiles (2.4) and (2.5). These profiles could be computed by differentiating $s(t)$ in (2.20) to obtain $\dot{s}(t)$ and $\ddot{s}(t)$, and then express $\dot{s}(t)$ and $\ddot{s}(t)$ as functions of $s(t)$. However, $v_1(s)$ and $v_2(s)$ are instead calculated directly using (2.4), (2.5) and (2.6). The acceleration profile $v_2(s)$ is given, using (2.16), by

$$v_2(s) = \begin{cases} \ddot{s}_{max}, & s_0 \le s \le s_1 \\ \ddot{s}_{min}, & s_1 \le s \le s_f \end{cases}$$

The relation (2.6) now gives

$$\frac{1}{2}v_1(s)^2 = \begin{cases} \ddot{s}_{max}s + s_0, & s_0 \le s \le s_1 \\ \ddot{s}_{min}(s - s_1) + \ddot{s}_{max}s_1 + s_0, & s_1 \le s \le s_f \end{cases} \tag{2.21}$$

Using (2.18) and (2.19), (2.21) becomes

$$\frac{1}{2}v_1(s)^2 = \begin{cases} \frac{1}{2}s, & 0 \le s \le s_1 \\ -\frac{1}{2}(s - s_1) + \frac{1}{2}s_1, & s_1 \le s \le 1 \end{cases} \tag{2.22}$$

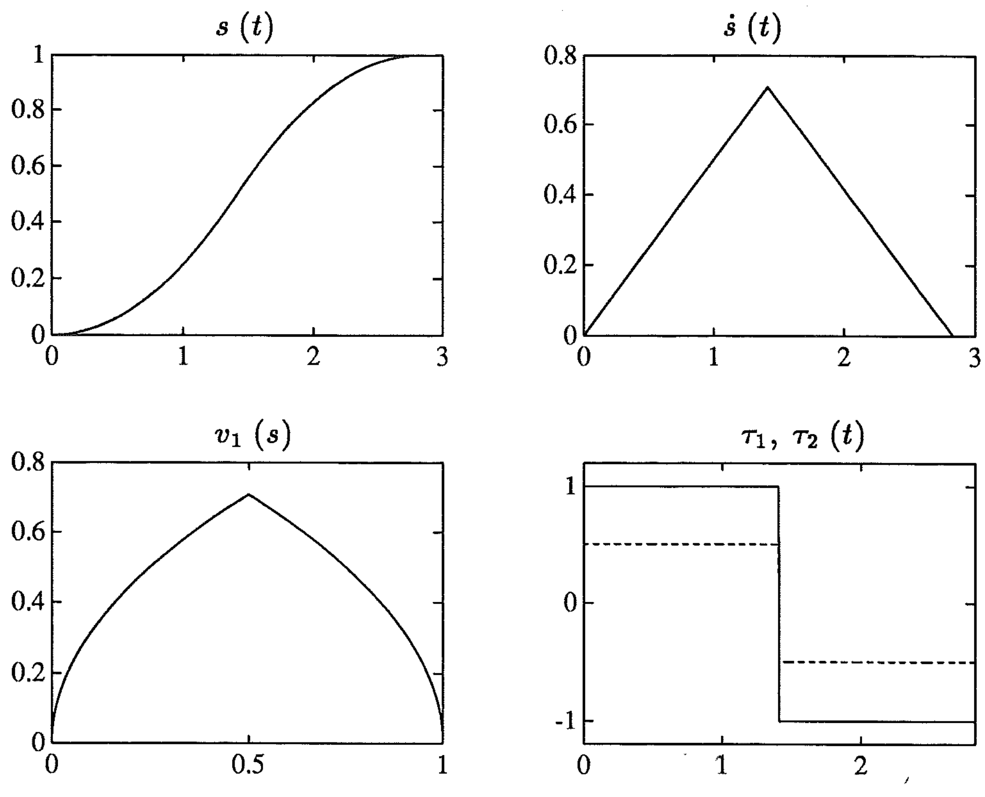


Figure 2.1 The analytical solution for the example.

The boundary condition $\dot{s}(t_f) = 0$ in (2.14), i.e. $v_1(s_f) = v_1(1) = 0$, now gives $s_1 = \frac{1}{2}$. This gives the velocity and acceleration profiles

$$\begin{aligned} v_1(s) &= \begin{cases} \sqrt{s}, & 0 \le s \le \frac{1}{2} \\ \sqrt{1-s}, & \frac{1}{2} \le s \le 1 \end{cases} \\ v_2(s) &= \begin{cases} \frac{1}{2}, & 0 \le s \le \frac{1}{2} \\ -\frac{1}{2}, & \frac{1}{2} \le s \le 1 \end{cases} \end{aligned} \tag{2.23}$$

The solution is shown in Figure 2.1, both as a time function $s(t)$, and as a velocity profile v_1. The upper left plot shows the path parameter $s(t)$ in (2.20). The upper right plot shows the path velocity $\dot{s}(t)$. The lower left plot shows the velocity profile $v_1(s)$ in (2.23). The lower right plot shows the torques $\tau_1(t)$, solid line, and $\tau_2(t)$, dashed line. The torques are computed from the path acceleration $\ddot{s}(t)$, using (2.11).

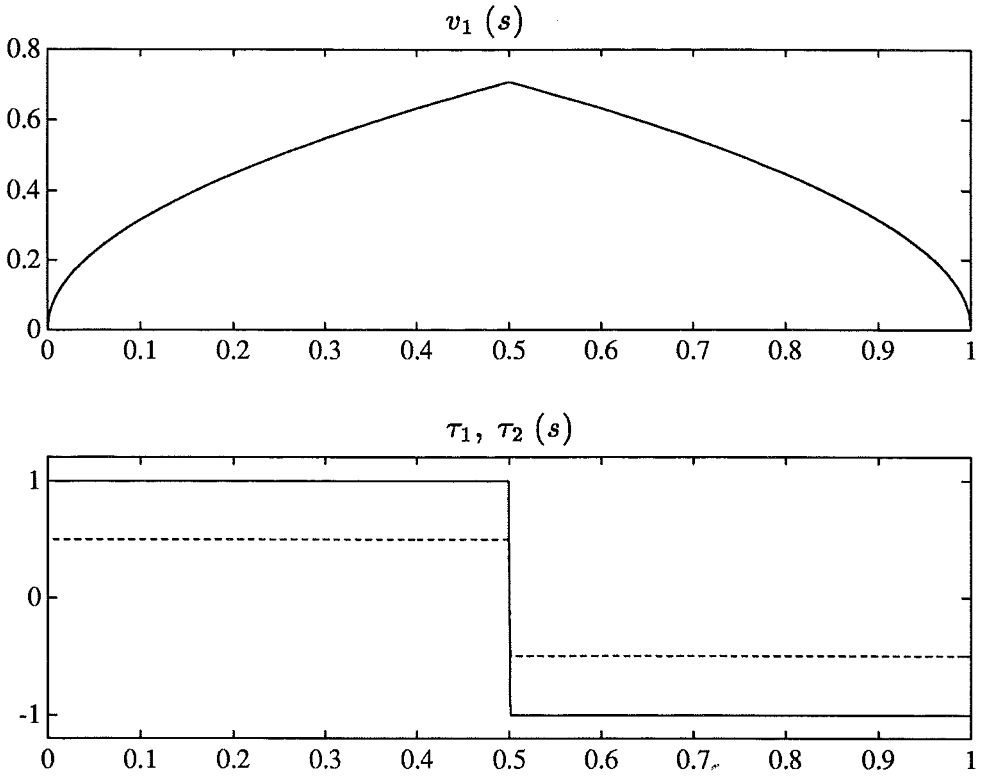


Figure 2.2 The phase plane solution for the example. The upper plot shows the optimal velocity profile. The lower plot shows the torques τ_1, solid line, and τ_2, dashed line, as functions of the path parameter s.

Phase Plane Optimization

For the example used here, it was possible, due to the constant limits on $\ddot{s}$, (2.15), to derive an analytical minimum time solution. In general, due to the robot model and/or the path description being more complicated, the limits on $\ddot{s}$ depend on s and $\dot{s}$, and it is not possible to derive an analytical solution.

For rigid robots, the minimum time optimization along a predefined path can be solved using phase-plane techniques [Bobrow *et al.*, 1985; Pfeiffer and R.Johanni, 1986; Shin and N.D.McKay, 1985]. The time optimal solution is computed in the phase-plane, i.e. in the s-$\dot{s}$-plane. The result is a velocity profile $v_1(s)$, i.e. a curve $\dot{s}$ as a function of s. The curve consists of segments where for each segment, the path acceleration $\ddot{s}$ is typically maximum or minimum. The result of using the phase-plane method on this example is shown in Figure 2.2. The upper plot shows the velocity profile $v_1(s)$. The velocity profile was obtained by numerical integration of (2.12) using maximum and minimum path acceleration (2.15). The lower plot shows the corresponding torques, computed from (2.11).

Parametric Optimization

The phase plane method is an efficient numerical method for minimum time optimization for rigid robots. It can however not be applied to higher order systems, e.g. flexible joint robots. An intuitive reason is that the phase-plane, i.e. the s-$\dot{s}$-plane is then not a sufficient state representation, because higher order derivatives are needed. Another approach is to use polynomial approximation to obtain a parametric representation of the velocity profile. The optimization can then be done using parametric optimization. An optimization method using this technique is presented for the flexible joint case in Chapter 3. As an illustration, an example of using the method for the rigid case is given. The method is based on polynomial approximation of the function

$$y(s) = \frac{v_1(s)^2}{2} \tag{2.24}$$

Note that direct approximation of the velocity profile $v_1(s)$ would lead to difficulties. This is seen as follows. Assume that $v_1(s)$ is approximated by a piecewise polynomial function. The boundary condition $\dot{s}(t_0) = 0$ in (2.14) gives, using (2.4), that $v_1(s_0) = 0$. It then follows from (2.6), since $v_1(s)$ is polynomial, that $v_2(s_0) = 0$, i.e. $\ddot{s}(t_0) = 0$, which gives $\tau(t_0) = 0$ by (2.11). A consequence of approximating $v_1(s)$ by a piecewise polynomial function would thus be that the initial value of the torque τ is zero, which of course makes it impossible to obtain a good approximation of the optimal velocity profile, where $\tau(t_0) \neq 0$, see e.g. Figure 2.1.

If instead the function $y(s)$ in (2.24) is approximated and the boundary conditions for $y(s)$ at $s = s_0$ are chosen as

$$y(s_0) = 0, \qquad \frac{dy(s_0)}{ds} \neq 0$$

we get, using (2.24), $v_1(s_0) = \dot{s}(t_0) = 0$, and, using (2.6), $v_2(s_0) = \ddot{s}(t_0) \neq 0$, i.e. $\tau(t_0) \neq 0$.

An example of an optimization where $y(s)$ in (2.24) is approximated by a piecewise polynomial function with third degree polynomial pieces is shown in Figure 2.3. The number of third degree polynomial pieces was 40, and the s-vector was discretized in 223 points. A comparison with the phase-plane solution in Figure 2.2 shows that the switch in the torques is somewhat smoothened. Nevertheless, for this example, the loss in traversal time due to polynomial approximation is small. Computation of the traversal time for the parametric solution in Figure 2.3 assuming constant path acceleration $\ddot{s}$ between the discretization points in s gives $t_f = 2.83$ seconds, which, up to two decimal places, is the same as the analytic traversal time $t_f = 2\sqrt{2} \approx 2.83$ seconds.

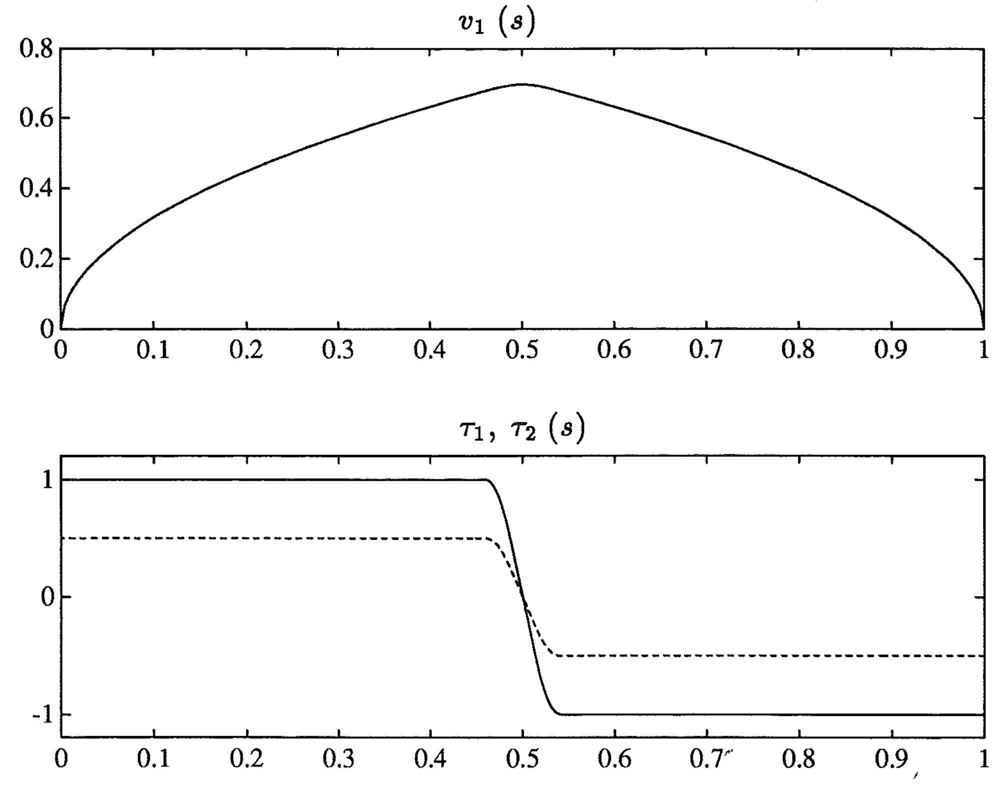


Figure 2.3 The parametric solution for the example. The upper plot shows the velocity profile. The lower plot shows the corresponding torques.

2.2 Using the nominal path parameter

The time optimal path parameter $s(t)$ in (2.20) is used to form a reference trajectory for a robot controller. The sensitivity to modeling errors is demonstrated by simulations.

The reference trajectory is denoted $q_r(t)$ and is computed as $q_r(t) = f(s(t))$, where $f(s)$ is the path description, given in (2.3). The reference trajectory $q_r(t) = f(s(t))$, and its first and second time derivatives are, using the chain rule as in (2.8), given by

$$
\begin{aligned}
q_r(t) &= f(s(t)) \\
\dot{q}_r(t) &= f'(s(t))\dot{s}(t) \\
\ddot{q}_r(t) &= f''(s(t))\dot{s}(t)^2 + f'(s(t))\ddot{s}(t)
\end{aligned}
\tag{2.25}
$$

The robot controller is a computed torque controller [Asada and Slotine, 1986] for the model (2.1). The controller is given by

$$
\tau = \hat{M}(\ddot{q}_r + K_v(\dot{q}_r - \dot{q}) + K_p(q_r - q)) \tag{2.26}
$$

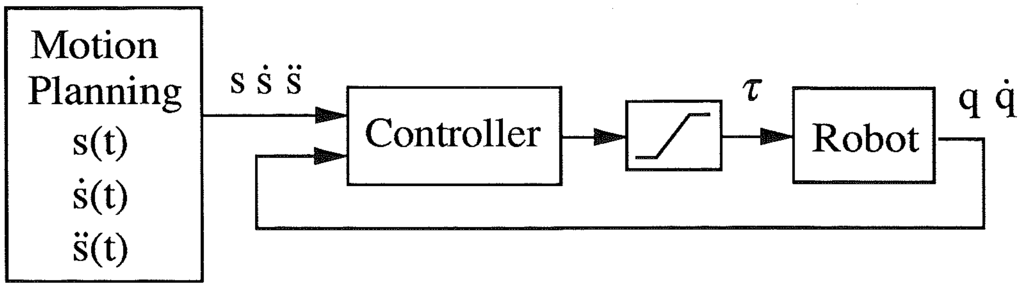


Figure 2.4 The system structure when the nominal path parameter is used directly. The block "motion planning" generates or reads out stored values of the time functions $s(t)$, $\dot{s}(t)$, and $\ddot{s}(t)$.

where $\hat{M}$ is an estimated mass matrix, and K_v and K_p are feedback matrices. If $\hat{M} = M$, combining (2.1) and (2.26) gives, introducing the tracking error $e = q_r - q$, that

$$\ddot{e} + K_v\dot{e} + K_p e = 0 \tag{2.27}$$

which gives asymptotic tracking of any reference trajectory.

The reference inputs to the robot controller (2.26) are $q_r(t)$, $\dot{q}_r(t)$, and $\ddot{q}_r(t)$. It follows from from (2.25) that the robot controller (2.26) can be regarded as having three scalar inputs, s, $\dot{s}$, and $\ddot{s}$.

The system structure is shown in Figure 2.4. The inputs to the robot controller are the nominal time functions $s(t)$, $\dot{s}(t)$, and $\ddot{s}(t)$. The torque constraints are represented by a saturation block, which limits the output of the robot controller.

Simulations

Simulations of the system in Figure 2.4 are presented. The ideal case with a perfect model is first discussed. A simulation where the controller (2.26) is used on a perfect model, i.e. (2.1) with $m_1 = m_2 = 1$, is shown in Figure 2.5. The model parameters were chosen as in (2.18) and the feedback matrices K_v and K_p in (2.26) were chosen as diagonal matrices with diagonal elements 20 and 100, respectively. The lower left plot in Figure 2.5 shows the path parameter $s(t)$ in (2.20) together with s_f, which represents the end point of the path. The mid left plot shows the velocity profile $v_1(s)$ in (2.23). The path parameter and the velocity profile are the same as in Figure 2.1, the upper left and the lower left plots. The reference trajectory $q_{r_i}(t) = f_i(s(t))$, $i = 1, 2$, is shown in the upper right plot. The resulting torque outputs from the controller (2.26) are shown in lower right plot. Due to the ideal situation these torques coincide with the nominal torques, shown in Figure 2.1, the lower right plot. The actual trajectories $q_1(t)$ and $q_2(t)$ are shown in the upper right plot. This plot shows that the tracking is perfect, as is expected since the simulation is done for an ideal case. The mid right plot shows that

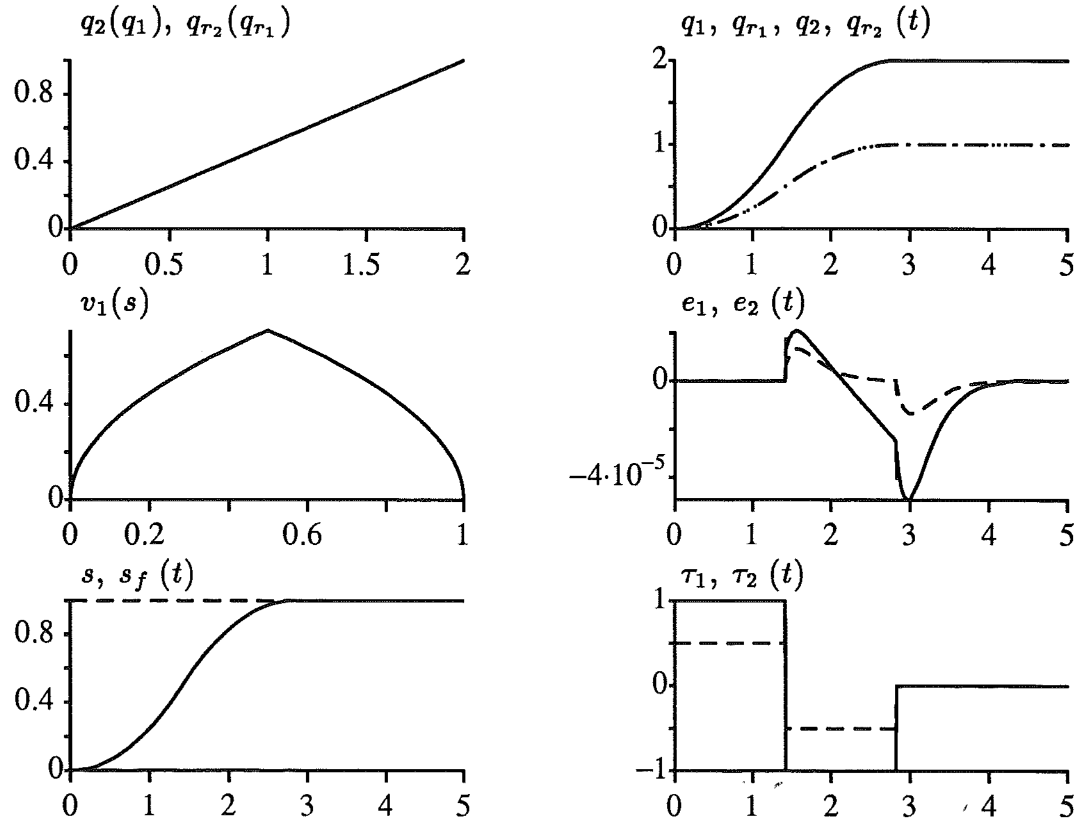


Figure 2.5 A simulation where the nominal time optimal path parameter is used in the reference trajectory. The model is perfect and the path following is good.

there are however small tracking errors $e_i = q_{r_i} - q_i$, $i = 1, 2$. These errors are due to errors in the numerical integration. The simulation was done with the initial conditions $e(0) = \dot{e}(0) = 0$, i.e. from (2.27), the correct solution has $e(t) \equiv 0$. The upper left plot shows the desired path (2.3) and the actual path, obtained by plotting $q_2(t)$ as a function of $q_1(t)$. As can be seen, the path following is perfect.

Introducing model errors The sensitivity to modeling errors is illustrated by using a robot model with increased mass and viscous friction. Equation (2.1) is thus modified to

$$m_i\ddot{q}_i + d_i\dot{q}_i = \tau_i, \quad i = 1, 2 \tag{2.28}$$

where the parameters are chosen as

$$m_1 = 1.1, \quad m_2 = 1, \quad d_1 = 0.1, \quad d_2 = 0 \tag{2.29}$$

A simulation where the robot model (2.28) is used, and where the robot

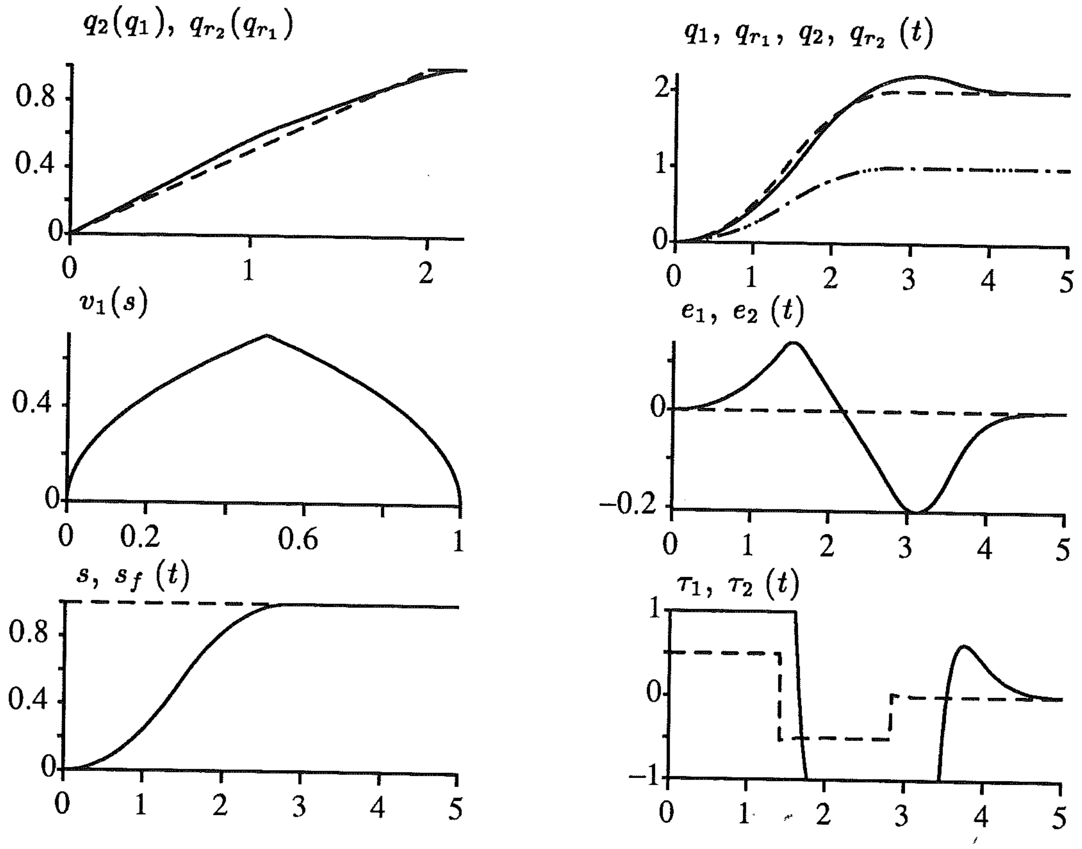


Figure 2.6 A simulation where model errors are introduced. As can be seen from the upper left plot, the path cannot be followed.

controller and the time optimal path parameter $s(t)$ are unchanged is shown in Figure 2.6.

The lower left plot and the mid left plot show the path parameter $s(t)$ and the velocity profile $v_1(s)$. These plots are the same as in the previous simulation, shown in Figure 2.5. Thus, the same reference trajectory $q_{r_i}(t) = f_i(s(t))$, $i = 1, 2$, as in the previous simulation is used. However, since the path parameter $s(t)$ and the robot controller are based on the, now erroneous, model (2.1), the situation is no longer ideal. This is seen in the upper right plot which shows the reference trajectories and the actual trajectories. The dashed line in this plot is the reference trajectory $q_{r_1}(t)$. The solid line is the actual trajectory $q_1(t)$. As can be seen, the actual trajectory cannot track the reference trajectory. Consider the situation around $t = 1.5$ seconds. The actual trajectory $q_1(t)$ is then behind the reference trajectory $q_{r_1}(t)$, i.e. the tracking error $e_1(t) = q_{r_1}(t) - q_1(t)$ is positive. This is seen in the mid right plot. The resulting torques are shown in the lower right plot. The nominal switching time is $\sqrt{2} \approx 1.41$ seconds, see (2.20). However, the large tracking error in joint one has the effect that the torque τ_1, solid line in the lower left plot, remains at the limit $\tau_1^{max} = 1$ also after the switch. A

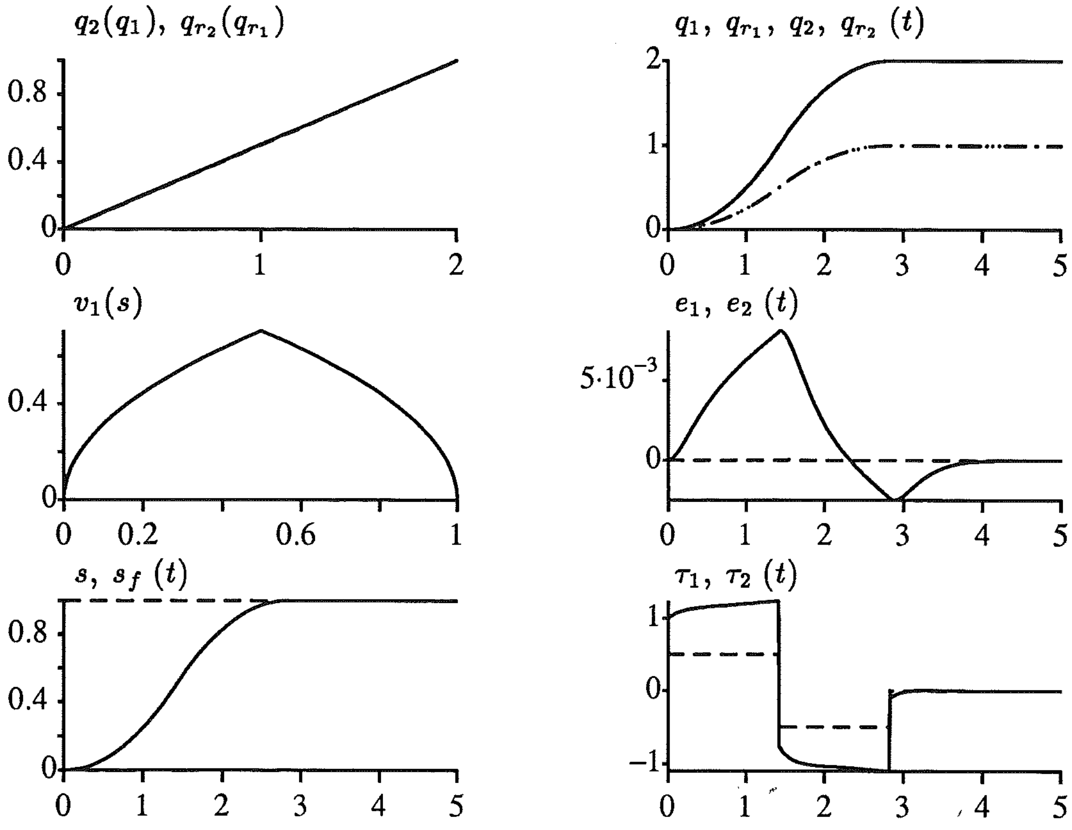


Figure 2.7 A simulation of the perturbed model, when the torque limits are removed. Note that the torque τ_1 required to track the nominal trajectory $q_{r_1}(t)$ is outside the limits ± 1.

similar phenomenon is seen in the final switch, where τ_1 remains at the limit $\tau_i^{min} = -1$ longer than τ_2. This is a result of a large negative tracking error, as can be seen in the mid right plot. This error is also seen in the upper right plot, as an overshoot in the actual trajectory q_1.

As a consequence, the path cannot be followed. This is seen in the upper left plot which shows the desired path (2.3), dashed line, and the actual path, solid line.

The simulation example in Figure 2.6 shows the degradation in performance that occurs when the model is not perfect. Note however that the modeling error used here is not large from the robot controller perspective. This is seen by simulating the same system as in Figure 2.6, but without the torque limits. The result of this simulation is shown in Figure 2.7. As can be seen from this figure, the path following is good when the torque limits are removed. Note that the torque τ_1, the solid line in the lower right plot, is outside the torque limits ± 1.

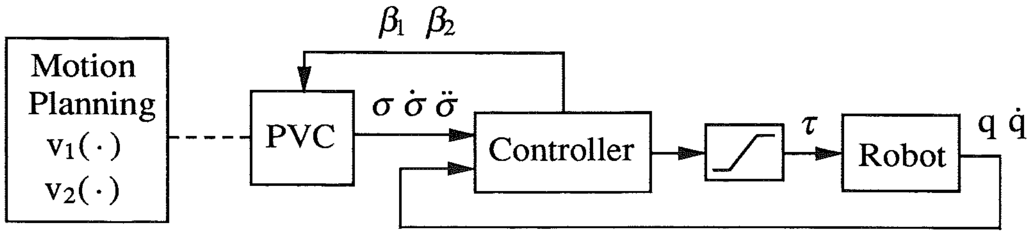

β_1 β_2
Motion Planning $v_1(\cdot)$ $v_2(\cdot)$
PVC
$\sigma\ \dot\sigma\ \ddot\sigma$
Controller
τ
Robot
q $\dot{q}$

Figure 2.8 The system structure when the path velocity controller is used. The feedback signals from the robot controller are β_1 and β_2.

2.3 Path Velocity Control

As was shown in Figure 2.6, using the nominal path parameter $s(t)$ in (2.20) resulted in path deviations, due to the limited torques. This section demonstrates in a simulation how a path velocity controller can be used to modify this nominal path parameter to a path parameter $\sigma(t)$ which results in good path following. A detailed presentation of path velocity control is given in Chapter 4.

The path parameter $\sigma(t)$ is used by the robot controller in the reference trajectory as $q_r(t) = f(\sigma(t))$. The reference trajectory and its time derivatives are thus, using (2.25) with $s(t)$ replaced by $\sigma(t)$

$$\begin{aligned} q_r(t) &= f(\sigma(t)) \\ \dot{q}_r(t) &= f'(\sigma(t))\dot\sigma(t) \\ \ddot{q}_r(t) &= f''(\sigma(t))\dot\sigma(t)^2 + f'(\sigma(t))\ddot\sigma(t) \end{aligned} \tag{2.30}$$

i.e. the reference inputs to the robot controller are now $\sigma(t)$, $\dot\sigma(t)$, and $\ddot\sigma(t)$. The system structure when using path velocity control is shown in Figure 2.8. The path velocity controller uses feedback from the robot controller. The feedback signals are denoted β_1 and β_2. A nominal velocity profile, given by the function v_1 in (2.4), and a nominal acceleration profile, given by the function v_2 in (2.5), are also used. The block "Motion planning" represents the profiles v_1 and v_2. The dashed line between this block and the path velocity controller means that these functions are in fact internal to the path velocity controller. They are used in the path velocity controller as nonlinear functions $v_1(\sigma)$ and $v_2(\sigma)$, i.e. they give the nominal velocity and acceleration as functions of the position σ. As can be seen in Figure 2.8, the path velocity controller introduces an additional feedback loop in the system. The path velocity controller uses feedback to modify the reference trajectory for the robot controller. Compare with Figure 2.4.

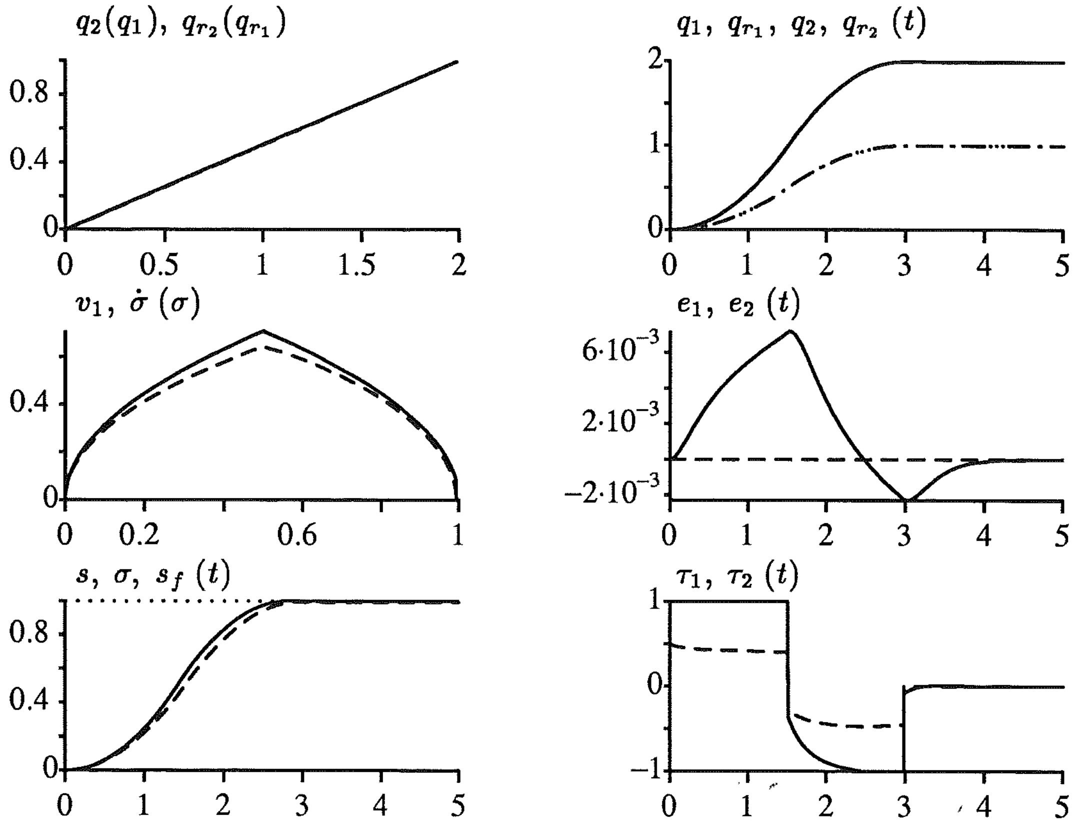


Figure 2.9 A simulation where the path velocity controller is used for modification of the nominal minimum time trajectory.

Simulation

A simulation where a path velocity controller is used as in Figure 2.8 and where the robot model is given by (2.28) and (2.29) is shown in Figure 2.9. The reference trajectory $q_r(t) = f(\sigma(t))$ is shown, together with the actual trajectories, in the upper right plot. As can be seen in this plot, the tracking is good. Note that the robot model used in this simulation is the same as in Figure 2.7, where the path following was unsatisfactory. The robot controller (2.26) is also unchanged. The reason for having good tracking in this simulation is that the reference trajectory is different from the reference trajectory in Figure 2.7. The reference trajectory is now $q_r(t) = f(\sigma(t))$ where $\sigma(t)$ is generated by the path velocity controller. The reference trajectory used in the previous simulations was $q_r(t) = f(s(t))$ where $s(t)$ is the nominal time optimal path parameter (2.20).

A comparison can be seen in the mid left plot. The solid line is the nominal velocity profile v_1 in (2.23). The dashed line is the actual velocity profile, i.e. $\dot\sigma$ as a function of σ. As can be seen, the actual velocity profile is lower than the nominal velocity profile. The result of using the path velocity controller is thus that the reference trajectory is modified, via feedback, such that the path velocity is reduced. Compared to the simulation in Figure 2.6, see the upper right plot around $t = 1.5$ seconds, where the actual trajectory was behind the reference trajectory, the reference trajectory is now adjusted to cope with the actual velocity obtained using maximum torque. This has the consequence that the desired path (2.3) can now be followed. This is seen in the upper left plot, which shows the desired and the actual paths.

The lower left plot shows the nominal path parameter $s(t)$, solid line, and the actual path parameter $\sigma(t)$, dashed line. As can be seen, the actual path parameter is delayed, resulting in larger traversal time. The traversal time t_f was computed as the time when the reference trajectory reached the end of the path, i.e. $\sigma(t_f) = s_f$. The traversal time was $t_f = 3$ seconds. The nominal traversal time is given in (2.20) as $2\sqrt{2} \approx 2.83$ seconds. Computation of the optimal traversal time for the perturbed model (2.28), using phase-plane optimization for computation of the minimum time velocity profile, gives the traversal time 2.97 seconds, i.e. the traversal time 3 seconds when the path velocity controller is used agrees well with the optimum. The motion is however not optimal, as can be seen in the lower right plot in Figure 2.9, where there is a time interval, approximately between 1.5 and 2.3 seconds, where none of the torques are at the limit. The mid right plot in Figure 2.9 shows the tracking errors. These errors are of the same magnitude as the tracking errors shown in Figure 2.7, where there are no torque limits. Hence, these errors are a result of decreased control performance for the robot controller, due to the model errors, and are not caused by the path velocity controller.

Using the path velocity controller gives a synchronization effect between the two joints. This is seen by comparing the first torque switch in the lower right plots in Figures 2.6 and 2.9. The lower right plot in Figure 2.9 shows that the torques now switch simultaneously. Note also that the switch occurs later in Figure 2.9, compared to the nominal switch, shown in Figure 2.5.

The path velocity controller will not be further evaluated here. A detailed discussion is deferred to Chapter 4, where specific algorithms for path velocity control are presented. Let it suffice to mention that the path velocity controller seems to work well in the simple example discussed here.

3

Path Velocity Planning

A nominal velocity profile can be obtained using minimum time optimization. Examples of such optimizations were given in Section 2.1. The robot model was chosen as a decoupled linear model (2.1), and the path was a straight line segment (2.3). This chapter extends the presentation to account for more complicated robot models and paths. A review of phase-plane optimization for rigid robots [Bobrow *et al.*, 1985; Shin and N.D.McKay, 1985; Pfeiffer and R.Johanni, 1986; Shiller and Lu, 1990] and a new method for flexible joint robots are presented. The robot models considered are a rigid body model [Asada and Slotine, 1986] and a model with joint flexibility [Spong, 1987], which is a natural and practically important first extension of the rigid model. The optimization methods are derived from a common optimal control formulation, which is presented in Section 3.1. The optimal control problem is of order pn, where n is the number of joints, and where $p = 2$ for rigid robots and $p = 4$ for flexible joint robots. In Section 3.2, the optimal control problem is reformulated, as was done in Section 2.1, by substituting the path constraint $q = f(s)$ into the robot dynamics, i.e. (2.9) is generalized. The result of the reformulation is an optimal control problem of order p. The dynamic system is a chain of p integrators, constrained by state dependent input constraints. These constraints are a generalization of (2.13).

An optimality result on the structure of the minimum time solution for the optimal control problem of order p is presented in Section 3.4. The result shows that for conditions that are typically satisfied for most parts of a path, the time optimal solution is bang-bang in the sense that the p-th order derivative of the path parameter s should either be maximized or minimized.

Section 3.5 presents an optimal control problem of order $p - 1$. This optimal control problem is obtained from the optimal control problem of order p in Section 3.2 by interpreting the path parameter s as the independent variable. A parametric optimization method for flexible joint robots is presented in Section 3.6. The method is an approximation of the optimal control problem of order $p-1$. A function of the robot velocity along the path is approximated by a piecewise polynomial function. This was also done in the subsection on parametric optimization in Section 2.1. The method has been implemented for the minimum time case. It is illustrated by numerical examples in Section 3.7.

3.1 Problem Formulation

An optimal control formulation of robot motion along a predefined path is presented.

Robot Models

The optimal control problem is formulated for a dynamic model where the input torque τ is given as a function of the joint variables q and its time derivatives up to order p. We will concentrate on two specific robot models, where $p = 2$ for the rigid model, and $p = 4$ for the flexible joint model.

Rigid body model The rigid model is given by

$$\tau = H(q)\ddot{q} + v(q, \dot{q}) + d(q)\dot{q} + g(q) \tag{3.1}$$

where $q \in \mathbb{R}^n$ is the vector of joint variables, $\tau \in \mathbb{R}^n$ is the vector of input torques, $H(q)$ is the inertia matrix, $v(q, \dot{q})$ is the vector of coriolis and centrifugal forces, $d(q)$ is the viscous friction matrix, and $g(q)$ is the vector of gravitational forces, all with obvious dimensions [Asada and Slotine, 1986].

Flexible joint model The flexible joint model is given by

$$\begin{aligned} &H(q)\ddot{q} + v(q, \dot{q}) + d(q)\dot{q} + g(q) + K(q - \theta) = 0 \\ &J\ddot{\theta} = \tau + K(q - \theta) \end{aligned} \tag{3.2}$$

which is the simplified model found e.g. in [Spong, 1987; Spong, 1990]. The vector of joint angles is denoted q, the actuator angles are denoted θ, and τ is the input torque. The joint stiffness matrix is denoted K, and J is the actuator inertia matrix. The variables $H(q)$, $v(q, \dot{q})$, $d(q)$, and $g(q)$ are defined as in (3.1). The model (3.2) is simplified from a full model on the form [Spong, 1990]

$$\begin{aligned}\begin{pmatrix} 0 \\ \tau \end{pmatrix} = \begin{pmatrix} M_{11}(q) & M_{12}(q) \\ M_{21}(q) & M_{22} \end{pmatrix} \begin{pmatrix} \ddot{q} \\ \ddot{\theta} \end{pmatrix} \\ + \begin{pmatrix} v_1(q,\dot{q},\dot{\theta}) \\ v_2(q,\dot{q},\dot{\theta}) \end{pmatrix} + \begin{pmatrix} g(q) \\ 0 \end{pmatrix} + \begin{pmatrix} K(q-\theta) \\ -K(q-\theta) \end{pmatrix}\end{aligned} \tag{3.3}$$

where the modeling assumptions done in the simplification are that the kinetic energy of the actuators is purely rotational, and that the actuator inertia is symmetric about the axis of rotation [Spong, 1987].

Common form for the robot dynamics For the models (3.1) and (3.2), the torque τ can be written as a function of q and its time derivatives as

$$\tau = h(q, \dot{q}, \ldots, q^{(p)}) \tag{3.4}$$

The rigid body dynamics (3.1) are already on this form with $p = 2$. The flexible joint model (3.2), can be transformed to the form (3.4) by solving for θ in the first equation in (3.2) to obtain

$$\theta = K^{-1}(H(q)\ddot{q} + v(q,\dot{q}) + d(q)\dot{q} + g(q)) + q \tag{3.5}$$

which then can be substituted into the second equation in (3.2) to obtain an expression on the form (3.4) with $p = 4$. The expression becomes

$$\begin{aligned}\tau = JK^{-1}\frac{d^2}{dt^2}(H(q)\ddot{q} + v(q,\dot{q}) + d(q)\dot{q} + g(q)) \\ + (H(q) + J)\ddot{q} + v(q,\dot{q}) + d(q)\dot{q} + g(q)\end{aligned} \tag{3.6}$$

For the full model (3.3), it is not possible in general to express the torque as in (3.4).

Torque Constraints

The torque constraints are given by limiting each joint torque, i.e.

$$\tau_i^{min} \leq \tau_i \leq \tau_i^{max}, \quad 1 \leq i \leq n \tag{3.7}$$

where n is the number of joints. We use the notation $\tau \in E$ as a shorthand notation for the torque constraints (3.7).

Path Description

The path is parametrized in joint space by a vector function $f(s) \in \mathbb{R}^n$ of the scalar path parameter $s \in \mathbb{R}$, $s_0 \leq s \leq s_f$, where $f(s_0)$ is the starting point and $f(s_f)$ is the end point of the path. We assume that the path is parametrized so that

$$f'(s) \neq 0, \quad s_0 \leq s \leq s_f \tag{3.8}$$

where 0 means the zero vector, i.e. for all $s_0 \leq s \leq s_f$, at least one component of $f'(s)$ is nonzero.

Optimal Control Problem

An optimal control problem is formulated. The formulation is done for a general loss function. The loss function is specialized to traversal time in Sections 3.3 and 3.6, where specific optimization methods are presented. The general loss function is however kept in Sections 3.2 and 3.5, where reformulations of the optimal control problem are presented.

Assume that the loss function to be minimized is a function of the robot state $(q, \dot{q}, \ldots, q^{(p-1)})$, and the input τ. An optimal control problem is then formulated as: find the function $\tau(t)$ that solves

$$\min_{\tau} \int_{t_0}^{t_f} L(q, \dot{q}, \ldots, q^{(p-1)}, \tau) dt \tag{$L1$}$$

where t_0 is specified and t_f is free. The constraints are the robot dynamics, written in the form (3.4) as

$$\tau = h(q, \dot{q}, \ldots, q^{(p)}) \tag{$D1$}$$

the path constraint and the torque constraints, formulated as

$$q = f(s), \quad \tau \in E \tag{$C1$}$$

and the boundary conditions that the initial and final states

$$\begin{aligned} &q(t_0), \ldots q^{(p-1)}(t_0) \\ &q(t_f), \ldots q^{(p-1)}(t_f) \end{aligned} \tag{$B1$}$$

should be given. An example of solving this optimal control problem for the case $p = 2$ was given in Section 2.1, e.g. (L1) corresponds to (2.7), (D1) corresponds to (2.1), the path $f(s)$ is given in (2.3), and the torque constraints $\tau \in E$ are given in (2.2). For the general case, the optimal control problem (L1, D1, C1, B1) has pn states, state equality constraints $q = f(s)$, and input inequality constraints $\tau \in E$.

3.2 Reducing the State Dimension

The state dimension of the optimal control problem (L1,D1,C1,B1) can be reduced from pn to p. The reduction is done by rewriting the robot dynamics (D1) and the loss function in (L1), using the path constraint $q = f(s)$. The idea is the same as in Section 2.1, i.e. the path constraint $q = f(s)$ is differentiated and substituted into the robot dynamics. For the general formulation (L1,D1,C1,B1), the substitution is done also for the loss function in (L1).

Substitution of $q = f(s)$ and its time derivatives up to order p, computed using the chain rule as in (2.8), into (D1) and (L1) gives the the loss function in (L1) and the robot dynamics (D1) as functions of $s, \dot{s}, \ldots, s^{(p)}$. The resulting loss function is written as

$$\int_{t_0}^{t_f} L_s(s, \dot{s}, \ldots, s^{(p)})dt \tag{3.9}$$

and the robot dynamics are written as

$$\tau = h_s(s, \dot{s}, \ldots s^{(p)}) \tag{3.10}$$

where the functions obtained after the substitution are denoted h_s, and L_s. For the two models (3.1) and (3.2), a special form for the function h_s can be obtained, giving the torque τ as

$$\tau = b_1(s)s^{(p)} + b_2(s, \ldots, s^{(p-1)}) \tag{3.11}$$

Explicit expressions for the vectors b_1 and b_2 are given below in Lemma 3.1.

Reduced Optimal Control Problem

Introducing a p-integrator with states $s, \ldots s^{(p-1)}$, and input $s^{(p)}$, an optimal control problem of order p can now be given as

$$\min_{s^{(p)}} \int_{t_0}^{t_f} L_s(s, \dot{s}, \ldots, s^{(p)})dt \tag{L2}$$

where L_s is introduced in (3.9). The dynamic constraints are the p-integrator

$$\begin{aligned} \frac{ds}{dt} &= \dot{s} \\ \frac{d\dot{s}}{dt} &= \ddot{s} \\ &\vdots \\ \frac{ds^{(p-1)}}{dt} &= s^{(p)} \end{aligned} \tag{D2}$$

The constraints corresponding to (C1), i.e. the constraints on the variables in the dynamic system, are obtained from the rewritten robot dynamics (3.10) as

$$\begin{aligned} &\tau = h_s(s, \dot{s}, \ldots, s^{(p)}) \\ &\tau \in E \end{aligned} \tag{C2}$$

For the models (3.1) and (3.2), we can use (3.11) to write these constraints as

$$\begin{aligned} &\tau = b_1(s)s^{(p)} + b_2(s, \ldots, s^{(p-1)}) \\ &\tau \in E \end{aligned} \tag{C2'}$$

The boundary conditions now become that the initial and final states in the p-integrator, i.e.

$$\begin{aligned} &s(t_0), \ldots, s^{(p-1)}(t_0) \\ &s(t_f), \ldots, s^{(p-1)}(t_f) \end{aligned} \tag{B2}$$

should be given. A comparison with the reformulated optimal control problem in Section 2.1 shows that (L2,D2,C2',B2) corresponds to (2.7), (2.12), (2.13), and (2.14).

Rewriting the robot dynamics

We show that the expression (3.11) is valid for the two models (3.1) and (3.2). The result is obtained by straightforward substitution of the path constraint $q = f(s)$ into the robot dynamics. For rigid robots, the result is obtained, as in [Bobrow *et al.*, 1985; Shin and N.D.McKay, 1985; Pfeiffer and R.Johanni, 1986], by substitution into the model (3.1). For the flexible joint case, the result is obtained by substitution into the model (3.6).

LEMMA 3.1
For the two models (3.1) and (3.2), the torque can be written as (3.11).

Proof: The rigid and flexible cases are considered separately.

Rigid Robots Moving the robot (3.1) along the path $f(s)$ gives, as in (2.8)

$$\begin{aligned} q &= f(s) \\ \dot{q} &= f'(s)\dot{s} \\ \ddot{q} &= f''(s)\dot{s}^2 + f'(s)\ddot{s} \end{aligned} \tag{3.12}$$

This can be substituted into (3.1) to get

$$\tau = a_1(s)\ddot{s} + a_2(s)\dot{s}^2 + a_3(s)\dot{s} + a_4(s) \tag{3.13}$$

where

$$\begin{aligned} a_1(s) &= H(f(s))f'(s) \\ a_2(s) &= H(f(s))f''(s) + v(f(s), f'(s)) \\ a_3(s) &= d(f(s))f'(s) \\ a_4(s) &= g(f(s)) \end{aligned} \tag{3.14}$$

and where the expression for $a_2(s)$ is obtained by using the fact that the elements of the coriolis and centrifugal vector $v(q,\dot q)$ are of the form $v_i = \sum_{j,k} v_{ijk}(q)\dot q_j \dot q_k$ [Asada and Slotine, 1986]. We see that (3.13) can be written as (3.11) with $p = 2$, and

$$\begin{aligned} b_1(s) &= a_1(s) \\ b_2(s,\dot s) &= a_2(s)\dot s^2 + a_3(s)\dot s + a_4(s) \end{aligned} \tag{3.15}$$

Flexible Joint Robots For the flexible robot (3.2), the result is derived from (3.6). Introduce the "rigid" torque τ_1 as

$$\tau_1 = H(q)\ddot q + v(q,\dot q) + d(q)\dot q + g(q) \tag{3.16}$$

The expression (3.6) can then be written as

$$\tau = JK^{-1}\frac{d^2\tau_1}{dt^2} + J\ddot q + \tau_1 \tag{3.17}$$

Since (3.16) is on the same form as (3.1), τ_1 can be written as

$$\tau_1 = a_1(s)\ddot s + a_2(s)\dot s^2 + a_3(s)\dot s + a_4(s) \tag{3.18}$$

The final result is now obtained by expressing $\frac{d^2\tau_1}{dt^2}$ as a function of s and its time derivatives. Taking time derivatives of τ_1 gives

$$\begin{aligned} \frac{d\tau_1}{dt} =& a_1(s)s^{(3)} + (a_1'(s) + 2a_2(s))\dot s\ddot s + a_2'(s)\dot s^3 + a_3'(s)\dot s^2 + a_3(s)\ddot s + a_4'(s)\dot s \\ \frac{d^2\tau_1}{dt^2} =& a_1(s)s^{(4)} + 2(a_1'(s) + a_2(s))\dot s s^{(3)} + (a_1''(s) + 5a_2'(s))\dot s^2\ddot s \\ &+ (a_1'(s) + 2a_2(s))\ddot s^2 + a_2''(s)\dot s^4 + a_3(s)s^{(3)} + 3a_3'(s)\dot s\ddot s \\ &+ a_3''(s)\dot s^3 + a_4'(s)\ddot s + a_4''(s)\dot s^2 \end{aligned} \tag{3.19}$$

Substituting $\frac{d^2\tau_1}{dt^2}$ from (3.19), $\ddot q$ from (3.12), and τ_1 from (3.18) into (3.17) now gives that the torque τ can be written as

$$\tau = \alpha(s)^T\varphi(\dot s, \ddot s, s^{(3)}, s^{(4)}) \tag{3.20}$$

where

$$\varphi^T = \begin{pmatrix} s^{(4)} & \dot{s}s^{(3)} & \dot{s}^2\ddot{s} & \ddot{s}^2 & \dot{s}^4 & s^{(3)} & \dot{s}\ddot{s} & \dot{s}^3 & \ddot{s} & \dot{s}^2 & \dot{s} & 1 \end{pmatrix} \tag{3.21}$$

and

$$\alpha = \begin{pmatrix} (JK^{-1}a_1(s))^T \\ (JK^{-1}2(a_1'(s) + a_2(s)))^T \\ (JK^{-1}(a_1''(s) + 5a_2'(s)))^T \\ (JK^{-1}(a_1'(s) + 2a_2(s)))^T \\ (JK^{-1}a_2''(s))^T \\ (JK^{-1}a_3(s))^T \\ (JK^{-1}3a_3'(s))^T \\ (JK^{-1}a_3''(s))^T \\ (JK^{-1}a_4'(s) + Jf'(s) + a_1(s))^T \\ (JK^{-1}a_4''(s) + Jf''(s) + a_2(s))^T \\ a_3(s)^T \\ a_4(s)^T \end{pmatrix} \tag{3.22}$$

and we see from (3.20), (3.21), and (3.22) that the torque can be written as (3.11) with $p = 4$. □

3.3 Phase-plane Optimization

This section presents a review of phase-plane optimization [Bobrow *et al.*, 1985; Pfeiffer and R.Johanni, 1986; Shin and N.D.McKay, 1985; Shiller and Lu, 1990].

Optimal Control Problem

The optimal control problem is a specialization of (L2,D2,C2',B2) with $p = 2$. The problem is to minimize the traversal time

$$t_f = \int_0^{t_f} dt = \int_{s_0}^{s_f} \frac{dt}{ds} ds = \int_{s_0}^{s_f} \frac{1}{\dot{s}} ds \tag{L3}$$

subject to the dynamic constraints

$$\begin{aligned} \frac{ds}{dt} &= \dot{s} \\ \frac{d\dot{s}}{dt} &= \ddot{s} \end{aligned} \tag{D3}$$

and the state-dependent input constraints

$$\tau = b_1(s)\ddot{s} + b_2(s,\dot{s}) \in E \qquad (C3')$$

where b_1 and b_2 are defined in (3.15). We specialize the boundary conditions (B2) to

$$s(t_0) = s_0, \quad \dot{s}(t_0) = 0, \quad s(t_f) = s_f, \quad \dot{s}(t_f) = 0 \qquad (B3)$$

Phase-plane Optimization

From the expression (L3) for the traversal time, it is seen that minimizing traversal time can also be seen as finding a velocity profile $\dot{s} = v_1(s)$ with as high velocity as possible. In the minimum time example in Section 2.1, this was done by integrating (D3) forward from $s = s_0$, with maximum acceleration (2.15a), and integrating backward from s_f with minimum acceleration (2.15b), and then connecting the curves. The resulting velocity profile then has one switch from maximum to minimum acceleration, see Figure 2.2. For the general case, the time optimal velocity profile may have more than one switch. The reason for this is that the maximum and minimum values of $\ddot{s}$ generally depend on s and $\dot{s}$. This means that for a given s, there may exist path velocities $\dot{s}$ such that there is no acceleration $\ddot{s}$ that gives all torques inside the limits. A consequence of using the simple strategy that was used in Section 2.1 could then be that the forward integration cannot continue after a certain s-value, e.g. $s = s_1$, due to the path velocity $\dot{s}$ being too large, i.e. $\dot{s}$ is such that there is no $\ddot{s}$ that results in $\tau \in E$. Similarly, the backward integration may not be possible to continue for s less than a certain s-value, e.g. $s = s_2$. If $s_1 < s_2$, it is thus not possible to connect the curves. The algorithm for phase plane optimization takes this into account in order to find switching points, where the path acceleration $\ddot{s}$ switches from maximum to minimum acceleration or vice versa. The algorithm is presented below in Algorithm 3.1. The constraints are first presented.

Constraints on path acceleration The torque constraints (C3') give admissible values of $\ddot{s}$ which are defined as follows.

DEFINITION 3.1
For given values of s and $\dot{s}$, the admissible values of $\ddot{s}$ are those that result in $\tau \in E$. □

The admissible values of $\ddot{s}$ are computed by finding the maximum and minimum admissible $\ddot{s}$. The torque τ_i is constrained by

$$\tau_i^{min} \leq \tau_i = b_{1_i}(s)\ddot{s} + b_{2_i}(s,\dot{s}) \leq \tau_i^{max}, \quad 1 \leq i \leq n \qquad (3.23)$$

Each joint i now gives upper and lower bounds on $\ddot{s}$, denoted $\ddot{s}^i_{max}$ and $\ddot{s}^i_{min}$, and computed by

$$\ddot{s}^i_{max}(s,\dot{s}) = \begin{cases} (\tau_i^{max} - b_{2_i}(s,\dot{s}))/b_{1_i}(s), & b_{1_i}(s) > 0 \\ (\tau_i^{min} - b_{2_i}(s,\dot{s}))/b_{1_i}(s), & b_{1_i}(s) < 0 \\ \infty, & b_{1_i}(s) = 0 \end{cases} \tag{3.24}$$

and

$$\ddot{s}^i_{min}(s,\dot{s}) = \begin{cases} (\tau_i^{min} - b_{2_i}(s,\dot{s}))/b_{1_i}(s), & b_{1_i}(s) > 0 \\ (\tau_i^{max} - b_{2_i}(s,\dot{s}))/b_{1_i}(s), & b_{1_i}(s) < 0 \\ -\infty, & b_{1_i}(s) = 0 \end{cases} \tag{3.25}$$

The bounds on $\ddot{s}$ are then obtained by maximizing and minimizing over i

$$\ddot{s}_{max}(s,\dot{s}) = \min_i \ddot{s}^i_{max}(s,\dot{s}), \quad \ddot{s}_{min}(s,\dot{s}) = \max_i \ddot{s}^i_{min}(s,\dot{s}) \tag{3.26}$$

These bounds give an admissible interval $[\ddot{s}_{min}(s,\dot{s}), \ddot{s}_{max}(s,\dot{s})]$ for the path acceleration $\ddot{s}$. Further, the bounds are finite. This is seen from the path assumption (3.8), which, using (3.14), (3.15), and the fact that the inertia matrix $H(f(s))$ in (3.14) is positive definite, shows that all components of the vector $b_1(s)$ cannot be zero simultaneously. This shows that the bounds (3.24) and (3.25) are finite, and hence the bounds on $\ddot{s}$ (3.26) are finite.

Constraints on path velocity The bounds on $\ddot{s}$ (3.26) depend on $\dot{s}$. Thus, there may exist values of $\dot{s}$ such that there is no $\ddot{s}$ that gives $\tau \in E$. The admissible values of $\dot{s}$ are therefore defined as follows.

DEFINITION 3.2
For a given value of s, $\dot{s}$ is admissible if there exists admissible $\ddot{s}$, i.e. $\ddot{s}$ that gives $\tau \in E$.

For the example in Section 2.1, the bounds on $\ddot{s}$ (2.15) are independent of $\dot{s}$, i.e. all values of $\dot{s}$ are admissible.

Interpretation of the constraints The constraints on $\ddot{s}$ and $\dot{s}$ can be given a geometric interpretation. The torque vector τ is written as in (C3'), i.e.

$$\tau = b_1(s)\ddot{s} + b_2(s,\dot{s})$$

The admissible torques satisfy $\tau \in E$. Given s and $\dot{s}$, consider the line with a direction given by $b_1(s)$, and passing through the end point of the vector $b_2(s,\dot{s})$. The boundary cases for the admissible values of $\ddot{s}$ are then found as the intersections between this line and the region E. If there is no intersection, there are no values of $\ddot{s}$ resulting in admissible torques, i.e. the path velocity $\dot{s}$ is not admissible. Figure 3.1 illustrates these constraints for the two-joint case.

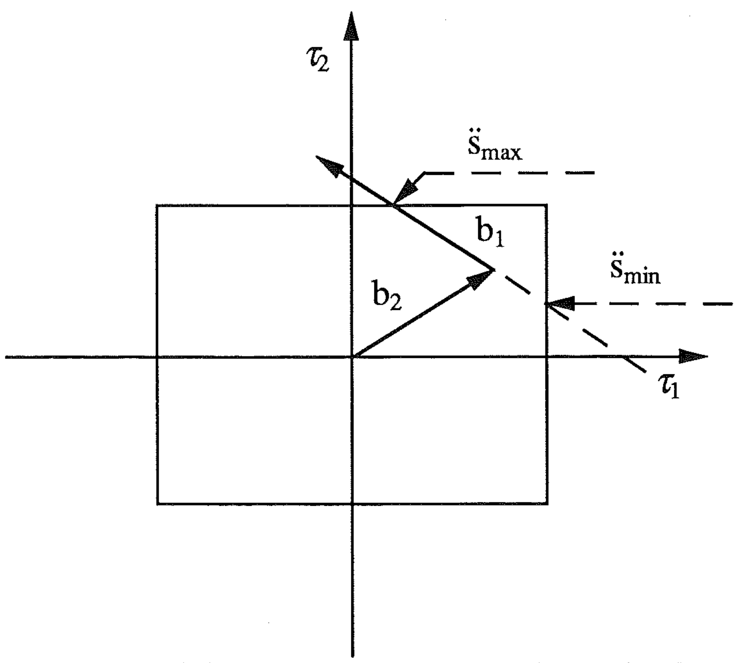


Figure 3.1 A geometric interpretation of the constraints on $\dot{s}$ and $\ddot{s}$. The region E, where τ is admissible, is illustrated by a rectangle.

Admissible region in the phase-plane The constraints on s and $\dot{s}$ define an admissible region in the phase-plane, i.e. in the s-$\dot{s}$-plane. The constraints on s are $s_0 \leq s \leq s_f$, and the constraints on $\dot{s}$ are, for each s, the admissible values of $\dot{s}$. It is thus a necessary condition on a feasible velocity profile to be inside this region.

Boundary of the admissible region The boundary of the admissible region is used in the optimization algorithm. For a given s, if $\dot{s}$ is admissible then $\ddot{s}_{min}(s,\dot{s}) \leq \ddot{s}_{max}(s,\dot{s})$. Hence, if $\ddot{s}_{min}(s,\dot{s}) > \ddot{s}_{max}(s,\dot{s})$, then $\dot{s}$ is not admissible. One might then conclude that at the boundary of the admissible region, it holds that $\ddot{s}_{min}(s,\dot{s}) = \ddot{s}_{max}(s,\dot{s})$. It can however happen, for certain values of s, that at the boundary of the admissible region, it holds that $\ddot{s}_{min}(s,\dot{s}) < \ddot{s}_{max}(s,\dot{s})$. These values of s are called critical points and are defined as follows.

DEFINITION 3.3

A critical point is a value of s such that one or more components of the vector $b_1(s)$ in (C3') are zero. □

This means that a critical point is where the vector $b_1(s)$ is perpendicular to one or more of the coordinate axes. For the two joint case, shown in Figure 3.1, a critical point is where b_1 is parallell to one of the coordinate axes.

A characterization of the boundary of the admissible region is given as follows. For a given s, separate the joints so that, perhaps after renumbering,

$$\begin{aligned} b_{1_i}(s) &\neq 0, \quad i \in I_1 = 1, \ldots, n_1 \\ b_{1_i}(s) &= 0, \quad i \in I_2 = n_1 + 1, \ldots, n \end{aligned}$$

For $\dot{s}$ admissible, we have $\ddot{s}_{min}(s,\dot{s}) \leq \ddot{s}_{max}(s,\dot{s})$. Using (3.26), it is seen that an equivalent formulation is

$$\ddot{s}^i_{min}(s,\dot{s}) \leq \ddot{s}^j_{max}(s,\dot{s}) \tag{3.27}$$

for all $i, j \in I_1$. For $\dot{s}$ admissible, it must also hold that

$$\tau_i^{min} \leq b_2(s,\dot{s}) \leq \tau_i^{max} \tag{3.28}$$

for all $i \in I_2$. At the boundary of the admissible region, we then have, either $\ddot{s}^i_{min}(s,\dot{s}) = \ddot{s}^j_{max}(s,\dot{s})$ for some $i, j \in I_1$, i.e. by (3.26), $\ddot{s}_{min}(s,\dot{s}) = \ddot{s}_{max}(s,\dot{s})$, or $b_2(s,\dot{s}) = \tau_i^m$ for some $i \in I_2$, where τ_i^m is either τ_i^{max} or τ_i^{min}.

We have thus established the following: at the boundary of the admissible region, either of the following alternatives hold.

1. There is one admissible value of $\ddot{s}$, i.e. $\ddot{s}_{min}(s,\dot{s}) = \ddot{s}_{max}(s,\dot{s})$.
2. The admissible values of $\ddot{s}$ are an interval $[\ddot{s}_{min}(s,\dot{s}), \ddot{s}_{max}(s,\dot{s})]$. This can only happen at critical points.

Computation of the admissible region The admissible region can be computed from quadratic inequalities in $\dot{s}$ [Shin and N.D.McKay, 1985]. This is seen as follows. Using (3.24) and (3.25), (3.27) can be written as

$$\frac{\tau_i^m - b_{2_i}(s,\dot{s})}{b_{1_i}(s)} \leq \frac{\tau_j^m - b_{2_j}(s,\dot{s})}{b_{1_j}(s)}$$

where τ_i^m and τ_j^m are either the upper or the lower torque limits. From (3.15), it is seen that this gives a quadratic inequality in $\dot{s}$. Each pair $i, j \in I_1$ thus gives an admissible set for $\dot{s}$. Similarly (3.28) gives, for each $i \in I_2$, admissible sets of $\dot{s}$, also from quadratic inequalities in $\dot{s}$. The intersection of these sets then give an admissible set for $\dot{s}$. Computation of the admissible set for $\dot{s}$, for each s, then gives the admissible region.

Maximum velocity curve Assume that the admissible region is simply connected. Further, assume that $\dot{s} = 0$ is admissible for all s. This means that, for a given s, the admissible values of $\dot{s}$ are defined as $0 \leq \dot{s} \leq \dot{s}_{max} \leq \infty$, where $\dot{s}_{max}$ is the maximum admissible $\dot{s}$. The boundary of the admissible region can then be found by maximization of $\dot{s}$, pointwise in s. This gives a maximum velocity curve $v_{max}(s)$, which is defined as

$$v_{max}(s) = \max_{\dot{s},\ddot{s}} \dot{s} \tag{3.29}$$

subject to the constraints (C3').

Remark 1. In [Shin and N.D.McKay, 1985] it is shown that the admissible region is not always simply connected. An algorithm for handling this case is given. It is also shown that if there is no viscous friction, i.e. $d(q) = 0$ in (3.1), the admissible region is simply connected.

Remark 2. The assumption that $\dot{s} = 0$ is always admissible means that we assume that it is always possible to stop anywhere on the path. The available torque must then be larger that the gravity torque. This is seen by setting $\dot{s} = 0$ in (3.13), which shows that if $\tau_i^{min} \leq a_{4_i}(s) \leq \tau_i^{max}$ for $1 \leq i \leq n$, then $\dot{s} = 0$ is admissible, since e.g. $\ddot{s} = 0$ results in $\tau \in E$.

Possible trajectory points Since the goal of the optimization is to construct a velocity profile with as high velocity as possible, it seems reasonable to search for points on the maximum velocity curve which are also possible points on a velocity profile. We call these points possible trajectory points. Note that, generally, the possible trajectory points are isolated points on the maximum velocity curve. This is due to the fact that the maximum velocity curve is obtained by pointwise maximization of $\dot{s}$ (3.29), i.e. the curve itself is not a possible velocity profile. The possible trajectory points are, e.g. [Shiller and Lu, 1990]:

1. Critical points where the admissible values of $\ddot{s}$ are an interval.
2. Tangency points. These are noncritical points where the unique admissible $\ddot{s}$ satisfies
$$\ddot{s} = \frac{d}{dt} v_{max}(s) = v'_{max}(s) v_1(s)$$
Using (2.5) and (2.6), it is seen that this is equivalent to $v_1'(s) = v'_{max}(s)$, i.e. the slope of the velocity profile is equal to the slope of the maximum velocity curve.
3. Points where the maximum velocity curve is discontinuous. This can happen if $f''(s)$ is discontinuous.

Optimization algorithm

The optimization algorithm constructs the time optimal velocity profile by forward and backward integration with either maximum or minimum acceleration $\ddot{s}$. This means that one of the differential equations

$$\ddot{s} = \ddot{s}_{max}(s, \dot{s}), \quad \ddot{s} = \ddot{s}_{min}(s, \dot{s}) \tag{3.30}$$

is integrated. The integration can also be done in s. This is seen using (2.6) which is equivalently written as

$$\ddot{s} = \frac{d}{ds} \frac{1}{2} \dot{s}^2$$

Introducing y as in (2.24), i.e. $y = \frac{\dot{s}^2}{2}$, now gives that (3.30) can be written as

$$\frac{dy}{ds} = \ddot{s}_{max}(s, \sqrt{2y}), \quad \frac{dy}{ds} = \ddot{s}_{min}(s, \sqrt{2y}) \tag{3.31}$$

The optimization algorithm is formulated as

ALGORITHM 3.1—Phase-plane Optimization

1. Set $s_1 := s_0$, and $\dot{s} := 0$.
2. Integrate forward from s_1 with maximum $\ddot{s}$. Stop if $s = s_f$ or if integration with max $\ddot{s}$ cannot continue without violating $\tau \in E$, i.e. if integration with max $\ddot{s}$ cannot continue without leaving the admissble region. If $s = s_f$, goto 6, otherwise set s_1 to the current s-value and goto 3.
3. From s_1 search forward along the maximum velocity curve for the first possible trajectory point. If no such point is found, goto 6. Denote the possible trajectory point s_2.
4. Integrate backwards from s_2 with minimum $\ddot{s}$. Stop if integration with minimum $\ddot{s}$ cannot continue without violating $\tau \in E$ or an intersection with the previously computed curve occurs. If integration cannot continue without violating $\tau \in E$, search for a new possible trajectory point by setting $s_1 := s_2$ and goto 3, otherwise goto 5.
5. Now we have constructed a curve going backwards from s_2, and which intersects the previously computed curve. Start a new iteration by setting $s_1 := s_2$ and goto 2.
6. Integrate from s_f backwards with minimum $\ddot{s}$. If the curve intersects the previously computed curve, stop, else the algorithm cannot find a solution.

Remark. It may happen that for a possible trajectory point that also is a critical point, it is not possible to use minimum acceleration when integrating backward in step 4, and/or maximum acceleration when integrating forward in step 2. This means that the algorithm cannot proceed at these points. These points are called singular critical points in [Shiller and Lu, 1990], where a modified algorithm to handle this is given. This algorithm also handles the case when the critical points are not isolated. This is called critical arcs in [Shiller and Lu, 1990]. For example, forward integration from an isolated singular critical point s is done by using maximum feasible acceleration for a small interval $[s, s+\varepsilon]$, and then using maximum acceleration $\ddot{s}_{max}(s, \dot{s})$. The maximum feasible acceleration is chosen so that the resulting velocity profile follows the maximum velocity curve in the interval $[s, s+\varepsilon]$. For a singular critical arc, the same technique is used, but for a longer s-interval, i.e. a velocity profile that follows the maximum velocity curve is constructed. □

Numerical Examples

Examples of minimum time velocity profiles are given. These velocity profiles are used in the simulation examples in Chapter 4 as nominal velocity profiles for the path velocity controller.

EXAMPLE 3.1
The first example is a decoupled robot, moving along an elliptical path. The robot model is the decoupled linear model (2.1). The torques are constrained by (2.2). The parameters are chosen as in (2.18), i.e.

$$\tau_i^{min} = -1, \quad \tau_i^{max} = 1, \quad m_i = 1, \quad i = 1,2$$

The path is the ellips

$$f_1(s) = 2\sin(s), \quad f_2(s) = 1 - \cos(s), \quad 0 \le s \le 2\pi \tag{3.32}$$

The path and the minimum time solution are shown in Figure 3.2. The path (3.32) is shown in the upper plot. The solid line in the mid plot is the minimum time velocity profile, computed by Algorithm 3.1. The dashed line in the mid plot is the maximum velocity curve (3.29). The dotted line is 0.5 if $\ddot{s}$ is maximum, and 0 if $\ddot{s}$ is minimum. The lower plot shows the torques τ_1, solid line, and τ_2, dashed line, as functions of s. The torques are computed from the velocity profile using (2.9). The traversal time was computed by assuming constant acceleration between the discretization points in s, resulting in the traversal time $t_f = 9.66$ seconds.

The switching points for the path acceleration $\ddot{s}$ are shown in Figure 3.2 as those points where the dotted line in the mid plot changes value. For this example, the switching points are $s = 0.52$, $s = 1.56$, $s = 3.14$, $s = 4.70$, and $s = 5.77$. The path acceleration switches from maximum to minimum at the switching points $s = 0.52$, $s = 3.14$, and $s = 5.77$, and from minimum to maximum at the switching points $s = 1.56$, and $s = 4.70$. The switching points $s = 1.56$, and $s = 4.70$ were used as starting points for backward and forward integration in steps 2 and 4 in Algorithm 3.1. Since the integration starts from points on the maximum velocity curve, the minimum time velocity profile touches the maximum velocity curve for these points. This can be seen in the mid plot in Figure 3.2. For this example, the velocity profile also touches the maximum velocity curve for the switching point $s = 3.14$. This point was however not used as a starting point for backward and forward integration. Instead, it is a point found both by forward integration from $s = 1.56$ in step 2 of the algorithm, and by backward integration from $s = 4.70$ in step 4 of the algorithm.

The switching points $s = 1.56$ and $s = 4.70$ are also critical points, see Definition 3.3. For this example, the critical points can be computed

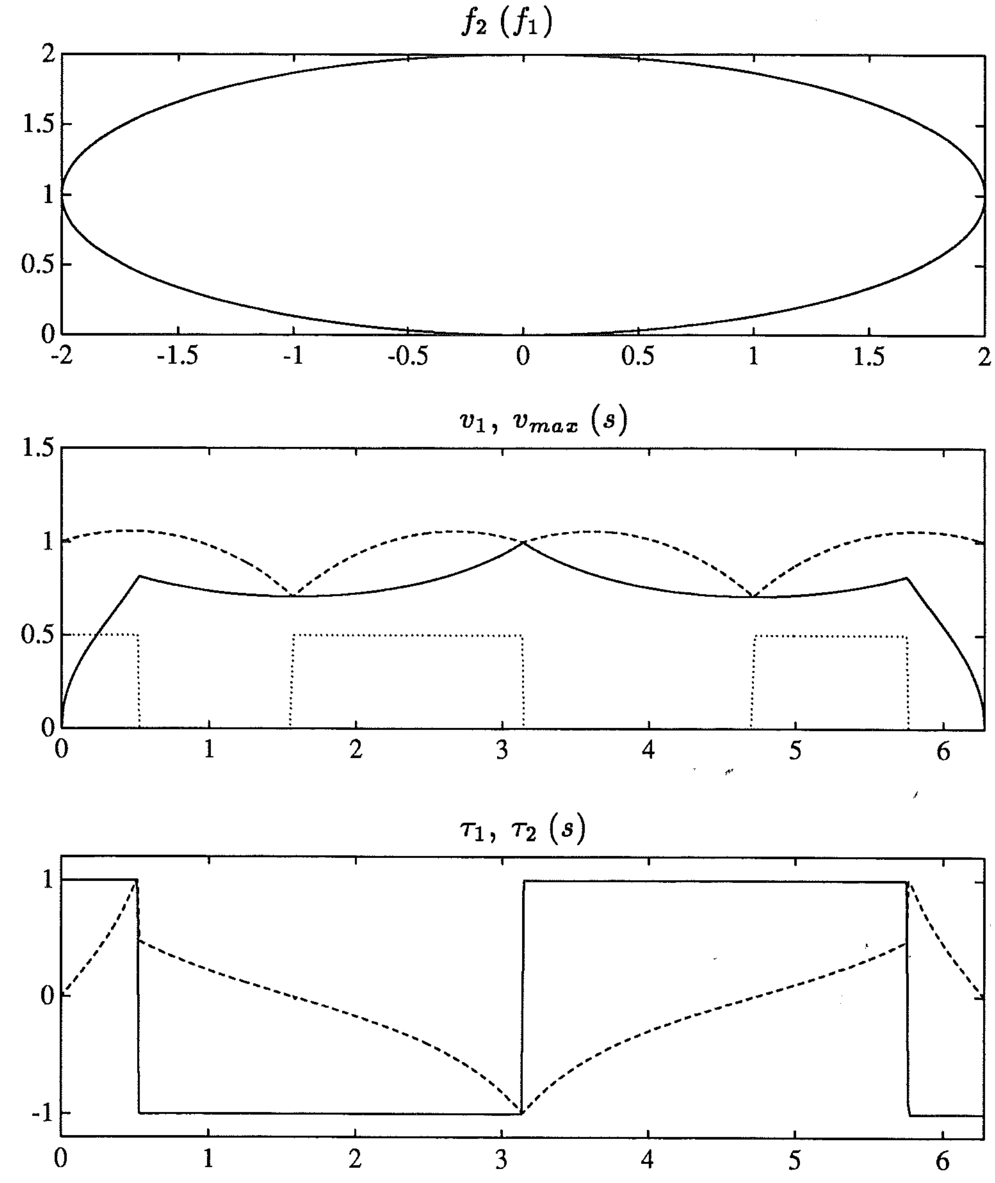


Figure 3.2 An example of a minimum time velocity profile. The upper plot shows the path. The mid plot shows the minimum time velocity profile, solid line, and the maximum velocity curve, dashed line. The dotted line is 0.5 if $\ddot{s}$ is maximum, and 0 if $\ddot{s}$ is minimum.

analytically. From (2.9) and (3.11) with $p = 2$, it is seen that $b_1(s) = Mf'(s)$. Since M is constant and diagonal, this gives, by Definition 3.3, the critical points as those where one or more components of $f'(s)$ are zero. The path (3.32) gives $f_1'(s) = 2\cos(s)$, $f_2'(s) = \sin(s)$. This gives the critical points $s = 0$, $s = \pi/2$, $s = \pi$, $s = 3\pi/2$ and $s = 2\pi$. The numerically obtained critcal points $s = 1.56$, and $s = 4.70$ thus correspond to the critical points

$s = \pi/2 \approx 1.57$ and $s = 3\pi/2 \approx 4.71$.

EXAMPLE 3.2
The same robot and torque constraints as in the previous example are used. The path is composed of two line segments and a circular arc. The path is shown in the upper plot in Figure 3.3. The path is parametrized as

$$
\begin{aligned}
&f_1(s) = \\
&\begin{cases} 2s, & 0 \le s \le 1 \\ 2.1 - 0.1\cos(10s-10) + 0.2\sin(10s-10), & 1 \le s \le 1 + \frac{\pi}{20} \\ 1.3 + s - \frac{\pi}{20}, & 1 + \frac{\pi}{20} \le s \le 2 + \frac{\pi}{20} \end{cases} \\
&f_2(s) = \\
&\begin{cases} s, & 0 \le s \le 1 \\ 0.8 + 0.2\cos(10s-10) + 0.1\sin(10s-10), & 1 \le s \le 1 + \frac{\pi}{20} \\ 2.9 - 2s + \frac{\pi}{10}, & 1 + \frac{\pi}{20} \le s \le 2 + \frac{\pi}{20} \end{cases}
\end{aligned}
\tag{3.33}
$$

The minimum time velocity profile is shown in the mid plot in Figure 3.3. The traversal time for this example was 5.60 seconds. For this path, the maximum velocity curve is finite only for $1 \le s \le 1 + \frac{\pi}{20}$. This is seen as follows. From the path parametrizarion (3.33), it is seen that for $0 \le s \le 1$ and $1 + \frac{\pi}{20} \le s \le 2 + \frac{\pi}{20}$, the second derivative $f''(s)$ is zero. Using (2.9), this gives that the torque τ only depends on $\ddot{s}$. Hence, the limits on $\ddot{s}$ are independent of $\dot{s}$. All values of $\dot{s}$ are therefore admissible, see Definition 3.2.

The switching points are $s = 0.52$, $s = 1.05$, and $s = 1.63$. The path acceleration switches from maximum to minimum at the switching points $s = 0.52$ and $s = 1.63$, and from minimum to maximum at the switching point $s = 1.05$. The minimum time velocity profile touches the maximum velocity curve at this switching point, as can be seen in the mid plot. The switching point $s = 1.05$ is also a critical point. This can be verified from

$$f_2'(s) = -2\sin(10s-10) + \cos(10s-10) = 0$$

which has the solution $s = (\pi/2 + 10 - \arctan(2))/10 \approx 1.046$.

From the path parametrization (3.33), it is seen that for $0 \le s \le 1$, this path is the same as the path used in Section 2.1, i.e. (2.3) with $\alpha_1 = 2$ and $\alpha_2 = 1$. The robot models are also the same. From Figure 2.2, it is seen that the minimum time solutions are also the same for $0 \le s \le 0.5$. The switching point in Figure 2.2 is $s = 0.5$. For this example, the switch occurs at $s = 0.52$. This is seen in the mid plot in Figure 3.3, where the dotted line changes value slightly after $s = 0.5$. For the example shown in Figure 2.2, the velocity at $s = 1$ is determined by the boundary condition on zero velocity $\dot{s}(t_f) = v_1(s_f) = v_1(1) = 0$. Here, it is possible to have $v_1(1) > 0$. The reason for this is that the constraint on the velocity at this point is given by

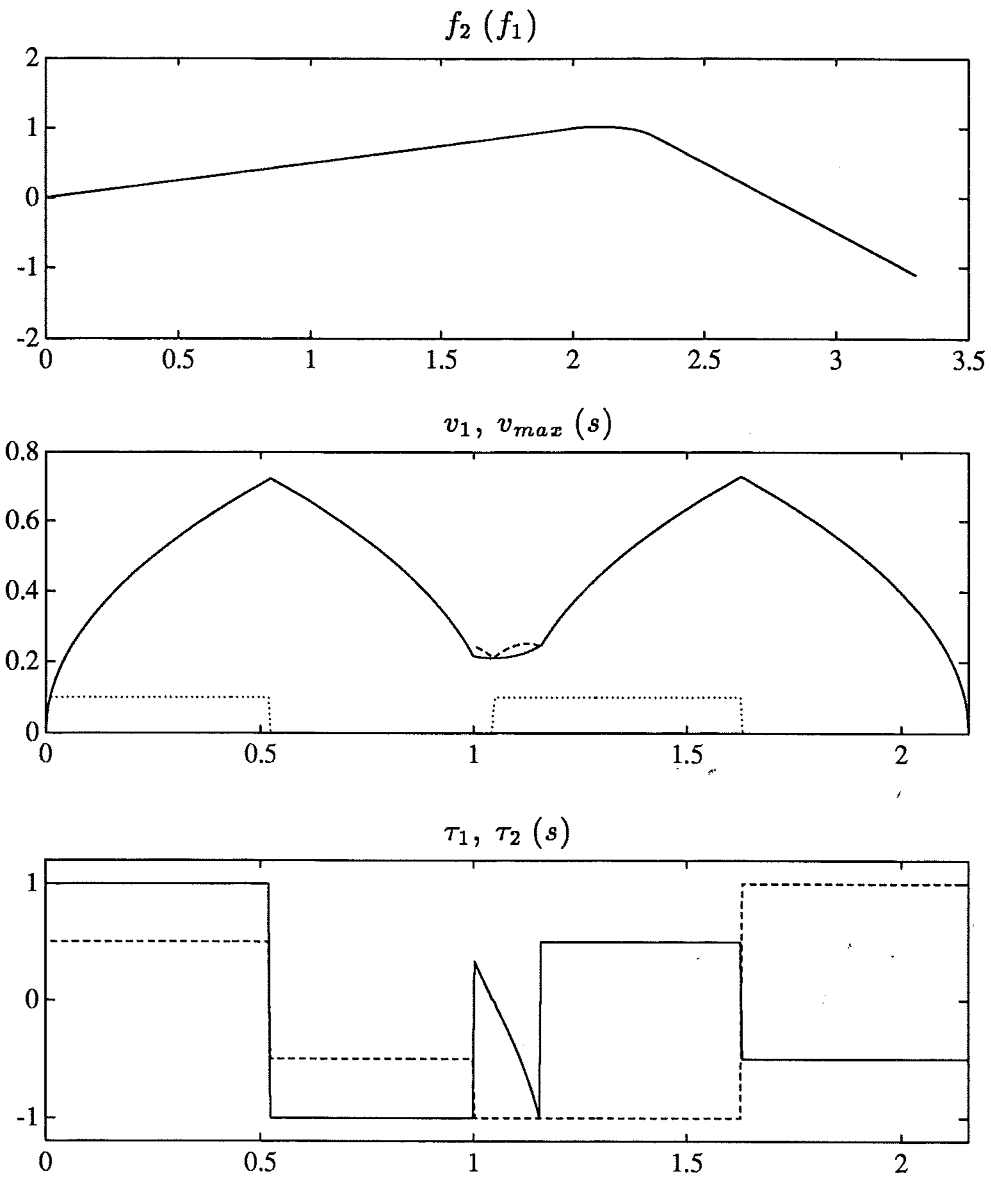


Figure 3.3 An example of a minimum time velocity profile. The path is shown in the upper plot. The path is composed of two straight line segments and a circular arc. The mid plot shows the minimum time velocity profile, solid line, and the maxium velocity curve, dashed line. The maximum velocity curve is not defined for the straight line segments of the path. The dotted line in the mid plot is 0.1 if $\ddot{s}$ is maximum and 0 if $\ddot{s}$ is minimum. The lower plot shows the torques.

the maximum velocity at the critical point $s = 1.05$. It is therefore possible to delay the switch from $s = 0.5$ to $s = 0.52$.

The point $s = 1$ is also a possible trajectory point, since it is a discontinuouity point on the maximum velocity curve. Backward integration from this point would actually give a higher velocity profile than the time optimal. It is however not possible to integrate forward from $s = 1$. This means that in Algorithm 3.1, forward integration in step 2 is not possible, and the algorithm moves directly to step 3, and searches for the next possible trajectory point, which is the critical point $s = 1.05$.

The velocity profile in Figure 3.3 also touches the maximum velocity curve for $s = 1 + \frac{\pi}{20}$. This is however no switching point, as can be seen in the mid plot where the dotted line shows that the path acceleration is kept at maximum through this point. Note also that the torques τ_1 and τ_2 are discontinuous at $s = 1$ and that τ_1 is discontinuous at $s = 1+\frac{\pi}{20}$, even though the path acceleration does not switch at these points. The reason for the discontinuouity in the torques at these points is that $f''(s)$ is discontinuous at these points.

EXAMPLE 3.3
In this example, a nonlinear robot model is used. The dynamic equations are [Asada and Slotine, 1986]

$$\begin{aligned}\tau_1 &= m_2 l_2^2(\ddot{q}_1 + \ddot{q}_2) + m_2 l_1 l_2 c_2(2\ddot{q}_1 + \ddot{q}_2) + (m_1 + m_2) l_1^2 \ddot{q}_1 - m_2 l_1 l_2 s_2 \dot{q}_2^2 \\ &\quad - 2 m_2 l_1 l_2 s_2 \dot{q}_1 \dot{q}_2 + m_2 l_2 g c_{12} + (m_1 + m_2) l_1 g c_1 \\ \tau_2 &= m_2 l_1 l_2 c_2 \ddot{q}_1 + m_2 l_1 l_2 s_2 \dot{q}_1^2 + m_2 l_2 g c_{12} + m_2 l_2^2(\ddot{q}_1 + \ddot{q}_2)\end{aligned} \tag{3.34}$$

where the parameters are chosen as

$$m_1 = m_2 = l_1 = l_2 = 1$$

The path is a circle in cartesian space, described by

$$f_x(s) = 1 + 0.5\cos(s), \quad f_y(s) = 0.5\sin(s), \quad 0 \le s \le 2\pi \tag{3.35}$$

The origin of the cartesian space is at the beginning of link 1. The torques are constrained by $|\ \tau_1\ | \le 30$, $|\ \tau_2\ | \le 10$. The path in joint space and the minimum time solution are shown in Figure 3.4. The path in joint space is shown in the upper plot. The mid plot shows the minimum time velocity profile and the maximum velocity curve. The lower plot shows the torques. The torques are computed from the velocity profile using (3.13). The traversal time for this example was 1.82 seconds.

The switching points are $s = 1.67$, $s = 4.49$ and $s = 6.09$. The path acceleration switches from maximum to minimum at $s = 1.67$ and $s = 6.09$,

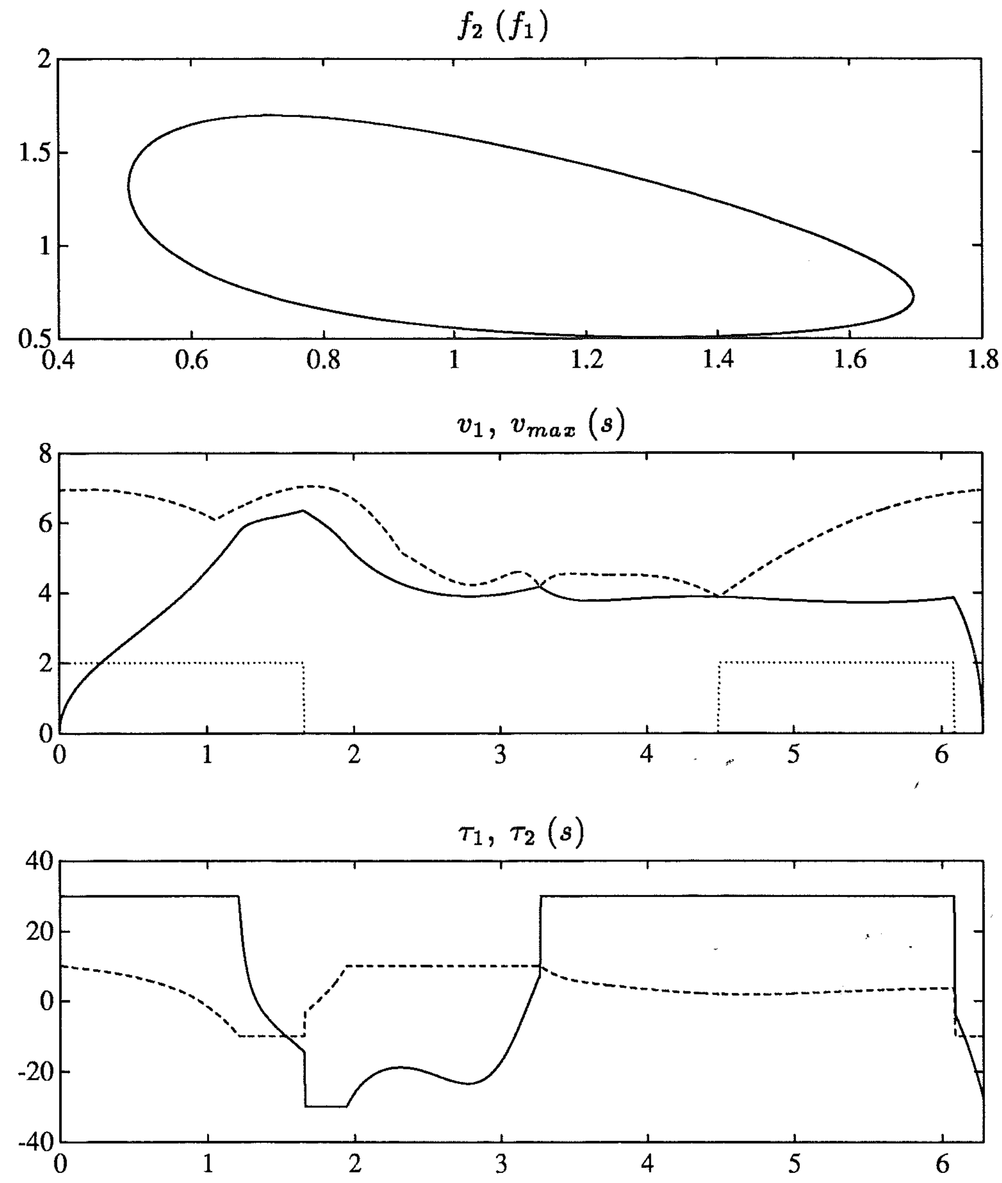


Figure 3.4 A minimum time velocity profile for a nonlinear robot model. The upper plot shows the path in joint space. The dotted line in the mid plot is 2 if $\ddot{s}$ is maximum and 0 if $\ddot{s}$ is minimum. The torque limits are ± 10 and ± 30.

and from minimum to maximum at $s = 4.49$. The critical points were computed as $s = 1.06$, $s = 2.33$, $s = 3.27$, and $s = 4.49$. The switching point $s = 4.49$ is thus a critical point. The velocity profile touches the maximum velocity curve for the critical points $s = 3.27$ and $s = 4.49$. Note however that the critical point $s = 3.27$ is not a switching point. The acceleration is kept at minimum through this point.

Matlab Implementation

The algorithm 3.1 has been implemented in Matlab [MathWorks, 1990] as a set of *m*-files. The main functions are the computation of the maximum velocity curve (3.29) and the minimum time optimization according to Algorithm 3.1. The following Matlab commands are used to generate the minimum time solution in Figure 3.2.

```
>> robot = makerobot([1 1],[0 0]);
>> path = makeellips([1 0],[-1 2],500,0,2*pi);
>> pr = makepathrobot(path,robot);
>> taumin = [-1 -1];
>> taumax = [1 1];
>> vmax = makevmax(pr,taumin,taumax);
>> mt = makemt(pr,vmax,taumin,taumax);
>> plotmt(mt);
```

Listing 3.1 Matlab commands for minimum time optimization

First, a robot is created by makerobot where the arguments are the parameters m_i and d_i in the model

$$m_i\ddot{q}_i + d_i\dot{q}_i = \tau_i, \quad 1 \leq i \leq n$$

The elliptical path, which is shown in the upper plot in Figure 3.2, is then created and the representation for the robot dynamics on the path is stored in the variable pr by calling the function makepathrobot which computes the vectors $a_i(s)$ in (3.14). The torque limits are then chosen and the maximum velocity curve is computed by the function makevmax. The minimum time optimization is then done by the function makemt and the result is displayed by the function plotmt.

Nonlinear two-link robots with revolute joints, i.e. nonlinear dynamics of the type (3.34), are also handled in the implementation. For example, the minimum time solution in Figure 3.4 is generated by the commands

```
>> robot = makerobot([1 1],[0 0],[1 1],[1 1],[0 0]);
>> path = makeellips([1 0],[0.5 0.5],1000,0,2*pi);
>> qpath = pathinvkin(robot,path);
>> pr = makepathrobot(qpath,robot,9.81);
>> taumin = [-30 -10];
>> taumax = [30 10];
>> vmax = makevmax(pr,taumin,taumax);
>> mt = makemt(pr,vmax,taumin,taumax);
>> plotmt(mt);
```

Listing 3.2 Matlab commands for minimum time optimization for a nonlinear two link robot

The argument list to the function `makerobot` is here extended with specification of the link lengths, the distance from the joints to the centers of mass, and the moment of inertias for the links. The path is specified in cartesian space by `makeellips` and then converted to joint space by the function `pathinvkin`. The minimum time optimization is then done, and the result is displayed.

Singular critical points Consider the critical point $s = \pi/2$ in Figure 3.2. This critical point is also a singular critical point, i.e. the optimization algorithm could not use maximum or minimum acceleration at this point. For a singular critical point, the algorithm actually chooses the acceleration such that the limiting torque, in this case τ_1, is continuous through the critical point. This acceleration is used for a small interval $[s, s + \varepsilon]$, and then maximum or minimum acceleration is used. Another approach taken by [Shiller and Lu, 1990] was discussed following the Algorithm 3.1, where the idea instead is to follow the maximum velocity curve for a small distance, and then use maximum or minimum acceleration.

Note however, that if the critical points are isolated, then, for ε sufficiently small, any choice of acceleration in $[s, s+\varepsilon]$ would suffice. This follows from the bounds on $\ddot{s}$ (3.26) being finite, see the discussion following (3.26). The contribution to the loss function (L3) of using different values of $\ddot{s}$ then tends to zero as as $\varepsilon \to 0$.

3.4 An Optimality Result

We give here an optimality result for the minimum time case, when the optimal control problem is (L2,D2,C2',B2). The result is that for s such that all components of $b_1(s)$ are nonzero, i.e. for s not being a critical point or part of a critical arc, see Definition 3.3, the minimum time solution of (L2,D2,C2',B2) has the property that $s^{(p)}$ is either maximum or minimum. The case $p = 2$ was treated by [Chen, 1988], however not considering the possibility of critical points. The proof is a straightforward application of section 13.11 in [Leitmann, 1981].

For the case $p = 4$, the result shows that for the minimum time control of the flexible joint robot (3.2) along the path $f(s)$ we have, for those s where all components of $b_1(s)$ are nonzero, either maximum or minimum $s^{(4)}$. Note also that, if J and K in (3.2) are diagonal, then the s-values for which the result does not hold, i.e. where one or more components of $b_1(s)$ are zero, are the same as for the rigid case, since, using (3.21) and (3.22), we get

$$b_1(s) = JK^{-1}a_1(s)$$

where $a_1(s)$ is defined in (3.14).

THEOREM 3.1
For the optimal control problem (L2,D2,C2',B2), the minimum time solution has the property that $u = s^{(p)}$ is maximized or minimized when s is such that all components of the vector $b_1(s)$ are nonzero.

Proof: Introduce the notation

$$z = \begin{pmatrix} z_1 \\ \vdots \\ z_p \end{pmatrix} = \begin{pmatrix} s \\ \vdots \\ s^{(p-1)} \end{pmatrix}, \quad u = s^{(p)} \tag{3.36}$$

The p-integrator (D2) is then written in state space form as

$$\dot{z} = Az + Bu \tag{3.37}$$

where

$$A = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \vdots \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & \dots & 0 & 1 \end{pmatrix}^T \tag{3.38}$$

The result is now obtained by applying Theorem 11.1 in [Leitmann, 1981, p. 118], with the extension to state dependent constraints on the input [Leitmann, 1981, p. 173]. Suppose that the function $u^*(t)$, with the corresponding state $z^*(t)$, given by (3.37), minimizes (L2) with $L_s = 1$, and the constraints (D2,C2',B2). Further, define the set $U(z)$ such that

$$u \in U(z) \Longleftrightarrow \tau \in E$$

where τ, using (3.11) and (3.36), is written as

$$\tau = b_1(z_1)u + b_2(z)$$

Then there is a vector function

$$\lambda(t) = \begin{pmatrix} \lambda_1(t) & \dots & \lambda_p(t) \end{pmatrix}^T$$

and a function $H(\lambda, z, u)$, given by

$$\begin{aligned} H(\lambda, z, u) &= \lambda_0 + \lambda^T(Az + Bu) \\ &= \lambda_0 + \lambda_1 z_2 + \lambda_2 z_3 + \ldots + \lambda_{p-1} z_p + \lambda_p u \end{aligned} \tag{3.39}$$

where λ_0 is constant, such that

$$\begin{aligned} &\max_{u \in U(z)} H(\lambda, z^*, u) = H(\lambda, z^*, u^*) \\ &H(\lambda, z^*, u^*) = 0 \end{aligned} \tag{3.40}$$

and where $\lambda(t)$ is a nonzero solution to the adjoint equations

$$\dot{\lambda}^T = -\lambda^T A + \nu^T \frac{\partial R}{\partial z} \tag{3.41}$$

The function R is formed from the active constraints in (C2') and ν is given by

$$\nu^T \frac{\partial R}{\partial u} = \lambda^T B \tag{3.42}$$

One should note that the adjoint equations (3.41) may be different for different subintervals of $[t_0, t_f]$. This is due to the fact that different constraints may be active in different subintervals.

We now show that for s such that all components of $b_1(s)$ are nonzero, the optimal control is found by maximizing or minimizing u. The derivation is analogous to the example given in section 13.11 in [Leitmann, 1981]. Suppose $t \in [t_\alpha, t_\beta]$ and that in this interval it holds that $b_{1i}(z_1(t)) \neq 0, 1 \leq i \leq n$. Further suppose that l constraints are active during $[t_\alpha, t_\beta]$, i.e. l components of the torque vector τ are at the limit. It is however then no restriction to assume $l = 1$, since, if τ_i is at the limit for some i, this gives u uniquely from

$$b_{1i}(z_1)u + b_{2i}(z) = \tau_i^m$$

where τ_i^m is either the maximum or the minimum limit for joint i, and hence the other active constraints must also result in the same u. We now have two possibilities. If $l = 0$, no constraints are active and we get $\nu = 0$ in (3.41). If $l = 1$, this gives the function R as

$$R = b_{1i}(z_1)u + b_{2i}(z) - \tau_i^m = 0 \tag{3.43}$$

where τ_i^m is either τ_i^{max} or τ_i^{min}. We now compute the right hand side in (3.41), using (3.42). This gives

$$\frac{\partial R}{\partial u} = b_{1i}(z_1)$$

and

$$\frac{\partial R}{\partial z} = \begin{pmatrix} \frac{\partial b_{1i}}{\partial z_1}u + \frac{\partial b_{2i}}{\partial z_1} & \frac{\partial b_{2i}}{\partial z_2} & \cdots & \frac{\partial b_{2i}}{\partial z_p} \end{pmatrix}$$

This gives, using (3.38), the two possibilities for the adjoint equations (3.41) as

$$\begin{aligned} \dot{\lambda}_1 &= 0 \\ \dot{\lambda}_j &= -\lambda_{j-1}, \quad j = 2, \dots, p \end{aligned} \tag{3.44}$$

or

$$\begin{aligned} \dot{\lambda}_1 &= \frac{\lambda_p}{b_{1i}(z_1)}\left(\frac{\partial b_{1i}}{\partial z_1}u + \frac{\partial b_{2i}}{\partial z_1}\right) \\ \dot{\lambda}_j &= -\lambda_{j-1} + \frac{\lambda_p}{b_{1i}(z_1)}\frac{\partial b_{2i}}{\partial z_j}, \quad 2 \le j \le p \end{aligned} \tag{3.45}$$

The idea is now to show that assuming $\lambda_p(t) \equiv 0$ for $t \in [t_\alpha, t_\beta]$ leads to all components of λ being identically zero. We get the following implications

$$\begin{aligned} \lambda_p(t) \equiv 0 \Rightarrow \dot{\lambda}_p(t) \equiv 0 \Rightarrow \lambda_{p-1}(t) \equiv 0 \Rightarrow \dot{\lambda}_{p-1}(t) \equiv 0 \Rightarrow \dots \Rightarrow \lambda_2(t) \equiv 0 \\ \Rightarrow \dot{\lambda}_2(t) \equiv 0 \Rightarrow \lambda_1(t) \equiv 0 \end{aligned}$$

which shows that $\lambda(t) \equiv 0$ on the subinterval $[t_\alpha, t_\beta]$. This implies, by the bilinear structure of (3.44) and (3.45), and since $\lambda(t)$ is continuous, that $\lambda(t) \equiv 0$ on the interval $[t_0, t_1]$. This gives a contradiction to the optimality condition of a nonzero solution to the adjoint equations (3.41). Since the interval $[t_\alpha, t_\beta]$ is any subinterval of $[t_0, t_f]$ where all components of b_1 are nonzero, this shows that there cannot be any such subinterval where λ_p is identically zero. Hence, the first equation in (3.40) together with (3.39) shows that $u = s^{(p)}$ must be either maximized or minimized. □

3.5 Optimization over a Fixed Interval

The optimal control problem with reduced state dimension, (L2, D2, C2, B2) is defined over an unspecified time interval $[t_0, t_f]$. The problem will now be converted to a fixed time optimal control problem, where the path parameter s is interpreted as time variable. This idea is also used in phase-plane optimization, Section 3.3, where the optimization algorithm constructs the solution as a function of s.

The conversion to optimization over a fixed interval is done by deriving a dynamic system in s. The system is of order $p - 1$, and the input is, as in (D2), the p-th order time derivative of the path parameter, $s^{(p)}$, however now regarded as a function of s. The states in the system are denoted $x_1, \dots, x_{p-1}$. The state variable x_1 is chosen as

$$x_1 = \frac{\dot{s}^p}{p} \tag{3.46}$$

The parametric optimization method, which is presented in Section 3.6, is based on polynomial approximation of x_1 in (3.46).

The motivation for the choice (3.46) is that the boundary conditions when using polynomial approximation become consistent. This was discussed for the case $p = 2$ in Section 2.1, see the discussion following (2.24). The case $p = 4$ is treated in Section 3.6.

A Dynamic System in s

A dynamic system is derived by finding $p-1$ first order differential equations in s for the state variables $x_1, \dots, x_{p-1}$. Define the function g as

$$g(x_1) = (px_1)^{1/p} \tag{3.47}$$

The path velocity $\dot{s}$ can then be expressed as a function of x_1, using (3.46), as

$$\dot{s} = g(x_1) \tag{3.48}$$

Let the state variables $x_1, \dots, x_{p-1}$ be related as

$$x_k = \frac{dx_{k-1}}{ds}, \quad k = 2, \dots, p-1 \tag{3.49}$$

A dynamic system in s is now obtained by expressing $\dot{s}, \dots, s^{(p)}$ as functions of $x_1, \dots, x_{p-1}$, and $\frac{dx_{p-1}}{ds}$. The following result is used.

Lemma 3.2
The p-th order time derivative of s can be expressed as

$$s^{(p)} = g'(x_1)g(x_1)^{p-1}\frac{dx_{p-1}}{ds} + F_p(x_1, \dots, x_{p-1}) \tag{3.50}$$

The function F_p is given below in the proof.

Proof: The proof is by induction. Differentiation of (3.48) with respect to time gives

$$\begin{aligned} \dot{s} &= g(x_1) \\ \ddot{s} &= g'(x_1)\frac{dx_1}{ds}\dot{s} = g'(x_1)g(x_1)\frac{dx_1}{ds} \end{aligned} \tag{3.51}$$

Thus, (3.50) holds for $p = 2$ with $F_2 = 0$. Suppose now that the Lemma holds for $s^{(k-1)}$, where $k \geq 3$, i.e. that

$$s^{(k-1)} = g'(x_1)g(x_1)^{k-2}\frac{dx_{k-2}}{ds} + F_{k-1}(x_1,\ldots,x_{k-2}) \qquad (3.52)$$

Differentiating (3.52) with respect to time gives, using (3.48) and (3.49),

$$\begin{aligned} s^{(k)} &= g''(x_1)x_2g(x_1)g(x_1)^{k-2}x_{k-1} \\ &+ g'(x_1)(k-2)g(x_1)^{k-3}g'(x_1)x_2g(x_1)x_{k-1} \\ &+ g'(x_1)g(x_1)^{k-2}\frac{dx_{k-1}}{ds}g(x_1) + \frac{d}{dt}F_{k-1}(x_1,\ldots,x_{k-2}) \end{aligned}$$

Further, we get

$$\frac{d}{dt}F_{k-1}(x_1,\ldots,x_{k-2}) = \sum_{i=1}^{k-2}\frac{\partial F_{k-1}(x_1,\ldots,x_{k-2})}{\partial x_i}x_{i+1}g(x_1)$$

which gives

$$s^{(k)} = g'(x_1)g(x_1)^{k-1}\frac{dx_{k-1}}{ds} + F_k(x_1,\ldots,x_{k-1})$$

where

$$\begin{aligned} F_k(x_1,\ldots,x_{k-1}) &= g''(x_1)x_2g(x_1)g(x_1)^{k-2}x_{k-1} \\ &+ g'(x_1)(k-2)g(x_1)^{k-3}g'(x_1)x_2g(x_1)x_{k-1} \\ &+ \sum_{i=1}^{k-2}\frac{\partial F_{k-1}(x_1,\ldots,x_{k-2})}{\partial x_i}x_{i+1}g(x_1) \end{aligned}$$

□

Using (3.47), we now get

$$g'(x_1)g(x_1)^{p-1} = \frac{1}{p}(px_1)^{1/p-1}p(px_1)^{(p-1)/p} = 1$$

which, using Lemma 3.2, gives that the p-th order derivative of s can be written as

$$s^{(p)} = \frac{dx_{p-1}}{ds} + F_p(x_1,\ldots,x_{p-1}) \qquad (3.53)$$

A dynamic system in s is now obtained using (3.49) and (3.53). Introduce the notation $u = s^{(p)}$. The dynamic system is then given by

$$\begin{aligned}
\frac{dx_1}{ds} &= x_2 \\
\frac{dx_2}{ds} &= x_3 \\
&\dots \\
\frac{dx_{p-2}}{ds} &= x_{p-1} \\
\frac{dx_{p-1}}{ds} &= u - F_p(x_1, \dots, x_{p-1})
\end{aligned} \tag{3.54}$$

The function g The definition of g (3.47) has the following consequences.

1. It gives a constant, nonzero term, i.e. one, in front of $\frac{dx_{p-1}}{ds}$. Compare (3.50) and (3.53). This means that the order of the dynamic system (3.54) is constant.
2. The system (3.54) is feedback linearizable. This will be used in Chapter 4.

For the two models (3.1) and (3.2), we have $p = 2$ and $p = 4$. The dynamic system (3.54) is now derived for these two cases.

EXAMPLE 3.4
For the case $p = 2$, the input is $u = \ddot{s}$, and the state variable x_1 is obtained from (3.46) as

$$x_1 = \frac{\dot{s}^2}{2}$$

This gives $\dot{s}$ as

$$\dot{s} = (2x_1)^{1/2}$$

Computing $\ddot{s}$, using the chain rule, then gives

$$\ddot{s} = \frac{1}{2}(2x_1)^{-1/2} 2 \frac{dx_1}{ds} \dot{s} = \frac{1}{2}(2x_1)^{-1/2} 2 \frac{dx_1}{ds} (2x_1)^{1/2} = \frac{dx_1}{ds}$$

which gives the dynamic system (3.54) as a linear system in s as

$$\frac{dx_1}{ds} = u \tag{3.55}$$

EXAMPLE 3.5
For the case $p = 4$, we get, using (3.46)

$$\dot{s} = (4x_1)^{1/4}$$

which gives $\ddot{s}$ as

$$\begin{aligned}\ddot{s} &= \frac{1}{4}(4x_1)^{-3/4}4\dot{x}_1 = \frac{1}{4}(4x_1)^{-3/4}4\frac{dx_1}{ds}\dot{s}\\ &= \frac{1}{4}(4x_1)^{-3/4}4x_2(4x_1)^{1/4} = x_2(4x_1)^{-1/2}\end{aligned}$$

Further differentiation gives

$$\begin{aligned}s^{(3)} &= x_3(4x_1)^{-1/4} - 2x_2^2(4x_1)^{-5/4}\\ s^{(4)} &= \frac{dx_3}{ds} + 5(\frac{x_2^3}{8x_1^2} - \frac{x_2x_3}{4x_1})\end{aligned}$$

The dynamic system (3.54) is now obtained as

$$\begin{aligned}\frac{dx_1}{ds} &= x_2\\ \frac{dx_2}{ds} &= x_3\\ \frac{dx_3}{ds} &= u - 5(\frac{x_2^3}{8x_1^2} - \frac{x_2x_3}{4x_1})\end{aligned} \tag{3.56}$$

where $u = s^{(4)}$.

Optimal control problem

The optimal control problem (L2,D2,C2,B2) can now be reformulated as an optimal control problem over the fixed interval $[s_0, s_f]$. The loss function in (L2) depends on $s, \dot{s}, \ldots, s^{(p)}$. From Lemma 3.2, it is seen that $\dot{s}, \ldots, s^{(p-1)}$ can be expressed as functions of $x_1, \ldots, x_{p-1}$. Using $u = s^{(p)}$, this gives that the loss function in (L2) can be written as

$$\int_{s_0}^{s_f} L_x(s, x_1, \ldots, x_{p-1}, u)dt$$

Similary, the function h_s in (C2) can be written as

$$h_x(s, x_1, \ldots, x_{p-1}, u)$$

The following optimization problem is obtained: find the function $u(s)$ that solves

$$\min_{u(s)} \int_{s_0}^{s_f} L_x(s, x_1, \ldots, x_{p-1}, u)dt \tag{L4}$$

with the constraints (3.54), i.e.

$$\begin{aligned} \frac{dx_1}{ds} &= x_2 \\ \frac{dx_2}{ds} &= x_3 \\ &\ldots \\ \frac{dx_{p-2}}{ds} &= x_{p-1} \\ \frac{dx_{p-1}}{ds} &= u - F_p(x_1, \ldots, x_{p-1}) \end{aligned} \tag{D4}$$

and

$$\begin{aligned} &h_x(s, x_1, \ldots, x_{p-1}, u) = \tau \\ &\tau \in E \end{aligned} \tag{C4}$$

For the models (3.1) and (3.2), (C4) can be written as

$$\begin{aligned} &b_1(s)u + b_2(s, x_1, \ldots, x_{p-1}) = \tau \\ &\tau \in E \end{aligned} \tag{C4'}$$

The boundary conditions now become that

$$\begin{aligned} &x_1(s_0), \ldots, x_{p-1}(s_0) \\ &x_1(s_f), \ldots, x_{p-1}(s_f) \end{aligned} \tag{B4}$$

should be given. For minimum time optimization, the loss function (L4) is obtained from (L2) as

$$\int_{t_0}^{t_f} dt = \int_{s_0}^{s_f} \frac{1}{\dot{s}} ds = \int_{s_0}^{s_f} (px_1)^{-1/p} ds \tag{3.57}$$

3.6 Parametric Optimization

A parametric optimization problem is now obtained from the optimal control problem (L4,D4,C4',B4) by approximating the first state variable $x_1(s) = \frac{\dot{s}^p}{p}$

by a piecewise polynomial function. We choose B-splines [de Boor, 1978] for doing this, and then get

$$x_1(s) = \sum_{i=1}^{n_\beta} \beta_i B_i(s) \tag{3.58}$$

where the functions $B_i(s)$ are B-splines. Introduce the vector

$$\beta = \begin{pmatrix} \beta_1 & \dots & \beta_{n_\beta} \end{pmatrix}$$

From (3.58), it is seen that x_1 can be regarded as a function of β. Using (3.54), it is seen that the variables $x_2, \dots, x_{p-1}$, and u are also functions of the vector β. Hence, the loss function in (L4) and the torque τ in (C4') are also functions of β. Suppose that s is discretized in m discretization points s_i. The loss function in (L4) is then approximated by a sum as

$$\sum_{i=1}^{m} L_x(s_i, x_1(s_i,\beta), \dots, x_{p-1}(s_i,\beta), u(s_i,\beta))$$

The torque vector in (C4') is evaluated at the discretization points in s as

$$\tau_i = b_1(s_i)u(s_i,\beta) + b_2(s_i, x_1(s_i,\beta), \dots, x_{p-1}(s_i,\beta)), \quad i = 1, \dots, m$$

The following optimization problem is now obtained: find the vector β that solves

$$\min_\beta \sum_{i=1}^{m} L_x(s, x_1(s_i,\beta), \dots, x_{p-1}(s_i,\beta), u(s_i,\beta)) \tag{$L5$}$$

with the constraints

$$\begin{aligned} &\tau_i = b_1(s_i)u(s_i,\beta) + b_2(s_i, x_1(s_i,\beta), \dots, x_{p-1}(s_i,\beta)) \\ &\tau_i \in E, \quad i = 1, \dots m \end{aligned} \tag{$C5'$}$$

The boundary conditions are the same as before, i.e.

$$\begin{aligned} &x_1(s_0), \dots, x_{p-1}(s_0) \\ &x_1(s_f), \dots, x_{p-1}(s_f) \end{aligned} \tag{$B5$}$$

should be given. For minimum time optimization, the loss function becomes, using (3.57)

$$\sum_{i=1}^{m} (px_1(s_i,\beta))^{-1/p} \tag{3.59}$$

The optimization problem (L5,C5',B5) is a nonlinear parametric optimization problem with nonlinear inequality constraints (C5'). The number of variables is n_β, see (3.58), and the number of constraints is $2nm + 2(p-1)$, i.e. upper and lower limits for each of the n joints that must be satisfied at the m discretization points in (C5'), and $2(p-1)$ boundary conditions from (B5).

Rigid Robots

For the case $p = 2$, we get $x_1(s) = \frac{\dot{s}^2}{2}$, i.e. $\frac{\dot{s}^2}{2}$ is approximated. Another approach is taken by [Marin, 1988] where instead the function $\frac{1}{\dot{s}}$ is approximated. The boundary conditions on zero robot velocity at the start and end points of the path is there solved by requiring $f'(s_0) = 0$ and $f'(s_f) = 0$. The method is also only given for the rigid case.

Boundary Conditions for $p = 4$

We discuss the choice of boundary conditions for the case $p = 4$. The boundary conditions for the case $p = 2$ were discussed in Chapter 2, see the discussion following (2.24).

The dynamic system is given by (3.56). Assume that $x_1(s)$ is chosen as a polynomial satisfying

$$x_1(s_i) = 0, \quad \frac{dx_1(s_i)}{ds} = 0, \quad \frac{d^2x_1(s_i)}{ds^2} = 0, \quad \frac{d^3x_1(s_i)}{ds^3} \neq 0 \tag{3.60}$$

where $s_i = s_0$ or $s_i = s_f$. Using (3.49), this can be written as

$$x_1(s_i) = 0, \quad x_2(s_i) = 0, \quad x_3(s_i) = 0, \quad x_4(s_i) \neq 0,$$

We now show that these boundary conditions imply

$$\dot{s}(t_i) = 0, \quad \ddot{s}(t_i) = 0, \quad s^{(3)}(t_i) = 0, \quad s^{(4)}(t_i) \neq 0, \tag{3.61}$$

where $t_i = t_0$ or $t_i = t_f$. Using (3.60), $x_1(s)$ can be written as

$$x_1(s) = (s - s_i)^3 P(s) \tag{3.62}$$

where $P(s_i) \neq 0$. From the derivation of (3.56), we get

$$\begin{aligned}
\dot{s} &= (4x_1)^{1/4} \\
\ddot{s} &= x_2(4x_1)^{-1/2} \\
s^{(3)} &= x_3(4x_1)^{-1/4} - 2x_2^2(4x_1)^{-5/4} \\
s^{(4)} &= \frac{dx_3}{ds} + 5\left(\frac{x_2^3}{8x_1^2} - \frac{x_2x_3}{4x_1}\right)
\end{aligned} \tag{3.63}$$

Substitution of (3.62) into (3.63), using (3.49), now gives the result (3.61).

3.7 Numerical Examples

The parametric optimization method for the cases $p = 2$ and $p = 4$ has been implemented, using Matlab [MathWorks, 1990] and the optimization routine NPSOL [Gill *et al.*, 1986]. The implementation is done for the minimum time case. The optimization problem is thus to minimize (3.59) subject to the constraints (C5') and (B5). The computation of the loss function (3.59) and the constraints (C5') are implemented as m-files, which are supplied to the optimization routine. The B-spline calculations are done in Matlab using routines from [de Boor, 1990]. An example for the case $p = 2$ was presented in Chapter 2. The result of the optimization is shown in Figure 2.3. Numerical examples for the case $p = 4$ are presented below.

Robot Model

The robot model is a linear, decoupled, flexible joint model with two joints. It is written on the form (3.2) as

$$\begin{aligned} M\ddot{q} + K(q-\theta) &= 0 \\ J\ddot{\theta} &= \tau + K(q-\theta) \end{aligned} \tag{3.64}$$

where

$$M = \begin{pmatrix} m_1 & 0 \\ 0 & m_2 \end{pmatrix}, \quad J = \begin{pmatrix} J_1 & 0 \\ 0 & J_2 \end{pmatrix}, \quad K = \begin{pmatrix} k_1 & 0 \\ 0 & k_2 \end{pmatrix} \tag{3.65}$$

The torques are constrained by

$$\tau_i^{min} \le \tau_i \le \tau_i^{max}, \quad i = 1, 2 \tag{3.66}$$

Using (3.6), the model (3.64) can be written as

$$\tau = JK^{-1}Mq^{(4)} + (M+J)\ddot{q} \tag{3.67}$$

When $K \to \infty$, this model reduces to the rigid model

$$\tau = (M+J)\ddot{q} \tag{3.68}$$

The model (3.64) is now written in the form (3.11). This can be done by using (3.20), (3.21), and (3.22). It can also be done more directly by differentiating $q = f(s)$ with respect to time, and substituting into (3.67). The result is

$$\begin{aligned} \tau = {} & JK^{-1}M(f'(s)s^{(4)} + 4f''(s)\dot{s}s^{(3)} + 6f'''(s)\dot{s}^2\ddot{s} + 3f''(s)\ddot{s}^2 + f''''(s)\dot{s}^4) \\ & + (J+M)(f'(s)\ddot{s} + f''(s)\dot{s}^2) \end{aligned} \tag{3.69}$$

Exact solution

In the first three examples, Example 3.6–3.8 below, the path is chosen as (2.3). Substitution of $f(s)$ from (2.3) into (3.69), using (3.65), then gives

$$\tau_i = \frac{J_i m_i}{k_i} \alpha_i s^{(4)} + (J_i + m_i)\alpha_i \ddot{s}, \quad i = 1,2 \tag{3.70}$$

Assume now that the time optimal solution has the property that one of the joints, e.g. joint one is limiting the motion, i.e. assume that the minimum time solution is such that the other torque, in this case τ_2, never reaches the torque limit. The optimization problem then becomes a point to point optimization problem for the scalar system

$$\tau_1 = \frac{J_1 m_1}{k_1} \alpha_1 s^{(4)} + (J_1 + m_1)\alpha_1 \ddot{s} \tag{3.71}$$

The problem is then to minimize traversal time subject to the constraints

$$\tau_1^{min} \leq \tau_1 \leq \tau_1^{max}$$

and the boundary conditions

$$\begin{array}{llll} s(t_0) = s_0, & \dot{s}(t_0) = 0, & \ddot{s}(t_0) = 0, & s^{(3)}(t_0) = 0 \\ s(t_f) = s_f, & \dot{s}(t_f) = 0, & \ddot{s}(t_f) = 0, & s^{(3)}(t_f) = 0 \end{array}$$

A method for obtaining an exact solution to this problem has been implemented [Dahl, 1992] and is used below for comparison with the result of the polynomial approximation. The exact solution is obtained by integrating the system dynamics (3.71) analytically when the input τ_1 is alternating between τ_1^{min} and τ_1^{max}. The resulting state is then set to a specified value, and equations for the switching times for τ_1 are obtained. The optimality is then checked as in [Ben-Asher *et al.*, 1987], by substituting the solution into the necessary, and in this case also sufficient, conditions for optimality. The implementation was done using Maple [Char *et al.*, 1988] to obtain the equations for the switching times. The implementation uses Matlab [MathWorks, 1990] for numerical calculations to check the optimality of a given solution.

Parametric Optimization

The parametric optimization requires the computation of the constraints (C5'). The constraints are obtained by expressing τ in (3.69) as a function of the parameter vector β in (3.58). This is done as follows. Using (3.63), $\dot{s}$, $\ddot{s}$, $s^{(3)}$ and $s^{(4)}$ can be expressed as functions of x_1, x_2, x_3, and $\frac{dx_3}{ds}$. Using (3.58) and (3.56) then gives τ as a function of β.

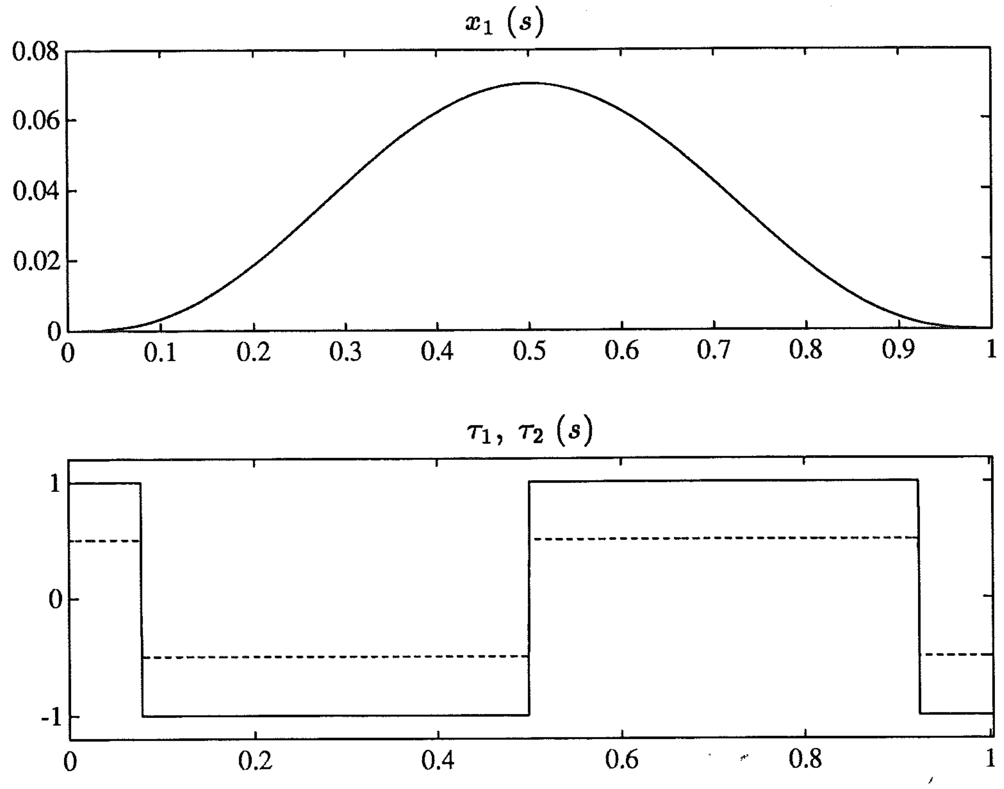


Figure 3.5 The exact time optimal solution. The joint stiffness constants are $k_1 = k_2 = 1$. The upper plot shows the optimal $x_1(s)$. The lower plot shows the corresponding torques τ_1, solid line, and τ_2, dashed line.

Numerical Examples

EXAMPLE 3.6

The path is the same as was used in Chapter 2, i.e. (2.3) with the parameter choice $\alpha_1 = 2$ and $\alpha_2 = 1$. The parameter values in (3.65) are chosen as

$$m_1 = m_2 = 0.5, \quad J_1 = J_2 = 0.5, \quad k_1 = k_2 = 1 \tag{3.72}$$

The torque limits are chosen as

$$\tau_i^{min} = -1, \quad \tau_i^{max} = 1, \quad i = 1,2$$

For the parameter choice (3.72), the corresponding rigid model (3.68) becomes the model (2.1) with $m_1 = m_2 = 1$, which is the model used in Chapter 2. The rigid solution, shown in Figure 2.2, is used below for comparison. The exact solution is shown in Figure 3.5. The upper plot shows the optimal $x_1(s)$, i.e. the optimal $\frac{\dot{s}^4}{4}$ as a function of s. The lower plot shows

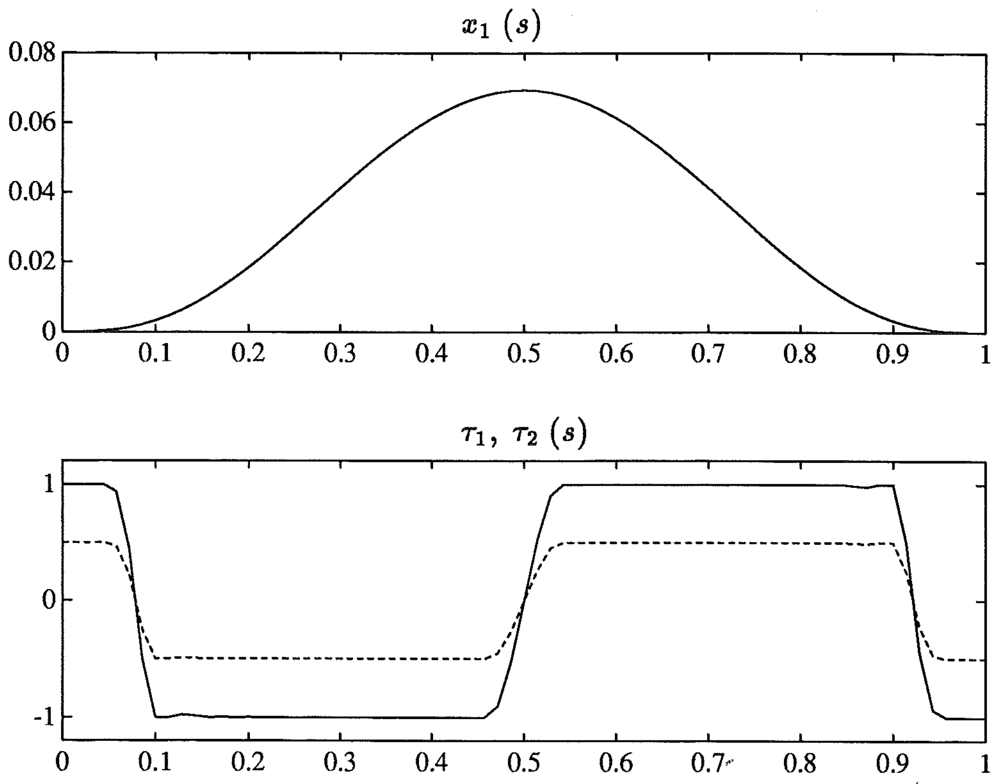


Figure 3.6 The result of using parametric optimization. The upper plot shows the optimal $x_1(s) = \dot{s}^4/4$ as a function of s. The lower plot shows the corresponding torques as functions of s.

the corresponding torques. As can be seen in the lower plot, there are three switches in the torques. Compare with the rigid solution shown in Figure 2.2, where there is only one switch. The traversal time for the exact solution is 3.56 seconds. The traversal time for the rigid solution is given in (2.20) as $t_f = 2\sqrt{2} \approx 2.83$ seconds.

The solution shown in Figure 3.5 was computed by a point to point optimization for joint one using the method described above. The result of this was the exact τ_1, shown in the lower plot in Figure 3.5. The torque $\tau_1(t)$ was then used as input to the system (3.71), which was integrated numerically. The result was the optimal $s(t)$, $\dot{s}(t)$, $\ddot{s}(t)$, $s^{(3)}(t)$, and $s^{(4)}(t)$. The optimal $s(t)$ and $\dot{s}(t)$ were then used to compute the optimal $x_1 = \frac{\dot{s}^4}{4}$, shown in the upper plot in Figure 3.5. The torque τ_2, shown in the lower plot, was computed from (3.70) for $i = 2$, using the optimal $\ddot{s}(t)$ and $s^{(4)}(t)$. Note that for this example, the torques are actually related as $\tau_1 = 2\tau_2$, as can be seen from (3.70).

A parametric solution is shown in Figure 3.6. The number of B-spline coefficients in the B-spline was 40, i.e. the length of the vector β in (3.58) was

40. The s-vector was discretized in 67 points. The degree of the polynomial pieces, i.e. the functions B_i in (3.58) was chosen as five. This means that the function $x_1(s)$ in (3.58) has continous fourth order derivatives [de Boor, 1978]. Using (3.11) and (3.63), it is seen that this implies continuous and differentiable torques, i.e. continuous and differentiable constraints for the optimization.

The upper plot in Figure 3.6 shows the optimal $x_1(s)$. The lower plot shows the corresponding torques. The torques are computed as in (C5'), i.e. only for the discretization points. As can be seen from this plot, the torque switches are smoothened. This was also the case in the rigid example in Chapter 2, shown in Figure 2.3. The traversal time for the parametric solution was computed by assuming constant acceleration between the discretization points in s. The result was the traversal time 3.6 seconds.

The optimization was done on a Sun Sparc Station 2. The computation time was 247 seconds. The computation time for Matlab calculations, i.e. the computation time for the evaluation of the loss function (3.57) and the constraints (C5'), was 177 seconds,

A comparison with the rigid solution is shown in Figure 3.7. The upper plot shows the B-spline velocity profile $v_1(s)$, solid line. This curve is thus related to $x_1(s)$ in Figure 3.6 as $v_1(s) = (4x_1(s))^{1/4}$. The upper plot also shows the exact velocity profile, dashed line. As can be seen in this plot, the B-spline solution agrees well with the exact solution. The dotted line in the upper plot is the rigid velocity profile. This velocity profile is also shown in Figure 2.2. The lower plot shows the limiting torque τ_1. The solid line is the result of the B-spline optimization, also shown in Figure 3.6. The dashed line is the exact flexible torque τ_1, also shown in Figure 3.5. The dotted line in the lower plot is the rigid torque τ_1. As can be seen, the switch in the rigid torque, $s = 0.5$, conicides with second switch in the exact flexible solution.

As can be seen in the upper plot in Figure 3.7, the velocity for the flexible joint model is for most parts of the path larger than the velocity for the rigid model. One might then expect that the traversal time for the flexible solution is smaller. This is however not the case. The numerical values for the traversal time is 3.56 seconds for the exact flexible joint case, and 2.83 seconds for the rigid case. The difference in traversal times can be seen in Figure 3.8 which shows the time optimal path parameters as functions of time. The solid line is the B-spline solution, and the dashed line is the exact solution. As can be seen, these solutions agree well. The dotted line is the rigid solution. As can be seen, the rigid path parameter starts much faster, i.e. even if it has lower velocity for a large s-interval, the fast start makes a large difference in traversal time. The rigid solution is also shown in the upper left plot in Figure 2.1.

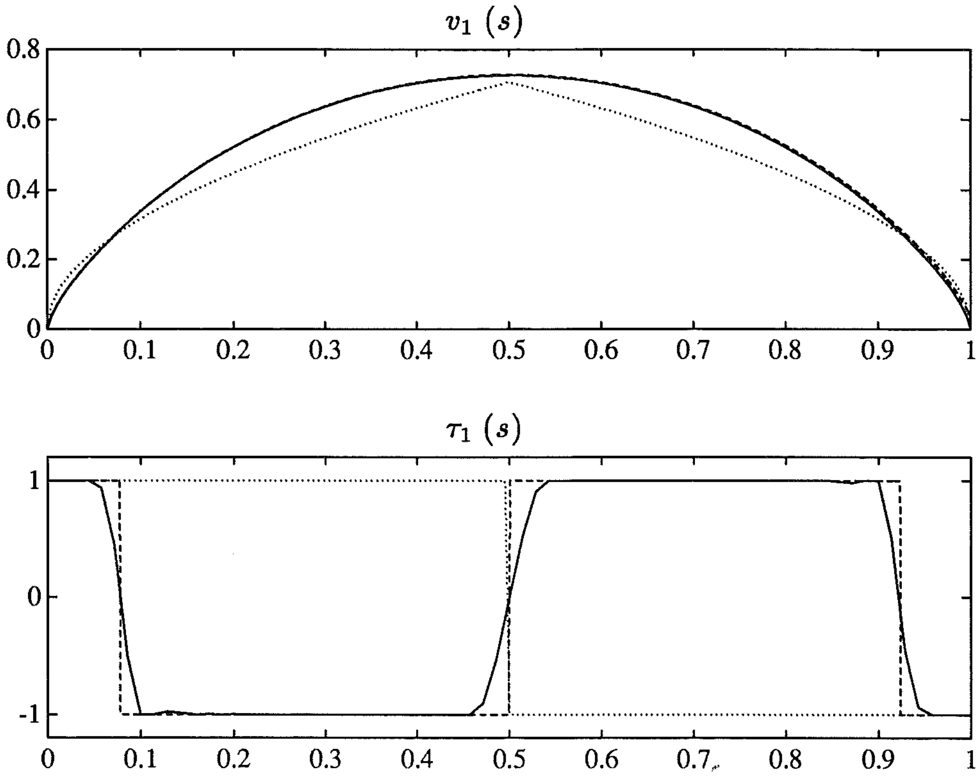


Figure 3.7 A comparison with the rigid solution. The upper plot shows the B-spline velocity profile, solid line, the exact velocity profile, dashed line, and the rigid velocity profile, dotted line. The lower plot shows the limiting torque τ_1. The solid line is the B-spline solution, the dashed line is the exact solution, and the dotted line is the rigid solution.

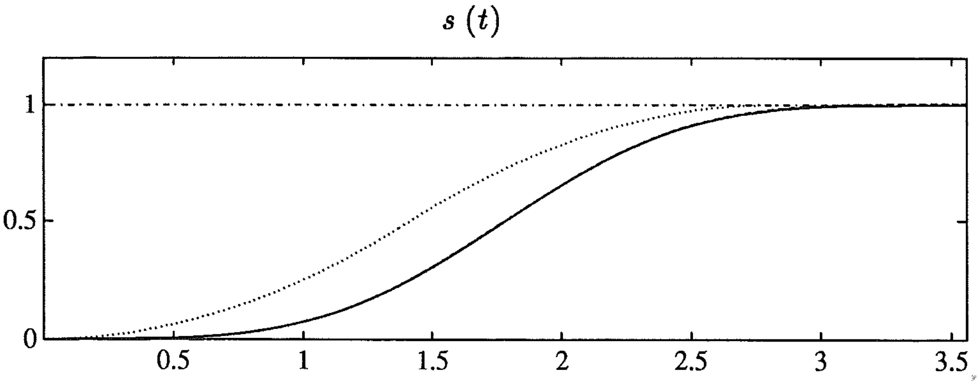


Figure 3.8 The time optimal path parameters as functions of time. The solid line is the B-spline solution, the dashed line is the exact solution, and the dotted line is the rigid solution. The dashed-dotted line is s_f, the maximum value of the path parameter.

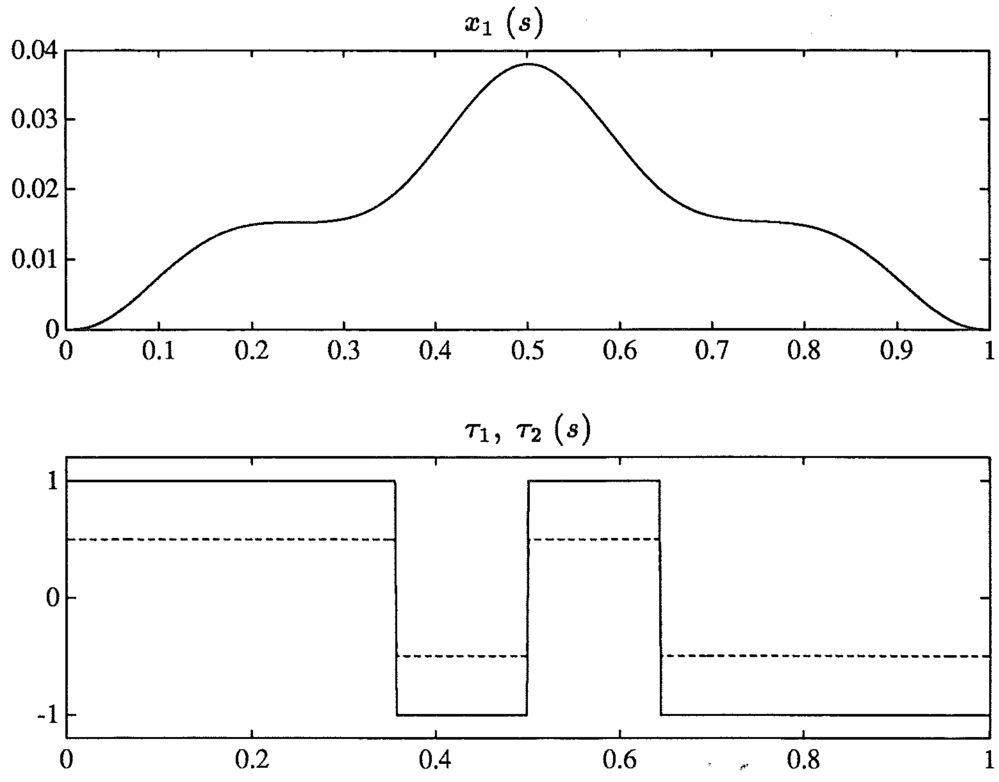


Figure 3.9 The exact time optimal solution. The joint stiffness constants are $k_1 = k_2 = 10$. The upper plot shows the optimal $x_1(s)$. The lower plot shows the corresponding torques τ_1, solid line, and τ_2, dashed line.

Example 3.7
The path is the same as in the previous example. The parameters in (3.65) are chosen as

$$m_1 = m_2 = 0.5, \quad J_1 = J_2 = 0.5, \quad k_1 = k_2 = 10$$

i.e. compared to the previous example, the stiffnes constant is increased from 1 to 10. The torque limits are the same as in the previous example.

The exact solution is shown in Figure 3.9. As can be seen in this plot, the switches in the torques are now closer to the mid point of the path. The number of switches is the same as before. The traversal time for the exact solution is 2.91 seconds.

The result of the B-spline optimization is shown in Figure 3.10. As can be seen, the torque switches are again smoothened. The number of B-spline coefficients in the B-spline was 40. The s-vector was discretized in 69 points. The computed traversal time from the B-spline solution is 2.94 seconds. The computation time was 349 seconds, and the computation time for Matlab

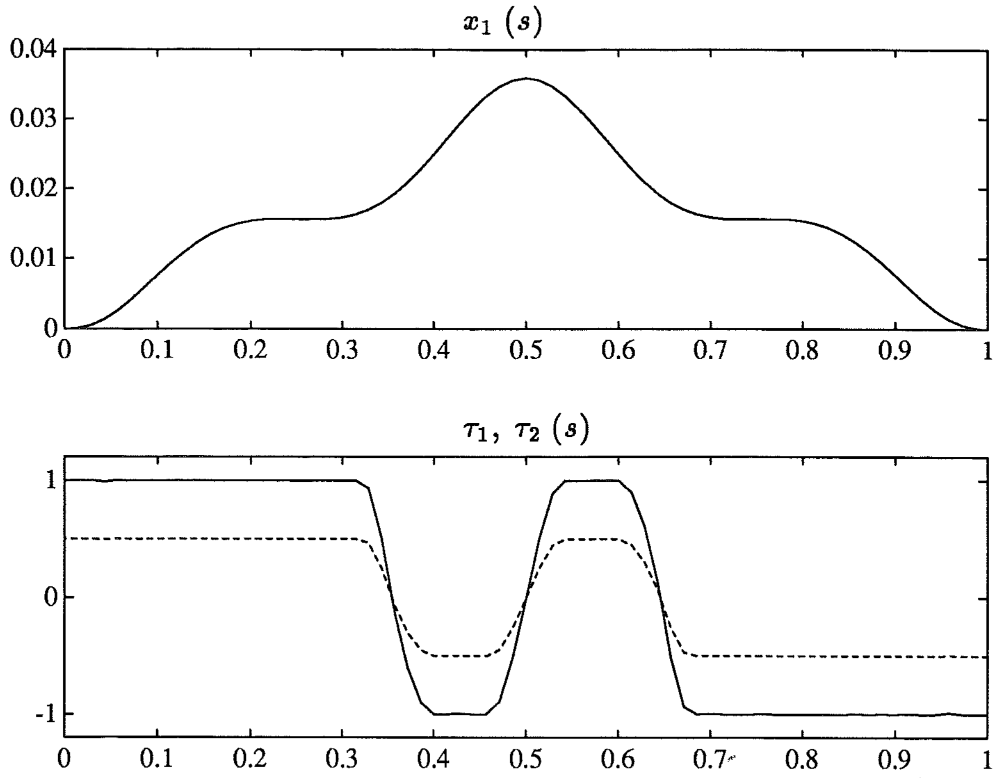


Figure 3.10 The result of using parametric optimization. The upper plot shows the optimal $x_1(s) = \dot{s}^4/4$ as a function of s. The lower plot shows the corresponding torques as functions of s.

calculations was 292 seconds. A comparison with the rigid solution is shown in Figure 3.11. As can be seen in the upper plot, the B-spline solution and the exact solution agree well. For this example, the maximum velocity of the rigid solution is larger than the maximum velocity for the flexible joint solution. This was not the case in the previous example, shown in Figure 3.7.

The difference in traversal times between the rigid solution and the flexible joint solution is now smaller than for the previous example. The numerical values are 2.83 seconds for the rigid solution and 2.91 seconds for the exact flexible solution. The optimal path parameters are shown as functions of time in Figure 3.12. As can be seen in this plot, the difference between the rigid solution and the flexible solution is now smaller. Compare with Figure 3.8. Note that the slope of the rigid solution, the dotted line, is for some time intervals larger than the slope of the flexible solution. This is also seen as a difference in the velocity profiles, shown in the upper plot in Figure 3.11.

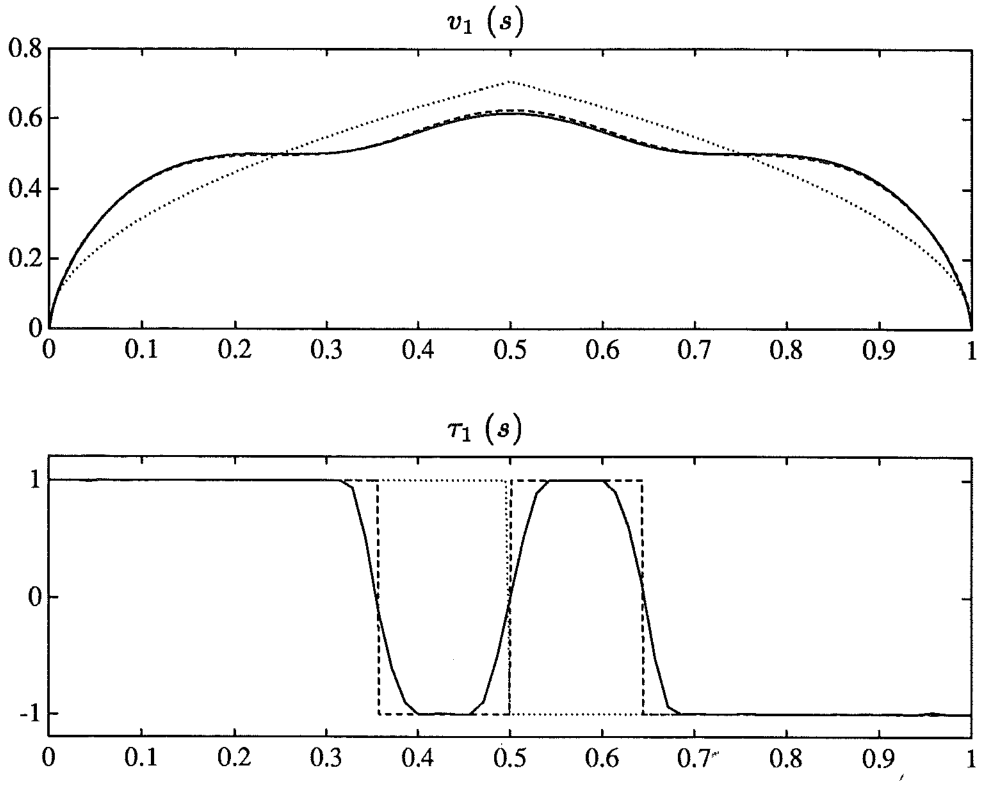


Figure 3.11 A comparison with the rigid solution. The upper plot shows the B-spline velocity profile, solid line, the exact velocity profile, dashed line, and the rigid velocity profile, dotted line. The lower plot shows the limiting torque τ_1. The solid line is the B-spline solution, the dashed line is the exact solution, and the dotted line is the rigid solution.

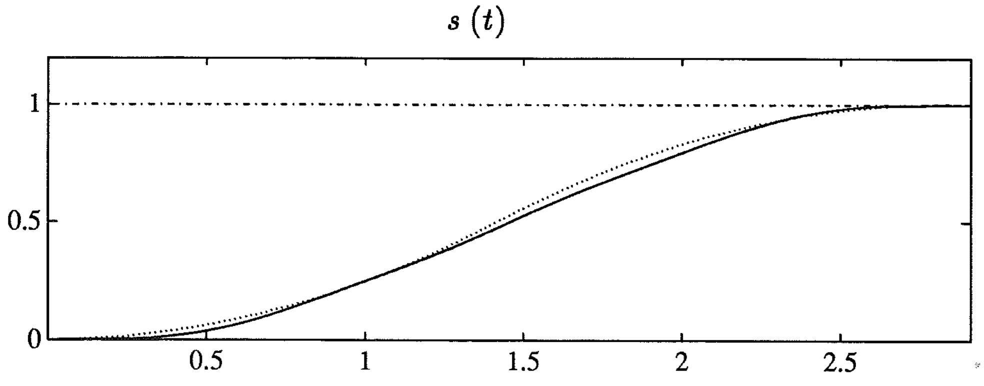


Figure 3.12 The time optimal path parameters as functions of time. The solid line is the B-spline solution, the dashed line is the exact solution, and the dotted line is the rigid solution. The dashed-dotted line is s_f, the maximum value of the path parameter.

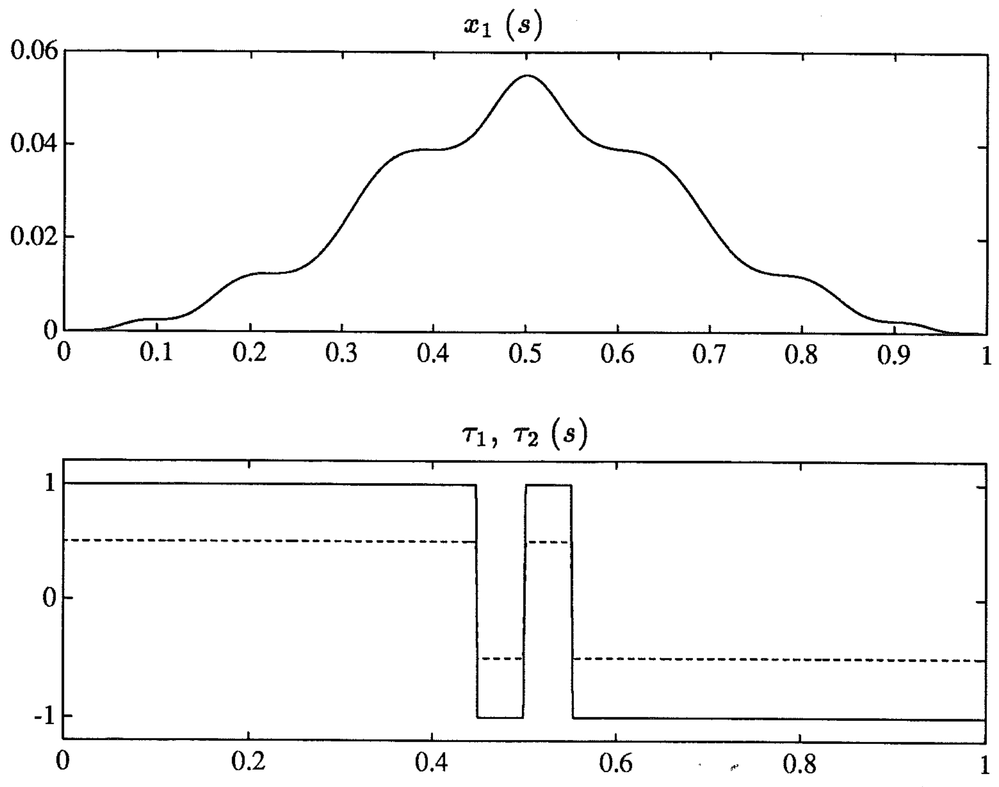


Figure 3.13 The exact time optimal solution. The joint stiffness constants are $k_1 = k_2 = 100$. The upper plot shows the optimal $x_1(s)$. The lower plot shows the corresponding torques τ_1, solid line, and τ_2, dashed line.

EXAMPLE 3.8

In this example, the stiffness constants are further increased to $k_1 = k_2 = 100$. The exact solution is shown in Figure 3.13. As can be seen in the upper plot, the optimal $x_1(s)$ oscillates more than in the previous example, shown in Figure 3.9. Further, the s-interval for the torque switches is reduced. The exact traversal time was 2.84 seconds. The difference to the rigid traversal time 2.83 seconds, is thus further decreased.

The result of the B-spline optimization is shown in Figure 3.14. Note that the torques are now more irregular compared to the previous examples. Numerical problems when the stiffness constants k_i increases can however be expected, since from (3.70) the term in front of $s^{(4)}$ decreases as k_i increases. For this example, the quality of the solution can be increased by increasing the number of B-spline coefficients. This however gives increased computation time.

The computation time for this example was 333 seconds, and the computation for Matlab calculations was 294 seconds. The number of B-spline coefficients in the B-spline was 40. The s-vector was discretized in 65 points.

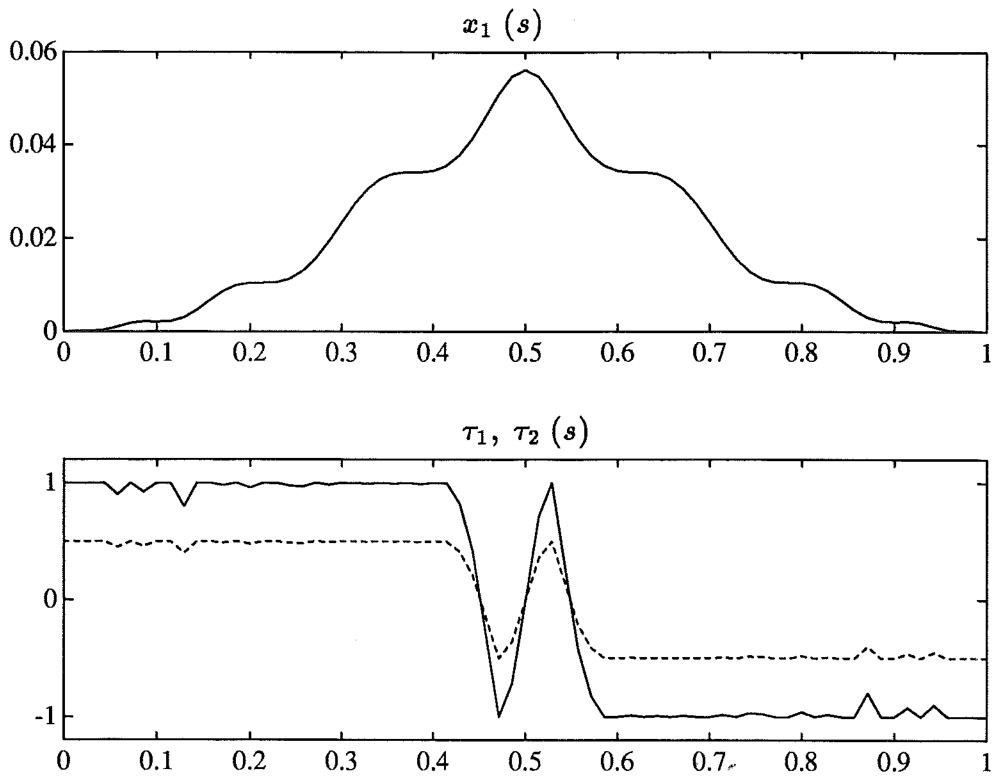


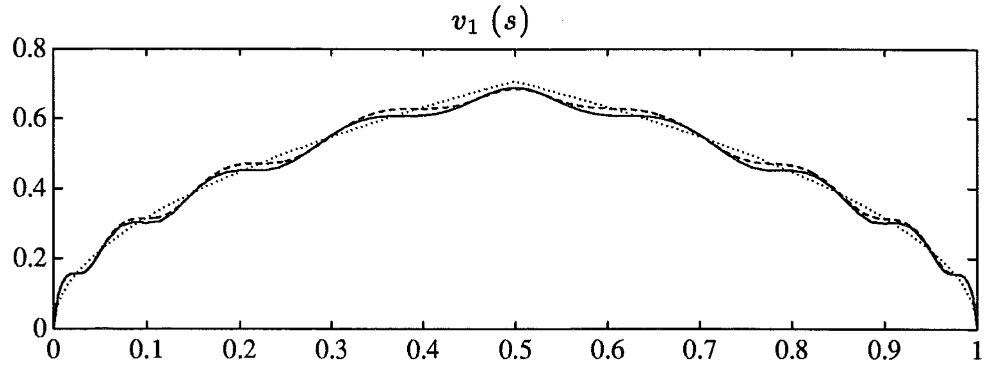
Figure 3.14 The result of using parametric optimization. The upper plot shows the optimal $x_1(s) = \dot{s}^4/4$ as a function of s. The lower plot shows the corresponding torques as functions of s.

The computed traversal time from the B-spline solution is 2.89 seconds. A comparison with the rigid solution is shown in Figure 3.15. As can be seen in the upper plot, the difference between the B-spline solution, solid line, and the exact flexible solution, dashed line, is now slightly larger. This can be expected, since the function $x_1(s)$ now varies more compared to the previous examples, but the number of B-spline coefficients is still 40. Compare with Figures 3.7 and 3.11. Note also that the average distance between the rigid solution and the flexible solution is now smaller than for the previous examples.

The optimal path parameters are shown as functions of time in Figure 3.16. As can be seen in this plot, the difference is reduced. Compare with Figures 3.8 and 3.12.

EXAMPLE 3.9

The path used in this example is the same elliptical path as in Example 3.1. The path is shown in the upper plot in Figure 3.2. The robot model and the torque constraints are the same as in Example 3.7. The rigid solution is



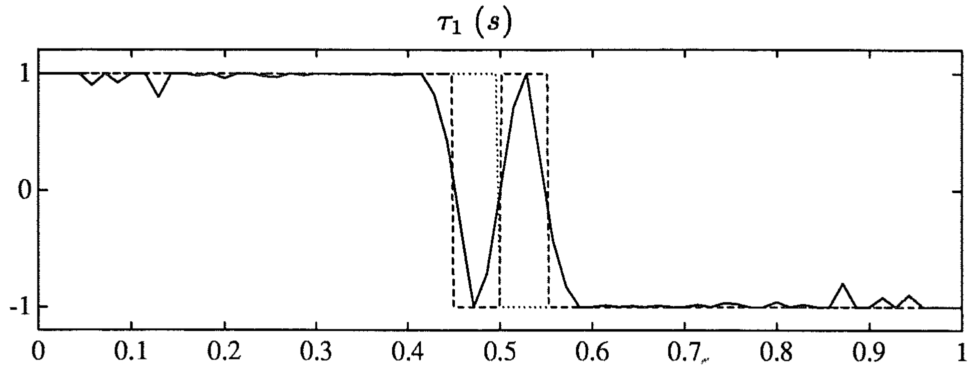


Figure 3.15 A comparison with the rigid solution. The upper plot shows the B-spline velocity profile, solid line, the exact velocity profile, dashed line, and the rigid velocity profile, dotted line. The lower plot shows the limiting torque τ_1. The solid line is the B-spline solution, the dashed line is the exact solution, and the dotted line is the rigid solution.

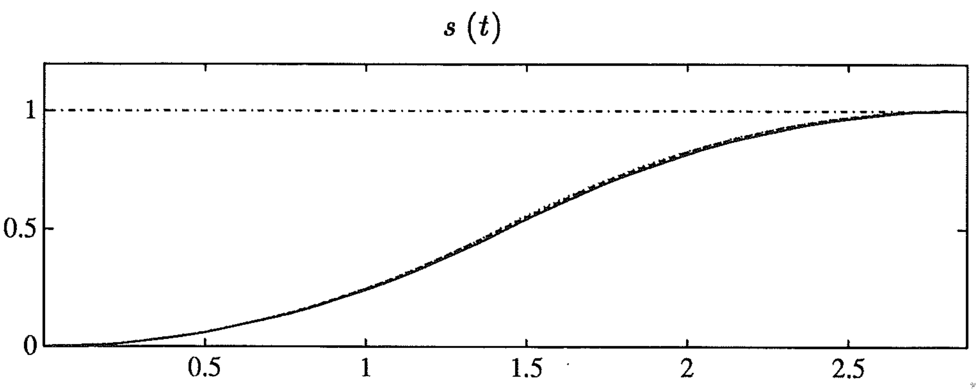


Figure 3.16 The time optimal path parameters as functions of time. The solid line is the B-spline solution, the dashed line is the exact solution, and the dotted line is the rigid solution. The dashed-dotted line is s_f, the maximum value of the path parameter.

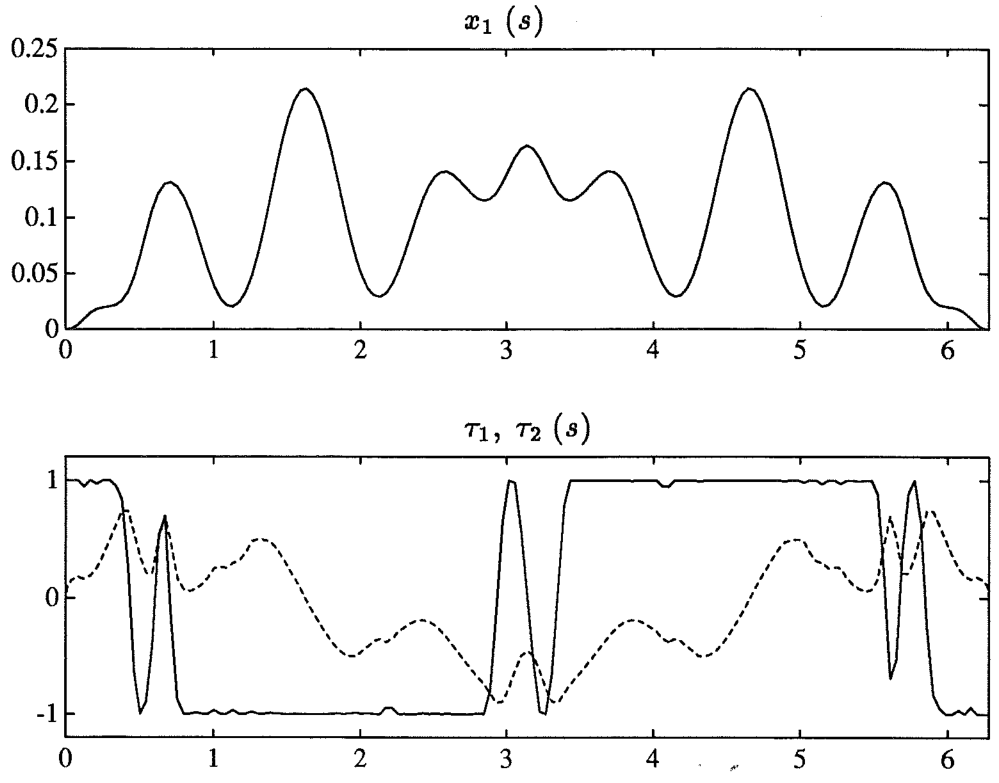


Figure 3.17 The result of using parametric optimization for the elliptical path. The upper plot shows the optimal $x_1(s) = \dot{s}^4/4$ as a function of s. The lower plot shows the corresponding torques as functions of s.

shown in Figure 3.2.

The result of using parametric optimization is shown in Figure 3.17. The number of B-spline coefficients in the B-spline was 80. The s-vector was discretized in 149 points. The computed traversal time from the B-spline result is 9.68 seconds. The rigid traversal time was given in Example 3.1 as 9.66 seconds.

As can be seen in the upper plot, $x_1(s)$ is oscillatory. The lower plot shows that joint one is limiting the motion. The computation time was 2872 seconds, and the computation time for Matlab calculations was 2600 seconds. A comparison with the rigid solution is shown in Figure 3.18. The upper plot shows the B-spline velocity profile, solid line, and the rigid velocity profile, dashed line. As can be seen in this plot, the B-spline velocity profile oscillates around the rigid solution. The same phenomenon is seen in the torque τ_2, shown in the lower plot. The torque τ_1 is however different. It is at the limits, and compared with the rigid torque, each rigid switch is replaced by three flexible switches.

Note that the reason for $x_1(s)$ being oscillatory depends both on the

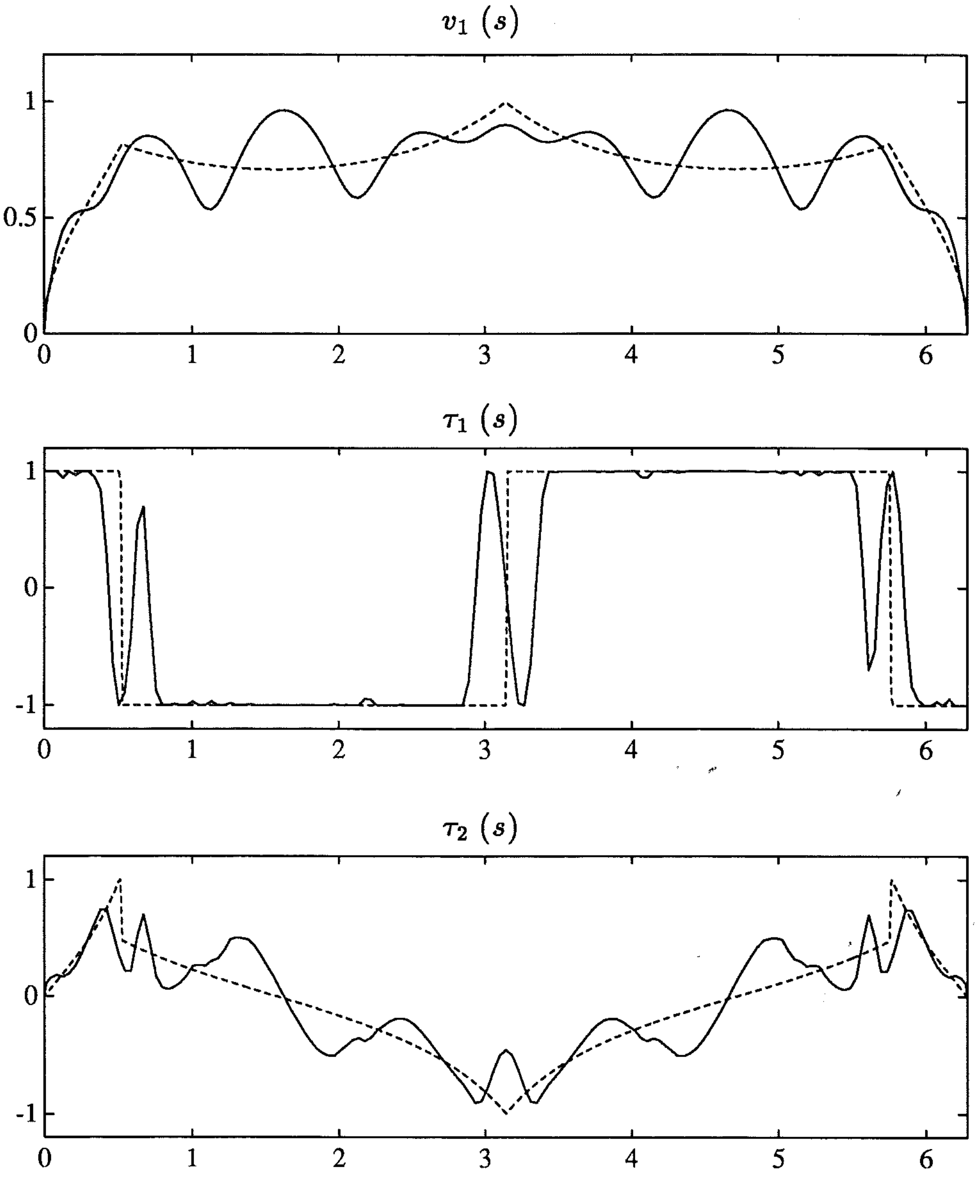


Figure 3.18 A comparison with the rigid solution. The upper plot shows the B-spline velocity profile, solid line, and the rigid velocity profile, dashed line. The mid plot and the lower plot show the torques.

stiffness constant and on the path, in the sense that the number of oscillation periods in the traversal time determines the structure of $x_1(s)$. This is seen by a comparison with Example 3.7, which uses the same robot model as this example, i.e. $k_1 = k_2 = 10$. The solution for Example 3.7, shown in Figure 3.10, is not as oscillatory as the solution for this example, shown in Figure 3.17. Instead, this example is more similar to Example 3.8, compare e.g.

Figures 3.15 and 3.18. The oscillation period is given from (3.71) as

$$T = \sqrt{\frac{J_1 + m_1}{J_1 m_1} k_1}$$

The parameter choice $m_1 = J_1 = 0.5$ then gives $T = \frac{\pi}{\sqrt{k_1}} \approx 0.99$. For this example, the traversal time is 9.68 seconds, i.e. there are 9.8 periods in the traversal time. In Example 3.8, the situation is similar. The parameter choice $k_1 = 100$ gives $T = \frac{\pi}{\sqrt{k_1}} = \frac{\pi}{\sqrt{100}} \approx 0.314$. The traversal time was 2.84 seconds, which gives 9.0 periods in the traversal time. For Example 3.7, where, as in this example, $k_1 = 10$, i.e. $T = 0.99$, the traversal time was 2.91 seconds, which gives 2.9 periods in the traversal time.

EXAMPLE 3.10
This example demonstrates how a "bang-bang" solution can be constructed from a B-spline optimization. The idea is to guess switching points where $s^{(4)}$ switches from maximum to minimum or vice versa. The bang-bang solution is then constructed by integration of (D2) for $p = 4$ with maximum or minimum $s^{(4)}$. The maximum and minimum values of $s^{(4)}$ are computed from (3.11) with $p = 4$, in the same way as limits on $\ddot{s}$ are computed in phase-plane optimization, see (3.23)–(3.26).

The path is the first quadrant of the unit circle, i.e. $f_1(s) = \cos(s)$, $f_2(s) = \sin(s)$, where $0 \leq s \leq \pi/2$. The parameter values in (3.65) are chosen as

$$m_1 = m_2 = 4, \quad J_1 = J_2 = 5, \quad k_1 = k_2 = 30 \tag{3.73}$$

The torque limits are chosen as

$$\tau_i^{min} = -10, \quad \tau_i^{max} = 10, \quad i = 1, 2$$

Figure 3.19 shows the result of B-spline optimization. The traversal time was 2.67 seconds, the number of B-spline coefficients was 60 and the s-vector was discretized in 109 points. The computation time was 1270 seconds, and the computation time for Matlab calculations was 990 seconds. Guessing the switching points $s = 0.51$, $s = 0.78$, and $s = 1.06$ gives the solution shown in Figure 3.20.

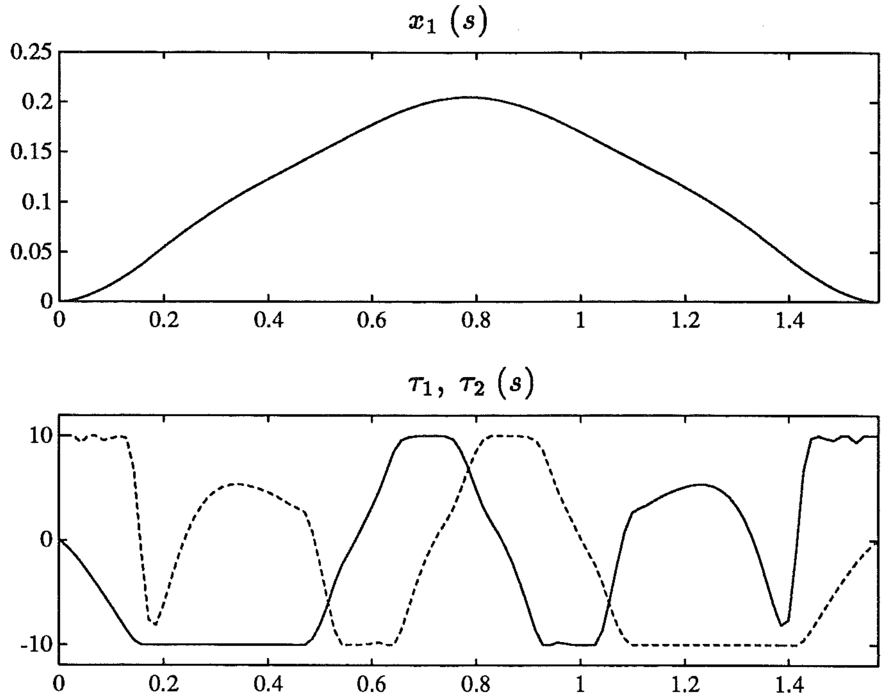


Figure 3.19 The result of B-spline optimization. The path is the first quadrant of the unit circle.

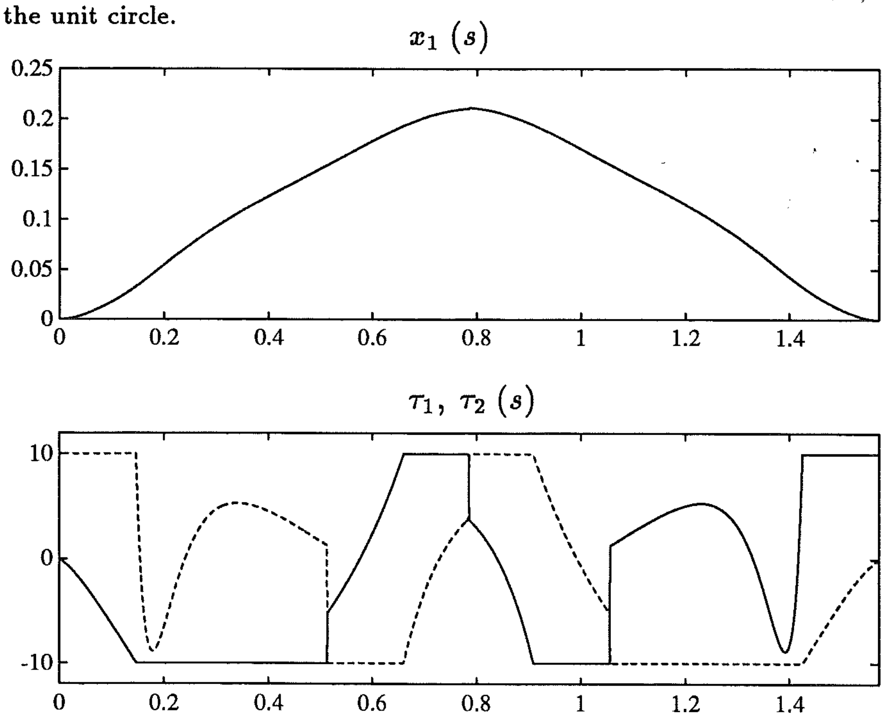


Figure 3.20 The switching solution, constructed by integration with maximum and minimum $s^{(4)}$.

3.8 Conclusions

Methods for specifying minimum time motion along a predefined path have been presented. The rigid body model (3.1) and the simplified flexible joint model (3.2), which is a practically important extension of (3.1), was used. Both models can be written with the torque τ as a function of the vector of joint variables q and its derivatives, equation (3.4). This makes it possible to use a common optimal control formulation (L2,D2,C2,B2) for both the phase plane optimization method [Bobrow *et al.*, 1985; Shin and N.D.McKay, 1985; Pfeiffer and R.Johanni, 1986] and a new method for flexible joint robots.

A review of phase-plane optimization was presented. The velocity and acceleration constraints were discussed, and were also given a geometric interpretation as in Figure 3.1. An optimality result, Theorem 3.1, shows that similar bang-bang properties hold for both rigid and flexible joint robots. With the exception of critical points, the p-th order time derivative of the path parameter, $s^{(p)}$, with $p = 2$ for rigid robots, and $p = 4$ for flexible joint robots, should either be maximized or minimized.

A method for minimum time optimization using parametric optimization was presented. The method is based on polynomial approximation using B-splines. Choosing the function to approximate as $\dot{s}^p/p$ gives the desired boundary conditions on nonzero initial and final $s^{(p)}$, (3.61). The method has been implemented, and numerical examples show the properties of the solution, and also how it differs from the rigid solution. For a given path, the flexible joint velocity profile can vary from the behavior shown in Figure 3.7, to a solution that oscillates around the rigid solution, Figure 3.15. It was also discussed, in Example 3.9, how the solution depends on both the joint stiffness and the path, in the sense that the number of oscillations in the traversal time determine the behavior. It was also shown how a "bang-bang" solution can be constructed, using the optimality result, by finding the switching points in the optimal B-spline, and then integrating with maximum or minimum $s^{(4)}$.

4

Path Velocity Control

From the construction of the minimum time velocity profile by Algorithm 3.1, and also from Theorem 3.1 for $p = 2$, it is seen that the minimum time solution is bang-bang in the sense that the path acceleration is either maximum or minimum, with the possible exception of critical points. The result of this is that at least one torque is always at the limit. This means that there is no margin for the limiting joint to control in the case of disturbances or modeling errors. This may result in large tracking errors, which then gives deviations from the path.

One approach to decrease the sensitivity of the minimum time solution is to reduce the torque limits in the minimum time optimization [Asada and Slotine, 1986, p.170]. The idea is to ensure that there is always torque available for closed loop action by having a prespecified torque margin for the controller. The result will most likely be a conservative motion, where the maximum allowable torque range is not utilized. Further, since the reduced torque limits are prespecified, it may be difficult to guarantee in advance that the actual torques required during motion are not outside the torque limits. Another method in the same spirit is described in [Shin and N.D.McKay, 1987], where bounds on parametric model errors are used for off-line modification of the velocity profile.

The above schemes are open loop, resulting in a nominal motion specification for the robot. A different approach is to use feedback.

In [Slotine and Spong, 1985], feedback is used to obtain a correction to a nominal reference trajectory. The nominal reference trajectory, which is defined on a time interval $[t_0, t_{f_{nom}}]$, is modified when the torques saturate. The modification is done such that the modified reference trajectory is defined on the same time interval $[t_0, t_{f_{nom}}]$. The idea is to maintain the traversal time, while choosing a different path than the nominal.

Path Velocity Control

It is obvious from the minimum time optimization problem that if the path cannot be traversed in the nominal minimum time, the traversal time has to be increased if path following is to be maintained. In path velocity control, this is done via feedback modification of the velocity of the reference trajectory. As in [Slotine and Spong, 1985], a nominal reference trajectory, defined on the time interval $[t_0, t_{f_{nom}}]$, is modified when the torques saturate. However, in path velocity control, the reference trajectory is modified such that the modified reference trajectory defines the same path as the nominal. The modified reference trajectory is thus defined on a time interval $[t_0, t_f]$, where, generally, $t_f > t_{f_{nom}}$.

The nominal reference trajectory is represented by a nominal velocity profile v_1, defined in (2.4), and a nominal acceleration profile v_2, defined in (2.5). The purpose of path velocity control is then to modify the nominal velocity profile, using feedback, such that the modified velocity profile results in motion along the path. The modification is done such that the modified velocity profile does not require more torque than what is available. If the robot controller has enough tracking performance, this will result in small tracking errors, which then will give motion along the path.

The path velocity controller was demonstrated in Chapter 2. A simulation example, shown in Figure 2.6, was used to illustrate how a a time optimal velocity profile may result in path deviations due to modeling errors. It was also illustrated, as shown in Figure 2.9, how the path deviations could be eliminated by using path velocity control.

This chapter gives a detailed presentation of path velocity control, and specific control algorithms are presented in Sections 4.3 and 4.4. The control algorithms are evaluated by simulation in Section 4.5. The focus in this chapter is on rigid robots, where the path velocity controller is a second order dynamic system. It is however possible to obtain also higher order path velocity controllers. This is described in Section 4.6.

Path Velocity Control Algorithms

In the simulation example in Chapter 2, shown in Figure 2.6, the effect of the modeling error was that, during the initial part of the motion, the maximum allowable acceleration for the actual robot model was lower than for the nominal model. This was seen in the simulation in Figure 2.6 where initially the robot was not able to track the reference trajectory. The path velocity controller presented in Section 4.3 aims at solving this problem of inadmissible acceleration by adjusting the acceleration of the reference trajectory such that the resulting acceleration does not exceed the maximum allowable.

Another aspect of the sensitivity of the minimum time solution can be seen from the construction of the minimum time velocity profile by Algorithm 3.1. Consider step 4 in Algorithm 3.1. The backward integration used in this step constructs a curve with minimum acceleration, starting from a point on the maximum velocity curve. The integration, if successful, is continued until an intersection with the previously computed curve occurs. This means that the velocity at the intersection point is the highest possible velocity that results in a velocity profile inside the admissible region. An example of this is seen in Figure 3.2, where the velocity at the point $s = 0.52$ is the highest possible velocity that results in admissible velocity at the point $s = 1.56$. If this velocity profile is used to form a reference trajectory it may happen during motion that the velocity at $s = 0.52$, due to modeling errors, is too high to meet the velocity constraint given by the maximum velocity curve. This means that even if minimum acceleration is used from $s = 0.52$, the velocity at $s = 1.56$ will be too high. As a result of this, the rest of the motion cannot be continued without moving away from the path. The path velocity controller presented in Section 4.4 aims at solving this problem of inadmissible velocity by introducing a scaling of the velocity profile, and then adjust the scaling factor by feedback.

4.1 Controller Parametrization

A robot controller for reference trajectory tracking is assumed available. The robot controller is designed for good tracking performance, disturbance rejection etc., and is kept unchanged in our scheme for path velocity control. The robot controller is however parametrized in the path parameter σ.

The reference trajectory used by the robot controller is denoted $q_r(t) \in \mathbb{R}^n$. The output of the robot controller is the torque $\tau \in \mathbb{R}^n$. We assume that the second time derivative of the reference trajectory is used by the robot controller. The reference trajectory and its derivatives are then given by (2.30).

The controller parametrization is, for a given controller, obtained by substituting $q_r(t) = f(\sigma(t))$ into the controller equation for τ. The parametrization has the form

$$\tau = \beta_1 \ddot{\sigma} + \beta_2 \tag{4.1}$$

This parametrization defines the feedback signals β_1 and β_2, shown in Figure 2.8. The controller parametrization (4.1) is exemplified by some commonly used robot controllers.

EXAMPLE 4.1
The controller (2.26), which is used in the simulation example in Chapter 2, is parametrized as

$$\tau = \beta_1(\sigma)\ddot{\sigma} + \beta_2(\sigma, \dot{\sigma}, q, \dot{q}) \tag{4.2}$$

where

$$\begin{aligned} \beta_1(\sigma) &= \hat{M} f'(\sigma) \\ \beta_2(\sigma, \dot{\sigma}, q, \dot{q}) &= \hat{M}(f''(\sigma)\dot{\sigma}^2 + K_v(f'(\sigma)\dot{\sigma} - \dot{q}) + K_p(f(\sigma) - q)) \end{aligned} \tag{4.3}$$

As can be seen in (4.3), the vector β_2 depends on the measured quantities q and $\dot{q}$, thus giving a feedback path from measured signals to the path velocity controller.

EXAMPLE 4.2
For a rigid robot, described by the model (3.1), typical robot controllers are a feedforward controller with position and velocity feedback, given by

$$\tau = \hat{H}(q_r)\ddot{q}_r + \hat{v}(q_r, \dot{q}_r) + \hat{d}(q_r)\dot{q}_r + \hat{g}(q_r) + K_p e + K_v \dot{e} \tag{4.4}$$

and a computed torque controller [Asada and Slotine, 1986]

$$\tau = \hat{H}(q)(\ddot{q}_r + K_p e + K_v \dot{e}) + \hat{v}(q, \dot{q}) + \hat{d}(q)\dot{q} + \hat{g}(q) \tag{4.5}$$

which both can be parametrized as (4.1). For the feedforward controller (4.4), the result is

$$\begin{aligned} \beta_1(\sigma) =& \hat{H}(f(\sigma))f'(\sigma) \\ \beta_2(\sigma, \dot{\sigma}, q, \dot{q}) =& (\hat{H}(f(\sigma))f''(\sigma) + \hat{v}(f(\sigma), f'(\sigma))\dot{\sigma}^2 \\ &+ (\hat{d}(f(\sigma))f'(\sigma))\dot{\sigma} + \hat{g}(f(\sigma)) + K_p e + K_v \dot{e} \end{aligned} \tag{4.6}$$

and for the computed torque controller (4.5), the result is

$$\begin{aligned} \beta_1(\sigma, q) =& \hat{H}(q)f'(\sigma) \\ \beta_2(\sigma, \dot{\sigma}, q, \dot{q}) =& \hat{H}(q)(f''(\sigma)\dot{\sigma}^2 + K_p e + K_v \dot{e}) + \hat{v}(q, \dot{q}) + \hat{d}(q)\dot{q} + \hat{g}(q) \end{aligned} \tag{4.7}$$

EXAMPLE 4.3
The controller parametrization can also be obtained when there are additional dynamics in the controller. A PID controller, obtained by extending the robot controller (2.26) with integral action is written as

$$\begin{aligned} \tau &= \hat{M}(\ddot{q}_r + K_v(\dot{q}_r - \dot{q}) + K_p(q_r - q) + i) \\ \frac{di}{dt} &= K_i(q_r - q) \end{aligned}$$

The controller parametrization then becomes

$$\tau = \beta_1(\sigma)\ddot{\sigma} + \beta_2(\sigma, \dot{\sigma}, q, \dot{q}, i)$$

where

$$\begin{aligned}
&\beta_1(\sigma) = \hat{M}f'(\sigma) \\
&\beta_2(\sigma, \dot{\sigma}, q, \dot{q}, i) = \hat{M}(f''(\sigma)\dot{\sigma}^2 + K_v(f'(\sigma)\dot{\sigma} - \dot{q}) + K_p(f(\sigma) - q) + i)
\end{aligned}$$

EXAMPLE 4.4
An example of a controller parametrization similar to (4.1) for the case of joint flexibility is given. Equation (3.6), which is obtained by rewriting the robot model (3.2), can be written on the form

$$\tau = JK^{-1}H(q)q^{(4)} + F(q, \dot{q}, \ddot{q}, q^{(3)})$$

A feedback linearizing controller [Spong, 1987] is now obtained by choosing

$$\tau = JK^{-1}H(q)w + F(q, \dot{q}, \ddot{q}, q^{(3)}) \tag{4.8}$$

which gives the linear system $q^{(4)} = w$. A linear controller for w is given by

$$w = q_r^{(4)} + \sum_{i=0}^{3} k_i(q_r^{(i)} - q^{(i)}) \tag{4.9}$$

The parametrization of the controller (4.8) and (4.9) is then obtained by differentiation of $q_r(t) = f(\sigma(t))$ to obtain $\dot{q}_r(t), \ldots, q_r^{(4)}(t)$ as functions of $\dot{\sigma}(t), \ldots, \sigma^{(4)}(t)$. This gives a controller parametrization on the form

$$\tau = \beta_1(\sigma, q)\sigma^{(4)} + \beta_2(\sigma, \dot{\sigma}, \ddot{\sigma}, \sigma^{(3)}, q, \dot{q}, \ddot{q}, q^{(3)}) \tag{4.10}$$

A path velocity controller for this type of parametrization is described in Section 4.6.

4.2 Limits on path acceleration

The controller parametrization (4.1) is used in the path velocity controller as the basis for connecting measurements to the path parameter σ by computing limits on the the path acceleration $\ddot{\sigma}$. This is done in the same way as is done in phase-plane optimization, see (3.23)–(3.26), but with the vectors b_1 and b_2 in the parametrization of the robot dynamics (3.11) replaced by the vectors

β_1 and β_2 in the controller parametrization (4.1). Thus, each torque τ_i is constrained by

$$\tau_i^{min} \leq \quad \tau_i = \beta_{1_i}\ddot{\sigma} + \beta_{2_i} \quad \leq \tau_i^{max}, \quad 1 \leq i \leq n \tag{4.11}$$

Each joint i now gives upper and lower limits on $\ddot{\sigma}$

$$\ddot{\sigma}^i_{max}(\beta_{1_i}, \beta_{2_i}) = \begin{cases} (\tau_i^{max} - \beta_{2_i})/\beta_{1_i}, & \beta_{1_i} > 0 \\ (\tau_i^{min} - \beta_{2_i})/\beta_{1_i}, & \beta_{1_i} < 0 \\ \infty, \quad \beta_{1_i} = 0 \end{cases} \tag{4.12}$$

and

$$\ddot{\sigma}^i_{min}(\beta_{1_i}, \beta_{2_i}) = \begin{cases} (\tau_i^{min} - \beta_{2_i})/\beta_{1_i}, & \beta_{1_i} > 0 \\ (\tau_i^{max} - \beta_{2_i})/\beta_{1_i}, & \beta_{1_i} < 0 \\ -\infty, \quad \beta_{1_i} = 0 \end{cases} \tag{4.13}$$

which gives the limits on $\ddot{\sigma}$ as

$$\begin{aligned} \ddot{\sigma}_{max}(\beta_1, \beta_2) &= \min_i \ddot{\sigma}^i_{max}(\beta_{1_i}, \beta_{2_i}) \\ \ddot{\sigma}_{min}(\beta_1, \beta_2) &= \max_i \ddot{\sigma}^i_{min}(\beta_{1_i}, \beta_{2_i}) \end{aligned} \tag{4.14}$$

The limits (4.14) provide a way to modify the reference trajectory, via modification of $\ddot{\sigma}$, such that the torque limits are not violated. The idea is thus that if the nominal reference trajectory results in inadmissible torques, it is modified by limiting $\ddot{\sigma}$.

Inadmissible limits Note that the limits (4.14) depend on measured quantities through the vectors β_1 and β_2. It can therefore not be guaranteed that $\ddot{\sigma}_{min}(\beta_1, \beta_2) \leq \ddot{\sigma}_{max}(\beta_1, \beta_2)$. This is further discussed below.

4.3 A Basic Algorithm

It was shown in the previous section how limits on $\ddot{\sigma}$ can be computed from the controller parametrization (4.1). This section presents an algorithm for path velocity control, which is obtained by combining the limits on path acceleration with feedback from a nominal velocity profile. The nominal velocity profile is given as the function v_1 in (2.4). The nominal velocity profile can be computed from optimization as in Chapter 3, but can also be specified by a robot operator/programmer. A nominal acceleration profile v_2, defined by (2.5), is also used in the path velocity controller.

The algorithm presented in this section was used in the simulation example in Chapter 2, Figure 2.9, and will also be used in the experimental evaluation in Chapter 5.

Limiting path acceleration

A possible algorithm for path velocity control could be to use the limits (4.14) to limit the nominal path acceleration, expressed as a function of σ by $v_2(\sigma)$, and then let $\ddot{\sigma}$ be the result of the limitation. Define the function $sat(x, x_{min}, x_{max})$ as the function that limits x by the limits x_{min} and x_{max}. The algorithm is then given by the second order dynamic system

$$\begin{aligned}
&\frac{d\sigma}{dt} = \dot{\sigma} \\
&\frac{d\dot{\sigma}}{dt} = \ddot{\sigma} \\
&\ddot{\sigma} = sat(v_2(\sigma), \ddot{\sigma}_{min}(\beta_1, \beta_2), \ddot{\sigma}_{max}(\beta_1, \beta_2))
\end{aligned} \tag{4.15}$$

This algoritm limits the nominal acceleration $v_2(\sigma)$ so that, if $v_2(\sigma)$ is not feasible, i.e. if $v_2(\sigma) < \ddot{\sigma}_{min}(\beta_1, \beta_2)$ or $v_2(\sigma) > \ddot{\sigma}_{max}(\beta_1, \beta_2)$, then $\ddot{\sigma} = \ddot{\sigma}_{min}(\beta_1, \beta_2)$ or $\ddot{\sigma} = \ddot{\sigma}_{max}(\beta_1, \beta_2)$, i.e. the path acceleration $\ddot{\sigma}$ is chosen as the minimum or maximum acceleration that gives all torques inside the limits.

Feedback using the nominal velocity profile

Suppose that the algorithm (4.15) is used for path velocity control and that the motion starts at time $t = t_0$. As long as the limits on $\ddot{\sigma}$ are not active, i.e. as long as $\ddot{\sigma}_{min}(\beta_1, \beta_2) \leq v_2(\sigma) \leq \ddot{\sigma}_{max}(\beta_1, \beta_2)$, we have $\ddot{\sigma} = v_2(\sigma)$. If the initial conditions are chosen as the initial conditions for the nominal path parameter $s(t)$, i.e. $\sigma(t_0) = s(t_0)$ and $\dot{\sigma}(t_0) = \dot{s}(t_0)$, this gives, by (2.5), $\sigma(t) = s(t)$, and hence, by (2.4), $\dot{\sigma} = v_1(\sigma)$, i.e. the algorithm generates the nominal velocity profile.

However, when the limits on $\ddot{\sigma}$ are active, then $\ddot{\sigma}$ saturates and we have $\ddot{\sigma} = \ddot{\sigma}_{max}(\beta_1, \beta_2)$ or $\ddot{\sigma} = \ddot{\sigma}_{min}(\beta_1, \beta_2)$, and the path velocity $\dot{\sigma}$ deviates from the nominal velocity profile $v_1(\sigma)$. This means that when the nominal acceleration is again feasible, i.e. when $\ddot{\sigma}_{min}(\beta_1, \beta_2) \leq v_2(\sigma) \leq \ddot{\sigma}_{max}(\beta_1, \beta_2)$, a situation where $\dot{\sigma} \neq v_1(\sigma)$ and $\ddot{\sigma} = v_2(\sigma)$ can occur. This may then lead to a path velocity $\dot{\sigma}$ moving away from the nominal velocity profile. For example, if $\ddot{\sigma} = v_2(\sigma) = 0$ during a time interval, $\dot{\sigma}$ will be kept constant at a level which depends on the previous limitation of $v_2(\sigma)$, which in turn, since the limits depend on measured quantities, may depend on disturbances acting on the system.

We therefore introduce feedback from the nominal velocity profile $v_1(\sigma)$ to ensure that the path velocity $\dot{\sigma}$ approaches $v_1(\sigma)$ when the limits on $\ddot{\sigma}$ are not active. Choosing the feedback as a nonlinear state feedback $\psi(\sigma, \dot{\sigma})$ gives the path velocity controller

$$\begin{aligned}
&\frac{d\sigma}{dt} = \dot{\sigma} \\
&\frac{d\dot{\sigma}}{dt} = \ddot{\sigma} \\
&u_r = \psi(\sigma, \dot{\sigma}) \\
&\ddot{\sigma} = sat(u_r, \ddot{\sigma}_{min}(\beta_1, \beta_2), \ddot{\sigma}_{max}(\beta_1, \beta_2))
\end{aligned} \tag{4.16}$$

where the variable u_r is an auxiliary variable. The purpose of the feedback $\psi(\sigma, \dot{\sigma})$ in (4.16) is to ensure that $\dot{\sigma}$ approaches the nominal velocity profile $v_1(\sigma)$, when the limits on $\ddot{\sigma}$ are not active, i.e. when $\ddot{\sigma}_{min}(\beta_1, \beta_2) \leq u_r \leq \ddot{\sigma}_{max}(\beta_1, \beta_2)$. The function ψ is specified below, where it is chosen such that the resulting system becomes linear when written as a differential equation with σ as the independent variable. The chosen ψ actually includes also v_1 and v_2, i.e.

$$\psi(\sigma, \dot{\sigma}) = \bar{\psi}(\sigma, \dot{\sigma}, v_1(\sigma), v_2(\sigma))$$

Before giving the specific choice of ψ, some general properties of ψ are discussed.

Time translation The result of having a feedback ψ that makes $\dot{\sigma}$ approach $v_1(\sigma)$ is that if $\dot{\sigma}$ converges to $v_1(\sigma)$, the path velocity controller (4.16) generates a time delayed version of the nominal path parameter. This is seen as follows. Suppose that the path velocity controller (4.16) is used, and that the motion starts at time $t = t_0$. Further, suppose that the limits on $\ddot{\sigma}$ are activated at $t = t_1$. This results in a path velocity $\dot{\sigma}$ that deviates from the nominal velocity profile $v_1(\sigma)$. The limits on $\ddot{\sigma}$ are deactivated at $t = \bar{t}_1$ and the result is that the feedback $\psi(\sigma, \dot{\sigma})$ makes $\dot{\sigma}$ approach the velocity profile $v_1(\sigma)$. At some further time $t = t_\sigma$, the path velocity $\dot{\sigma}$ has converged to the nominal velocity profile, i.e. $\dot{\sigma} = v_1(\sigma)$ for a finite time interval after t_σ. This gives, using (2.6), that $\ddot{\sigma} = v_2(\sigma)$, i.e. after $t = t_\sigma$, the path parameter σ satisfies the differential equation

$$\ddot{\sigma} = v_2(\sigma) \tag{4.17}$$

with the initial values $\sigma(t_\sigma)$ and $\dot{\sigma}(t_\sigma)$. Since the nominal path parameter $s(t)$ also satifies the differential equation (4.17), see (2.5), there must be a time $t = t_s$ such that $\sigma(t_\sigma)$ and $\dot{\sigma}(t_\sigma)$ are related to the nominal path parameter $s(t)$ as

$$\begin{aligned}
\sigma(t_\sigma) &= s(t_s) \\
\dot{\sigma}(t_\sigma) &= \dot{s}(t_s)
\end{aligned}$$

Introduce Δ by $t_\sigma = t_s + \Delta$. This gives that for $t \geq t_\sigma$, the path parameter σ satisfies the differential equation

$$\begin{aligned} \ddot{\sigma} &= v_2(\sigma) \\ \sigma(t_\sigma) &= s(t_\sigma - \Delta) \\ \dot{\sigma}(t_\sigma) &= \dot{s}(t_\sigma - \Delta) \end{aligned} \tag{4.18}$$

This shows, by time invariance of the differential equation $\ddot{x} = v_2(x)$, that, for $t \geq t_\sigma$, $\sigma(t)$ and $s(t)$ are related as $\sigma(t) = s(t - \Delta)$.

The result of using the feedback ψ is thus that if $\dot{\sigma}$ converges to $v_1(\sigma)$, the path velocity controller generates a time delayed version of the nominal path parameter. The path velocity controller thus has the desired effect that when the limits on $\ddot{\sigma}$ are active, i.e. when the torques saturate, the traversal time is increased in order to maintain path tracking. Further, if the nominal path parameter is the result of a minimum time optimization, the use of the path velocity controller (4.16) then gives that when the path velocity $\dot{\sigma}$ has converged to the nominal velocity profile $v_1(\sigma)$, the trajectory generated from that time is a time translated optimal trajectory. This means e.g. that the synchronization between the torques is preserved, i.e. the switching occurs simultaneously as for the nominal case, but at different time instants.

Choice of the feedback ψ

When the limits on $\ddot{\sigma}$ are not active, the path velocity controller (4.16) reduces to the dynamic system

$$\begin{aligned} \frac{d\sigma}{dt} &= \dot{\sigma} \\ \frac{d\dot{\sigma}}{dt} &= \ddot{\sigma} \\ \ddot{\sigma} &= \psi(\sigma, \dot{\sigma}) \end{aligned} \tag{4.19}$$

The feedback ψ is now chosen as

$$\psi(\sigma, \dot{\sigma}) = v_2(\sigma) + \frac{\alpha}{2}(v_1(\sigma)^2 - \dot{\sigma}^2) \tag{4.20}$$

The dynamic system (4.19) then becomes

$$\begin{aligned} \frac{d\sigma}{dt} &= \dot{\sigma} \\ \frac{d\dot{\sigma}}{dt} &= \ddot{\sigma} \\ \ddot{\sigma} &= v_2(\sigma) + \frac{\alpha}{2}(v_1(\sigma)^2 - \dot{\sigma}^2) \end{aligned} \tag{4.21}$$

The path acceleration $\ddot{\sigma}$ in (4.21) is a sum of two terms, of which the first is the nominal acceleration profile $v_2(\sigma)$, and the second is a feedback from the nominal velocity profile $v_1(\sigma)$ and the actual path velocity $\dot{\sigma}$. We now show that the dynamic system (4.21) has the property that $\dot{\sigma}$ approaches the nominal velocity profile. The relation

$$\ddot{\sigma} = \frac{d\dot{\sigma}}{dt} = \frac{d\dot{\sigma}}{d\sigma}\frac{d\sigma}{dt} = \frac{d}{d\sigma}\frac{1}{2}\dot{\sigma}^2 \tag{4.22}$$

is used. Using (2.6), with s replaced by σ, and (4.22), the last equation in (4.21) can be written as

$$\frac{d}{d\sigma}\frac{1}{2}\dot{\sigma}^2 = \frac{d}{d\sigma}\frac{1}{2}v_1(\sigma)^2 + \alpha(\frac{1}{2}v_1(\sigma)^2 - \frac{1}{2}\dot{\sigma}^2)$$

which, introducing the notation

$$x_r(\sigma) = \frac{1}{2}v_1(\sigma)^2, \quad x = \frac{1}{2}\dot{\sigma}^2$$

can be written as

$$\frac{dx}{d\sigma} = \frac{dx_r(\sigma)}{d\sigma} + \alpha(x_r(\sigma) - x) \tag{4.23}$$

This shows that choosing ψ according to (4.20) gives a linear system when the path parameter σ is interpreted as time variable. This means that the chosen feedback has the property that $x \to x_r(\sigma)$ with a σ-time constant $\frac{1}{\alpha}$. Assuming $\dot{\sigma} > 0$, this gives $\dot{\sigma} \to v_1(\sigma)$.

The basic algorithm for path velocity control

The complete path velocity controller (4.16) is now obtained as

$$\begin{aligned} \frac{d\sigma}{dt} &= \dot{\sigma} \\ \frac{d\dot{\sigma}}{dt} &= \ddot{\sigma} \\ u_r &= v_2(\sigma) + \frac{\alpha}{2}(v_1(\sigma)^2 - \dot{\sigma}^2) \\ \ddot{\sigma} &= sat(u_r, \ddot{\sigma}_{min}(\beta_1, \beta_2), \ddot{\sigma}_{max}(\beta_1, \beta_2)) \end{aligned} \tag{4.24}$$

A block diagram of the complete system when this controller is used is shown in Figure 4.1.

The path velocity controller (4.24) is a feedback mechanism for generating the path parameter $\sigma(t)$ and its time derivatives $\dot{\sigma}(t)$ and $\ddot{\sigma}(t)$. The feedback is through the input signals β_1 and β_2, which are functions of measured

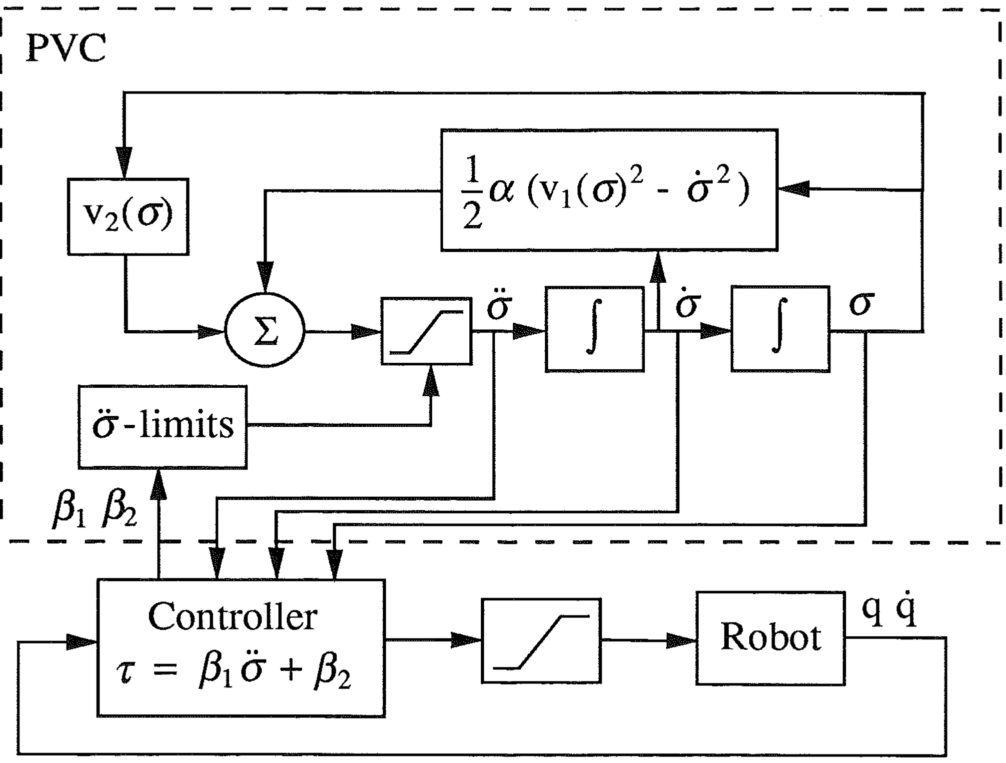


Figure 4.1 The complete system when the basic algorithm for path velocity control is used

quantities, as shown e.g in (4.3) and (4.7). There is also internal feedback, defined by u_r in (4.24).

The path velocity controller uses the nominal velocity profile v_1 in the form $v_1(\sigma)$, i.e. as a nonlinear function of one of the states in the system. This means that the "reference value" $v_1(\sigma)$ for the path velocity controller is not a given time function. It is instead a function that gives the nominal velocity as a function of the current position. The purpose of the internal feedback is then to achieve the nominal velocity for the current position, i.e. to achieve $\dot{\sigma} \to v_1(\sigma)$. This means that if the reference trajectory is behind the nominal reference trajectory, the path velocity controller does not try to catch up, i.e. it does not try to achieve $\sigma(t) \to s(t)$. Instead, as shown in (4.18), it tries to achieve $\sigma(t) \to s(t-\Delta)$.

Inadmissible limits Since the limits on $\ddot{\sigma}$ (4.14) depend on measured quantities, it can happen that $\ddot{\sigma}_{min}(\beta_1,\beta_2) > \ddot{\sigma}_{max}(\beta_1,\beta_2)$. In our implementation, we choose to ignore the result of the computation (4.11)–(4.14) if this occurs. This means that if $\ddot{\sigma}_{min}(\beta_1,\beta_2) > \ddot{\sigma}_{max}(\beta_1,\beta_2)$, we use $\ddot{\sigma} = u_r$ in (4.24). This gives a stable generation of the reference trajectory since, when the limits are not active, we get a linear system in σ (4.23). Further, there are no parameters to tune.

Computational Complexity An advantage with the path velocity controller (4.24) is the limited amount of computation needed. Using the path velocity controller requires that the robot controller is written in the form (4.1). The computation of β_1 and β_2 is of the same complexity as the computations done in the unparametrized controller, compare e.g. (4.6) and (4.4). The path velocity controller then adds a second order dynamic system (4.24), which includes the computation of the limits on $\ddot{\sigma}$ according to the computational procedure (4.11)–(4.14). In conclusion, the computations are of the same complexity as for the case of using only the robot controller.

Also the memory requirements are of the same order as for conventional schemes. The nominal velocity profile and acceleration profile are stored as functions of σ in $v_1(\sigma)$ and $v_2(\sigma)$, and the path and its derivatives are stored as $f(\sigma)$, $f'(\sigma)$, and $f''(\sigma)$. If a conventional scheme is used, the reference trajectory is instead stored as functions of time t in $q_r(t), \dot{q}_r(t), \ddot{q}_r(t)$.

4.4 Velocity Profile Scaling

A time optimal velocity profile may touch the maximum velocity curve for some s-values. This is seen, e.g. in the mid plot in Figure 3.2. If the computation of such a velocity profile is based on an erroneous model, it may happen that for s-values close to the touching points, the velocity profile computed by the minimum time optimization is actually above the true maximum velocity curve.

One possibility to prevent the velocity from being too high is to introduce a constant scaling of the nominal velocity profile. Choosing a scaling factor less than one then gives a reduction of the velocity profile, and hence a margin to the maximum velocity curve.

A continuation of this idea is to try to estimate a suitable scaling factor during motion. A path velocity controller, where this is done is presented below. The control algorithm is an extension of the basic algorithm (4.24). The algorithm modifies the scaling factor using feedback. The scaling factor is then used in the basic algorithm as if it was constant.

Constant scaling in the basic algorithm

A constant scaling of the velocity profile is introduced in the basic algorithm (4.24) by replacing the nominal velocity profile v_1 with a scaled nominal velocity profile γv_1, where γ is a constant scaling factor.

Constant time scaling A constant scaling of the nominal velocity profile is equivalent to a constant time scaling of the nominal path parameter $s(t)$. This is seen as follows. Introduce a time scaled nominal path parameter by

$\bar{s}(t) = s(\gamma t)$. This gives

$$\dot{\bar{s}}(t) = \frac{d}{dt}s(\gamma t) = \gamma \dot{s}(\gamma t) = \gamma v_1(s(\gamma t)) = \gamma v_1(\bar{s}(t))$$

which shows that the velocity profile of the time scaled path parameter is γv_1. Further differentiation gives

$$\ddot{\bar{s}}(t) = \frac{d}{dt}\gamma \dot{s}(\gamma t) = \gamma^2 \ddot{s}(\gamma t) = \gamma^2 v_2(s(\gamma t)) = \gamma^2 v_2(\bar{s}(t)) \tag{4.25}$$

which shows that the acceleration profile of the time scaled path parameter is $\gamma^2 v_2$.

Path Velocity Controller A path velocity controller with constant velocity profile scaling is now obtained from (4.24) by replacing v_1 with γv_1, and replacing v_2 with $\gamma^2 v_2$. This gives the controller

$$\begin{aligned}
\frac{d\sigma}{dt} &= \dot{\sigma} \\
\frac{d\dot{\sigma}}{dt} &= \ddot{\sigma} \\
u_r &= \gamma^2 v_2(\sigma) + \frac{\alpha}{2}(\gamma^2 v_1(\sigma)^2 - \dot{\sigma}^2) \\
\ddot{\sigma} &= sat(u_r, \ddot{\sigma}_{min}(\beta_1, \beta_2), \ddot{\sigma}_{max}(\beta_1, \beta_2))
\end{aligned} \tag{4.26}$$

The path velocity controller (4.26) has the property that when the limits on $\ddot{\sigma}$ are not active, $\dot{\sigma}$ approaches $\gamma v_1(\sigma)$. A similar analysis as resulted in (4.18) gives a relation between $\sigma(t)$ and $s(t)$.

Suppose that the algorithm (4.26) is used, and this results in $\dot{\sigma} = \gamma v_1(\sigma)$ in a time interval after e.g. $t = t_\sigma$. This gives, using (2.6), that $\ddot{\sigma} = \gamma^2 v_2(\sigma)$, i.e. after $t = t_\sigma$, the path parameter σ satisfies the differential equation $\ddot{\sigma} = \gamma^2 v_2(\sigma)$ with the initial values $\sigma(t_\sigma)$ and $\dot{\sigma}(t_\sigma)$. Since, from (4.25), the time scaled nominal path parameter $\bar{s}(t)$ also satisfies this differential equation, there must exist Δ such that

$$\begin{aligned}
\ddot{\sigma} &= \gamma^2 v_2(\sigma) \\
\sigma(t_\sigma) &= \bar{s}(t_\sigma - \Delta) \\
\dot{\sigma}(t_\sigma) &= \dot{\bar{s}}(t_\sigma - \Delta)
\end{aligned}$$

which shows that $\sigma(t)$ and $\bar{s}(t)$ are related as $\sigma(t) = \bar{s}(t - \Delta)$. Using the relation $\bar{s}(t) = s(\gamma t)$, this gives $\sigma(t) = s(\gamma(t - \Delta))$, i.e. for $t \geq t_\sigma$, the path velocity controller (4.26) generates a time scaled, time translated version of the nominal path parameter.

Feedback modification of γ

A feedback scheme for adjusting the scaling factor γ in (4.26) is given. The idea is to obtain a scaling of the nominal velocity profile such that the scaled nominal velocity profile is not above the true maximum velocity curve, which may be unknown due to modeling errors. The algorithm for doing this is as follows. Suppose that, during an acceleration phase, i.e. when the path acceleration is nominally maximum, the limits on $\ddot{\sigma}$ are activated due to the nominal acceleration being too high. This is is treated as an indication that the velocity may be too high further along the path, and the algorithm therefore decreases γ. The reduction of γ is done such that the scaled nominal velocity profile $\gamma v_1(\sigma)$ approaches the actual velocity profile $\dot{\sigma}$, when the upper limit on $\ddot{\sigma}$ is active, i.e. when $\ddot{\sigma}$ is maximum. The specific feedback used is a filtering of the actual scaling factor obtained during motion. This scaling factor, denoted $\hat{\gamma}$, thus satisfies $\dot{\sigma} = \hat{\gamma} v_1(\sigma)$. A first order low pass filtering of $\hat{\gamma}$ is written in the σ-time scale as

$$\frac{d\gamma}{d\sigma} = k(\hat{\gamma} - \gamma) \tag{4.27}$$

The result of using the filter is that γ approaches $\hat{\gamma}$ with a σ-time constant $\frac{1}{k}$. The σ-time scale facilitates tuning of the parameter k. The value of k is e.g. chosen, by inspection of the nominal velocity profile, so that $\frac{1}{k}$ is smaller than the length of the first acceleration interval. If the initial acceleration is too high, then γ will adjust to the actual velocity profile during the first acceleration interval. The filter (4.27) is implemented, using the relations

$$\frac{d\gamma}{d\sigma}\dot{\sigma} = \frac{d\gamma}{dt}, \quad \dot{\sigma} = \hat{\gamma} v_1(\sigma)$$

as

$$\frac{d\gamma}{dt} = \dot{\sigma} k(\dot{\sigma}/v_1(\sigma) - \gamma) \tag{4.28}$$

Path velocity control including velocity profile scaling

A path velocity controller with feedback modification of the scaling factor is now obtained by combining the path velocity controller (4.26) with adjustment of γ according to (4.28). The complete controller becomes

$$\begin{aligned}
\frac{d\sigma}{dt} &= \dot{\sigma} \\
\frac{d\dot{\sigma}}{dt} &= \ddot{\sigma} \\
u_r &= \gamma^2 v_2(\sigma) + \frac{\alpha}{2}(\gamma^2 v_1(\sigma)^2 - \dot{\sigma}^2) \\
\ddot{\sigma} &= sat(u_r, \ddot{\sigma}_{min}(\beta_1, \beta_2), \ddot{\sigma}_{max}(\beta_1, \beta_2)) \\
\dot{\gamma} &= \begin{cases} \dot{\sigma}k(\dot{\sigma}/v_1(\sigma) - \gamma), & \gamma v_1(\sigma) \geq \dot{\sigma} \\ 0, & \gamma v_1(\sigma) < \dot{\sigma} \end{cases}
\end{aligned} \tag{4.29}$$

where (4.28) has been extended with the condition that γ is only modified when the actual velocity $\dot{\sigma}$ is lower than the scaled nominal velocity profile $\gamma v_1(\sigma)$. The algorithm used in the simulations also includes logic such that γ only is adjusted when the limits on $\ddot{\sigma}$ are active.

Analysis of Constant Scaling

The algorithm (4.29) tries to estimate a suitable scaling factor. This subsection gives an approximate analysis in order to investigate when this is possible, i.e. when the scaling factor obtained during acceleration can be used also during the following deceleration without violating the torque limits. The analysis also gives insight into what type of model errors that can be handled by the path velocity controllers (4.24) and (4.29).

The idea is to derive requirements on a scaling factor that results in admissible torques. This problem was also treated by [Hollerbach, 1984], where, given a nominal trajectory, a scaling factor is chosen such that the scaled trajectory results in admissible torques. The robot model used in [Hollerbach, 1984] was the rigid model (3.1).

We will here treat the specific problem of choosing a scaling factor that results in admissible torques when the nominal trajectory is represented by a nominal velocity profile, computed from a minimum time optimization as in Algorithm 3.1. The analysis is based on the structure of the minimum time solution, and gives requirements on the scaling factor, that are functions of real and estimated parameters in the robot model.

A simplified robot model is used. The model is a decoupled linear model on the form

$$m_i \ddot{q}_i + d_{f_i} sign(\dot{q}_i) = \tau_i \tag{4.30}$$

where $1 \leq i \leq n$, the number of joints. The torque constraints are given by

$$\tau_i^{min} \leq \tau_i \leq \tau_i^{max}$$

Moving the robot (4.30) along a path $f(s)$ gives, using $q_i = f_i(s)$,

$$m_i(f_i'(s)\ddot{s} + f_i''(s)\dot{s}^2) + d_{f_i} sign(f_i'(s)\dot{s}) = \tau_i \tag{4.31}$$

Since $\dot{s} > 0$, the friction term $d_{f_i} sign(f_i'(s)\dot{s})$ can be written as

$$\delta_i(s) = d_{f_i} sign(f_i'(s)\dot{s}) = d_{f_i} sign(f_i'(s)) \tag{4.32}$$

The model (4.31) can now be written as

$$m_i(f_i'(s)\ddot{s} + f_i''(s)\dot{s}^2) + \delta_i(s) = \tau_i \tag{4.33}$$

Using (2.4) and (2.5), (4.33) can be written as

$$m_i(f_i'(s)v_2(s) + f_i''(s)v_1(s)^2) + \delta_i(s) = \tau_i \tag{4.34}$$

Suppose now that the velocity profile v_1 is a constant scaling of a nominal velocity profile v_{1_n}, i.e. $v_1(s) = \gamma v_{1_n}(s)$, and $v_2(s) = \gamma^2 v_{2_n}(s)$. Equation (4.34) is then written as

$$m_i(f_i'(s)\gamma^2 v_{2_n}(s) + f_i''(s)\gamma^2 v_{1_n}(s)^2) + \delta_i(s) = \tau_i \tag{4.35}$$

This shows how a constant scaling of the velocity profile, or, equivalently, how a constant time scaling, influences the torques. In order to have admissible torques, the scaling factor γ must be chosen so that

$$\tau_i^{min} \le \tau_i = m_i(f_i'(s)\gamma^2 v_{2_n}(s) + f_i''(s)\gamma^2 v_{1_n}(s)^2) + \delta_i(s) \le \tau_i^{max} \tag{4.36}$$

for all s and i, $1 \le i \le n$.

Requirements on γ Suppose that, for a given s, joint j is the limiting joint, and that the limitation is max, i.e. if γ is chosen so that $\tau_j \le \tau_j^{max}$, all other torques will also be inside the torque limits. Using (4.36), this gives that the scaling factor γ should satisfy

$$m_j\gamma^2(f_j'(s)v_{2_n}(s) + f_j''(s)v_{1_n}(s)^2) + \delta_j(s) \le \tau_j^{max} \tag{4.37}$$

Suppose now that the nominal velocity profile $v_{1_n}(s)$ is computed from a minimum time optimization, based on the model

$$\hat{m}_i\ddot{q}_i + \hat{d}_{f_i} sign(\dot{q}_i) = \tau_i, \quad 1 \le i \le n \tag{4.38}$$

From (4.30) and (4.34), it is seen that (4.38) can be written as

$$\hat{m}_i(f_i'(s)v_{2_n}(s) + f_i''(s)v_{1_n}(s)^2) + \hat{\delta}_i(s) = \tau_i$$

where

$$\hat{\delta}_i(s) = \hat{d}_{f_i} sign(f'(s)) \tag{4.39}$$

Further, suppose that joint j is limiting also in the minimum time optimization. Since the nominal velocity profile $v_{1_n}(s)$ is a minimum time velocity profile for the model (4.38), the limiting joint j is at the limit, i.e.

$$\hat{m}_j(f_j'(s)v_{2_n}(s) + f_j''(s)v_{1_n}(s)^2) + \hat{\delta}_j(s) = \tau_j^{max} \tag{4.40}$$

A reasonable assumption on the friction level is that both the real friction level $\delta_j(s)$ and the estimated friction level $\hat{\delta}_j(s)$ are smaller than the maximum allowable torque, i.e. $\tau_j^{max} - \delta_j(s) > 0$, and $\tau_j^{max} - \hat{\delta}_j(s) > 0$. Inserting (4.40) into (4.37) then gives

$$\gamma^2 \leq \frac{\hat{m}_j}{m_j} \frac{\tau_j^{max} - \delta_j(s)}{\tau_j^{max} - \hat{\delta}_j(s)}$$

If the limitation instead is min, i.e. if γ is chosen so that $\tau_j \geq \tau_j^{min}$, then all other torques are inside the torque limits, the inequality (4.37) is replaced by

$$m_j\gamma^2(f_j'(s)v_{2_n}(s) + f_j''(s)v_{1_n}(s)^2) + \delta_j(s) \geq \tau_j^{min} \tag{4.41}$$

and (4.40) is replaced by

$$\hat{m}_j(f_j'(s)v_{2_n}(s) + f_j''(s)v_{1_n}(s)^2) + \hat{\delta}_j(s) = \tau_j^{min} \tag{4.42}$$

Inserting (4.42) into (4.41), under the assumption that $\tau_j^{min} - \delta_j(s) < 0$, and $\tau_j^{min} - \hat{\delta}_j(s) < 0$, gives

$$\gamma^2 \leq \frac{\hat{m}_j}{m_j} \frac{\tau_j^{min} - \delta_j(s)}{\tau_j^{min} - \hat{\delta}_j(s)}$$

The requirements on γ that result in admissible torques for a given s, can now be summarized as

$$\gamma^2 \leq \frac{\hat{m}_j}{m_j} \frac{\tau_j^{m} - \delta_j(s)}{\tau_j^{m} - \hat{\delta}_j(s)} \tag{4.43}$$

where $\tau_j^m = \tau_j^{max}$ or $\tau_j^m = \tau_j^{min}$.

EXAMPLE 4.5—Friction errors

Requirements on γ when friction is underestimated are given. Suppose that the nominal velocity profile is given by Figure 3.2. From the figure it is seen that joint one is limiting during the first acceleration and deceleration phase.

The velocity profile in Figure 3.2 was computed using the model (2.1), with the parameters $m_1 = m_2 = 1$. With the notation used here, these are the parameters $\hat{m}_1$ and $\hat{m}_2$ in (4.38). In (4.43), we thus have $j = 1$ and $\hat{m}_1 = 1$. For the acceleration phase we have $\tau_1^m = \tau_1^{max} = 1$, and for the deceleration phase we have $\tau_1^m = \tau_1^{min} = -1$. It remains to compute $\hat{\delta}_1(s)$ and $\delta_1(s)$ in (4.43).

From Figure 3.2, it is seen that the first deceleration phase ends at $s = \frac{\pi}{2}$. From (3.32), we get $f_1'(s) = \cos(s) \geq 0$ for $0 \leq s \leq \frac{\pi}{2}$, i.e. $f_1'(s) \geq 0$ both for the acceleration phase and for the deceleration phase. The model (2.1) contains no friction, which gives $\hat{d}_{f_1} = 0$ in (4.38). Using (4.39), this gives $\hat{\delta}_1(s) = 0$. Assuming $d_{f_1} \geq 0$, we get, using (4.32), $\delta_1(s) = d_{f_1} \geq 0$, both for the acceleration phase and for the deceleration phase. The requirement on γ (4.43) during acceleration then becomes

$$\gamma^2 \leq \gamma_a^2 = \frac{1}{m_1}(1 - d_{f_1}) \tag{4.44}$$

For the deceleration phase we get, using $\tau_1^m = \tau_1^{min} = -1$,

$$\gamma^2 \leq \gamma_d^2 = \frac{1}{m_1}(1 + d_{f_1}) \tag{4.45}$$

Suppose now that the mass parameter is known and that the friction is underestimated, i.e. $m_1 = \hat{m}_1 = 1$ and $d_{f_1} > \hat{d}_{f_1} = 0$. This means, as is seen from (4.44), that γ_a, the required γ during acceleration satisfies $\gamma_a < 1$. From (4.45), it is seen that γ_d, the required γ during deceleration satisfies $\gamma_d > 1$.

This indicates, by the following reasoning, that the basic algorithm (4.24) would suffice. Since the required scaling during acceleration is $\gamma_a < 1$, the nominal velocity during acceleration is too high. Using the basic algorithm (4.24) would then lead to activation of the upper limit on $\ddot{\sigma}$, which then reduces the velocity. Further, since the requirement on γ during the deceleration is $\gamma_d > 1$, using the basic algorithm where $\dot{\sigma} \to v_1(\sigma)$ will result in admissible torques also during the deceleration. In fact, since the algorithm uses $\gamma = 1$, the motion during deceleration would be conservative, i.e. the maximum allowable torque range is not utilized.

Note that if there are both mass errors and friction errors, the requirement during the deceleration (4.45) may result in either $\gamma_d < 1$ or $\gamma_d \geq 1$, depending on the actual parameter values m_1 and d_{f_1}. For the latter case, the basic algorithm can be used. □

EXAMPLE 4.6—Mass errors
Suppose now that the model error is in the mass parameter and that there is no friction, i.e. $m_1 \neq 1$ and $d_{f_1} = 0$. From (4.44) and (4.45), we then get the same requirement

$$\gamma^2 \leq \frac{1}{m_1} \tag{4.46}$$

for the deceleration and the acceleration. This indicates that the algorithm with velocity profile scaling (4.29) can be used, since, if the γ obtained during acceleration satisfies $\gamma^2 \leq \frac{1}{m_1}$, and this γ also is used during the following deceleration, the requirement (4.46) is satisfied for both the acceleration and the deceleration. However, if the γ obtained during acceleration is slightly larger than the required, the velocity during the deceleration phase will be too high. Note however that if the actual robot model has a small amount of friction, i.e. $d_{f_1} = \epsilon > 0$, the requirement on acceleration (4.44) becomes

$$\gamma^2 \leq \gamma_a^2 = \frac{1}{m_1}(1 - \epsilon)$$

and the requirement on deceleration (4.45) becomes

$$\gamma^2 \leq \gamma_d^2 = \frac{1}{m_1}(1 + \epsilon)$$

This shows that $\gamma_a < \gamma_d$. Hence, if the result of adjusting γ during acceleration is $\gamma = \gamma_a$, then this γ can be used also during the following deceleration, which indicates that it is possible to use the algorithm (4.29). □

4.5 Simulations

The performance of the path velocity controller is illustrated by simulations. The simulations demonstrate the effect of modeling errors. The nominal velocity profile is computed from minimum time optimization, and model errors are introduced by having an actual robot model in the simulation that is different from the model used in the minimum time optimizaton. It is shown how the nominal minimum time velocity profile results in path deviations, and how the path deviations can be reduced by using path velocity control.

The simulations are described in Examples 4.7–4.13 below. The nominal velocity profiles were presented in Examples 3.1–3.3. The nominal velocity profiles from Examples 3.1 and 3.2 are used in all simulation examples except Example 4.9, where the nominal velocity profile is taken from Example 3.3.

Robot models

The robot model used in Examples 3.1 and 3.2 is the linear model

$$\hat{m}_i \ddot{q}_i = \tau_i, \quad i = 1, 2 \tag{4.47}$$

with the parameters $\hat{m}_1 = \hat{m}_2 = 1$. The actual robot model used in the simulations is given by

$$m_i \ddot{q}_i + d_i \dot{q}_i + d_{f_i} sign(\dot{q}_i) = \tau_i, \quad i = 1, 2 \tag{4.48}$$

The robot model used in Example 3.3 is the nonlinear model (3.34). The actual robot model used in the simulations is obtained by adding viscous friction $d_i \dot{q}_i$, $i = 1, 2$, and nonlinear friction $d_i sign(\dot{q}_i)$, $i = 1, 2$ to the right hand sides in equation (3.34).

Nominal minimum time solution

The nominal velocity profiles from Examples 3.1–3.3 are used in the simulations as nominal velocity profiles for the path velocity controller. We give here also the nominal time functions $s(t)$, $f(s(t))$, and $\tau(t)$.

The velocity profile from Example 3.1, and the corresponding time functions are shown in Figure 4.2. The upper left plot shows the nominal velocity profile. This velocity profile is the same as shown in Figure 3.2. The lower left plot shows the nominal path parameter $s(t)$. The upper right plot shows the nominal trajectories $q_1(t) = f_1(s(t))$ and $q_2(t) = f_2(s(t))$. The lower right plot shows the torques $\tau_1(t)$ and $\tau_2(t)$.

The velocity profiles from Examples 3.2 and 3.3, and the corresponding nominal time functions are shown in Figures 4.3 and 4.4.

Simulations

EXAMPLE 4.7—Friction compensation, elliptical path
This example demonstrates the path velocity controller when there is friction in the actual robot model used in the simulation, but not in the nominal model used in the minimum time optimization. The nominal velocity profile was computed in Example 3.1. The path is thus the elliptical path shown in the upper plot in Figure 3.2. The nominal velocity profile and the nominal time functions are shown in Figure 4.2. The model (4.48) was used in the simulation. Nonlinear friction was added to joint one by choosing the parameters in (4.48) as $m_1 = m_2 = 1$ and $d_1 = d_2 = 0$, $d_{f_1} = 0.1$, and $d_{f_2} = 0$. The robot controller is the linear controller (2.26). The controller parameters K_v and K_p in (2.26) were chosen as diagonal matrices with diagonal elements 18 and 81, respectively.

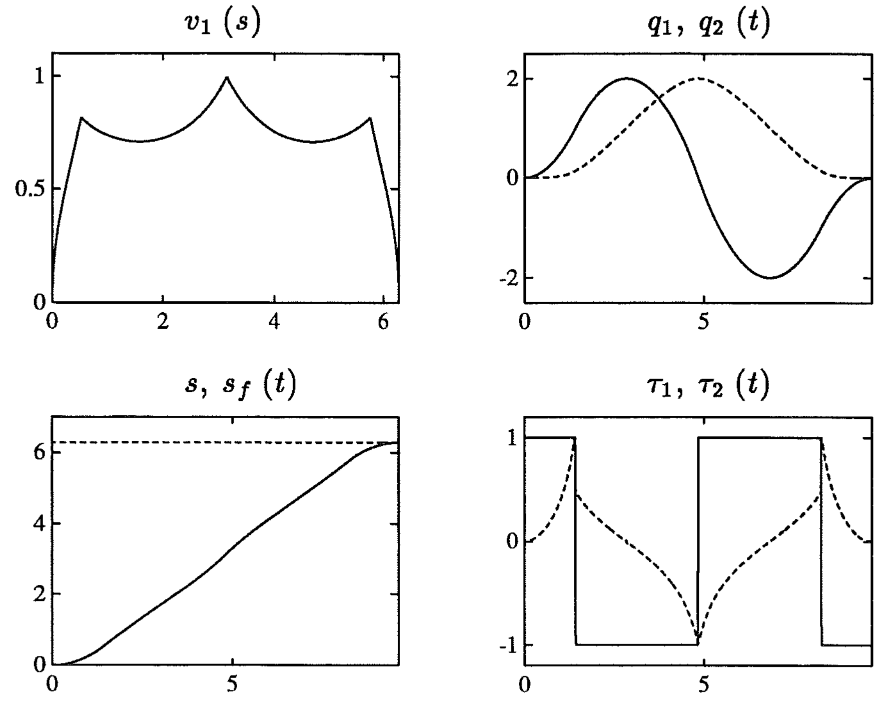


Figure 4.2 The velocity profile from Example 3.1, and the corresponding time functions.

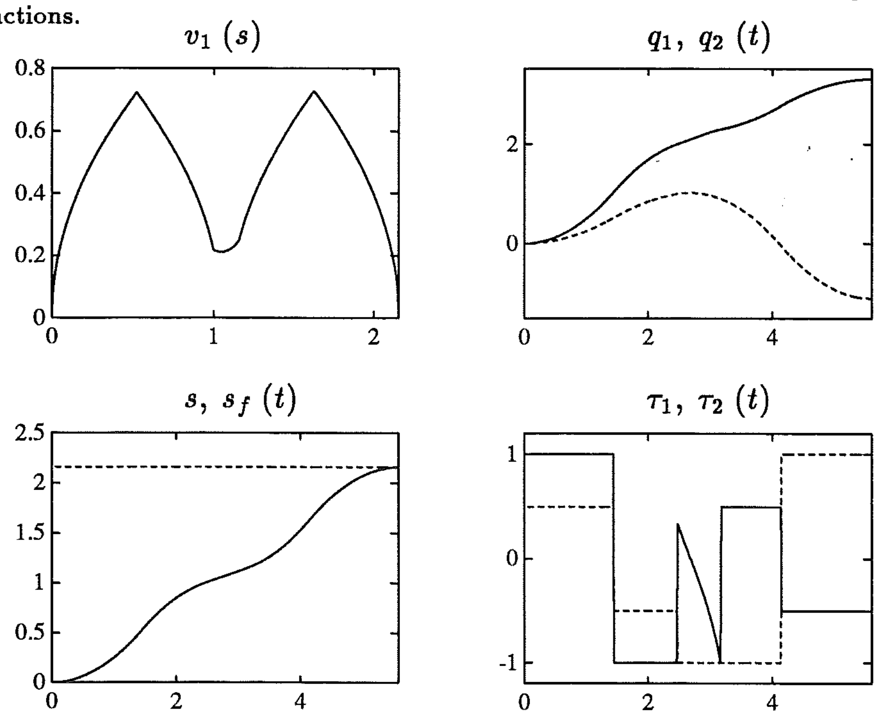


Figure 4.3 The velocity profile from Example 3.2, and the corresponding time functions.

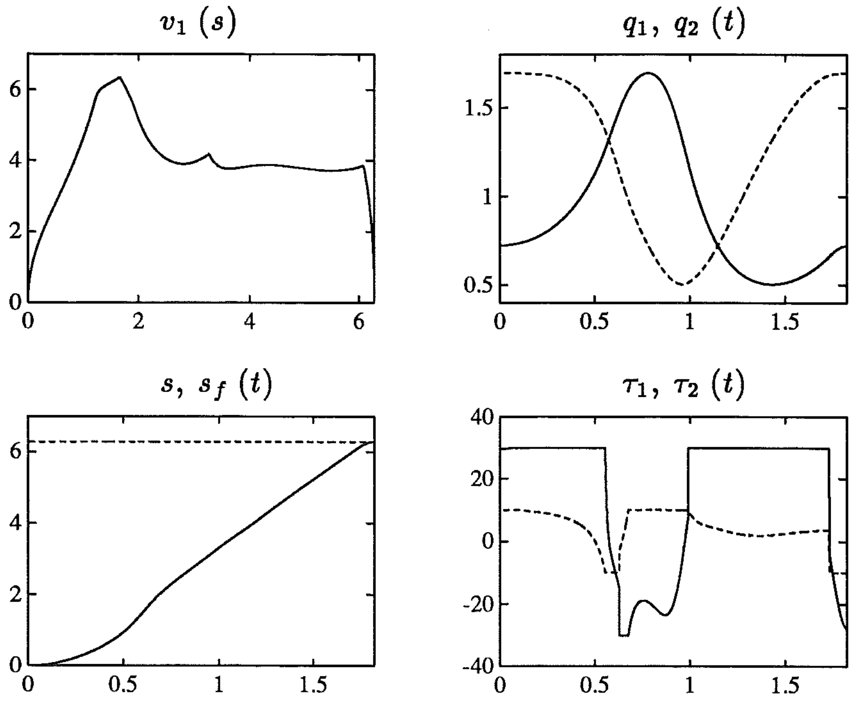


Figure 4.4 The velocity profile from Example 3.3, and the corresponding time functions.

The result of using the nominal path parameter without the path velocity controller is shown in Figure 4.5. The lower left plot shows the nominal path parameter $s(t)$ and s_f, which represents the end of the path. The nominal velocity profile is shown in the mid left plot. The reference trajectories and the actual trajectories are shown in the upper right plot. The dashed line is the reference trajectory $q_1(t) = f_1(s(t))$. This reference trajectory is also shown in the upper right plot in Figure 4.2, solid line. The solid line in the upper right plot in Figure 4.5 is the actual trajectory $q_1(t)$. As can be seen in the plot, there are large deviations from the reference trajectory. These deviations are also seen in the mid right plot, which shows the tracking errors $e_1(t)$, solid line, and $e_2(t)$, dashed line.

The lower right plot in Figure 4.5 shows the torques, $\tau_1(t)$ and $\tau_2(t)$, which are the outputs of the robot controller, in this case the linear controller (2.26). The solid line is τ_1 and the dashed line is τ_2. As can be seen in this plot, the torque τ_2 agrees well with the nominal τ_2, shown in the lower right plot in Figure 4.2. This is expected since there is no model error in joint two. The torque τ_1 is however different from the nominal. Consider the first nominal switch in τ_1. From Figure 4.2, it is seen that the switch occurs at approximately 1.5 seconds, and that the nominal switch is synchronized with τ_2 in the sense that τ_2 is discontinuous at the switching point. In the

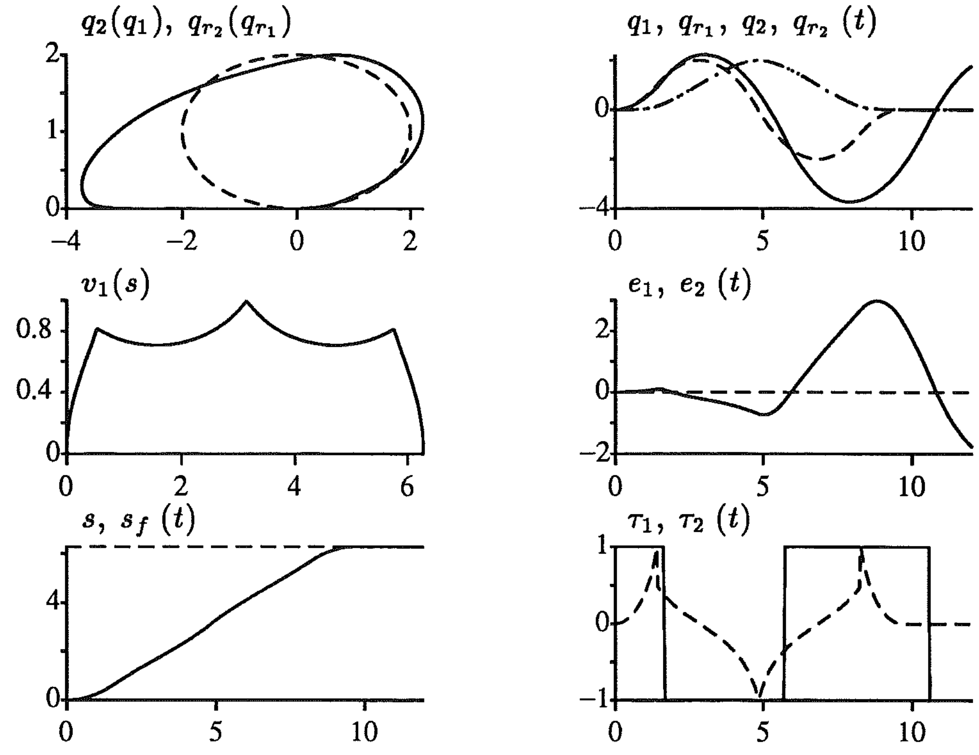


Figure 4.5 Using the nominal velocity profile on a model where friction has been added to joint one, results in path deviation.

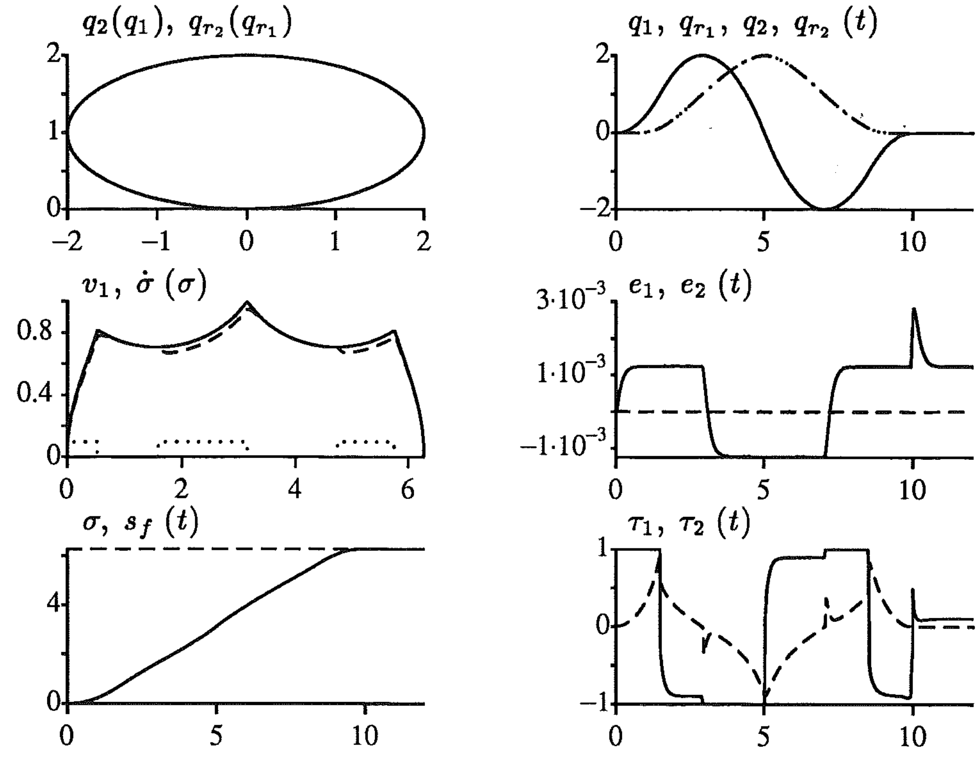


Figure 4.6 The result of using the basic algorithm on a model where friction has been added to joint one.

simulation in Figure 4.5, it is seen that the torques are not synchronized in the same way. Instead, τ_1 remains at the limit after the first nominal switching time. The same phenomenon occurs around 5 seconds, which is the second nominal switch in τ_1, as can be seen from Figure 4.2. The time interval where τ_1 remains at the limit has here increased. The interval where τ_1 remains at the limit is further increased at the final nominal switch in τ_1.

The actual robot model in this simulation has increased friction in joint one, compared to the nominal model. From the lower right plot in Figure 4.2, it is seen that the torque for joint one is nominally at the limit. The result when using the reference trajectory is then that the torque τ_1 saturates, which makes it impossible to track the nominal reference trajectory. This is seen at the first nominal switch in τ_1, where, as can be seen in the mid right plot in Figure 4.5, there is a positive tracking error. The feedback action in the robot controller then gives that τ_1 remains at the limit after the nominal switching time. Further, as can be seen in the mid right plot in Figure 4.5, the tracking error for consecutive switching times is increasing.

The upper left plot in Figure 4.5 shows the desired path, dashed line, and the actual path, solid line. As can be seen in this plot, the path cannot be followed, a result of the large tracking error in joint one.

A simulation where the basic algorithm for path velocity control (4.24) is used is shown in Figure 4.6. The mid left plot shows the nominal velocity profile $v_1(\sigma)$, solid line, and the actual velocity profile, dashed line, obtained by plotting $\dot{\sigma}$ as a function of σ. The dotted line in the mid left plot is a signal that indicates if the limits on $\ddot{\sigma}$ are active, i.e. if $\ddot{\sigma}$ is maximum or minimum. The dotted line is nonzero when the limits are active and zero otherwise. From the mid plot in Figure 3.2, where the dotted line indicates if the path acceleration is nominally maximum or minimum, it is seen that for the simulation in Figure 4.6, the limits on $\ddot{\sigma}$ are activated during acceleration. This means that the nominal acceleration is too high, and therefore limited by the path velocity controller. The result is that the path velocity is reduced, resulting in an actual velocity profile below the nominal velocity profile. During deceleration, the limits on $\ddot{\sigma}$ are not active. The actual velocity profile then converges to the nominal velocity profile with a σ-time constant $\frac{1}{\alpha}$. The parameter α was chosen as $\alpha = 10$.

The result of the velocity reduction is that the reference trajectory can now be followed. This is shown in the upper right plot, where it is seen that the tracking is good. The tracking errors are also small, as can be seen from the mid right plot.

The torques are shown in the lower right plot. As can be seen in this plot, the first switch in τ_1 is now synchronized with τ_2. Compare with the lower right plot in Figure 4.5. The second and third switches in τ_1 are also synchronized with τ_2. The second switch occurs in this simulation at 5 seconds. A comparison with the lower right plot in Figure 4.2 shows that the switch is delayed. This agrees well with the discussion following (4.18), where it was shown that the path velocity controller maintains the synchronization, but delays the switching times.

As can be seen in the lower right plot in Figure 4.6, the torque τ_1 is for some intervals not at the limit. A comparison with Figure 3.2 shows that the path acceleration is nominally minimum in these intervals. When the path acceleration is nominally maximum, the torque τ_1 is at the limit. The behavior thus agrees with what could be predicted from the approximate analysis in Example 4.5, where it was indicated that when friction is underestimated, which is the case here, the basic algorithm would suffice, and also that the motion during deceleration whould actually be conservative, i.e. none of the torques are at the limit.

The lower left plot shows the path parameter $\sigma(t)$ and s_f. The traversal time was computed as $\sigma(t_f) = s_f$, resulting in $t_f = 9.99$ seconds. The nominal traversal time, given in Example 3.1, is $t_{f_{nom}} = 9.66$ seconds.

The upper left plot shows that the path can now be followed, a result of the reference trajectory modification done by the path velocity controller.

EXAMPLE 4.8—Friction compensation, corner path

In this example, the same model as in the previous example is used in the simulation. The path and the nominal velocity profile are now from Example 3.2. The path is thus the corner path, shown in the upper plot in Figure 3.3. The nominal velocity profile and the nominal time functions are shown in Figure 4.3.

A simulation where the nominal path parameter is used is shown in Figure 4.7. The lower left plot shows the nominal path parameter. The nominal path parameter is also shown in the lower left plot in Figure 4.3. The mid left plot in Figure 4.7 shows the nominal velocity profile, also shown in the upper left plot in Figure 4.3. The upper right plot shows the reference trajectories and the actual trajectories. As can be seen, the trajectory $q_1(t)$, solid line, deviates from the reference trajectory $q_{r_1}(t)$, dashed line. This deviation is also seen as a tracking error $e_1(t)$, shown in the mid right plot. A comparison with the previous example, shown in Figure 4.5, shows that the tracking error in this simulation is significantly smaller.

The lower right plot shows the torques. As can be seen in this plot, the torque τ_2 agrees well with the nominal τ_2, shown in the lower right plot in Figure 4.3. This is expected, since, as in the previous example, the model error is only in joint one, where the friction is underestimated. As was the case in the previous example, the torque τ_1 remains at the limit after the first nominal switch. It can also be seen that τ_1 remains at the lower limit after the second discontinuity in τ_2. This is not the case for the nominal τ_1, as can be seen in Figure 4.3.

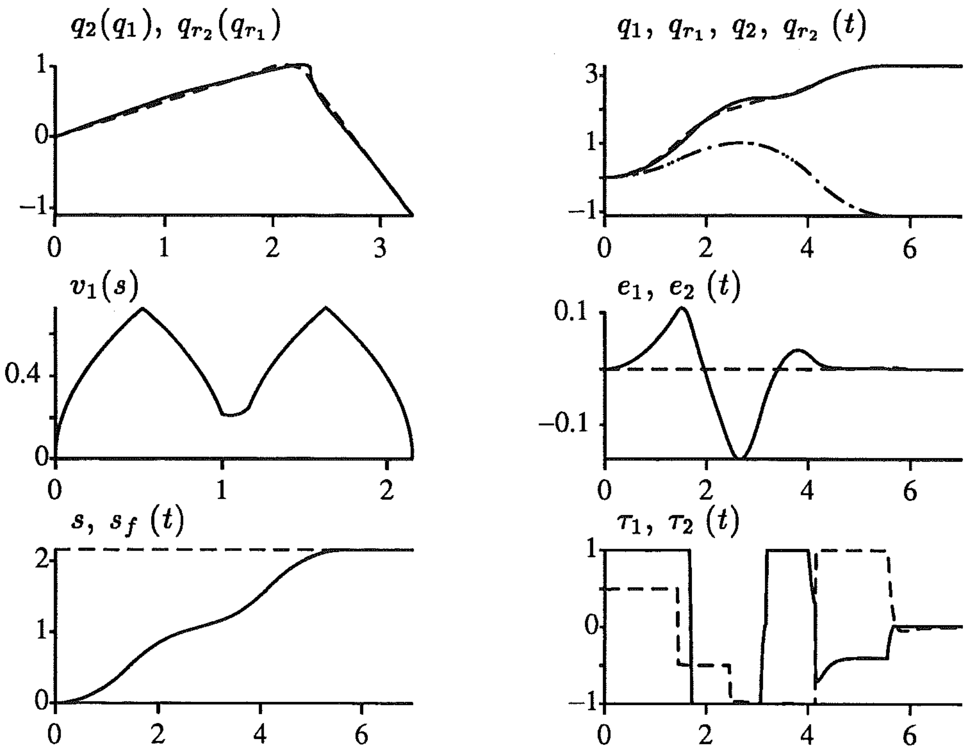


Figure 4.7 The result of using the nominal velocity profile on a model where friction has been added to joint one. The result is path deviation, however not as large as for the elliptical path, shown in Figure 4.5.

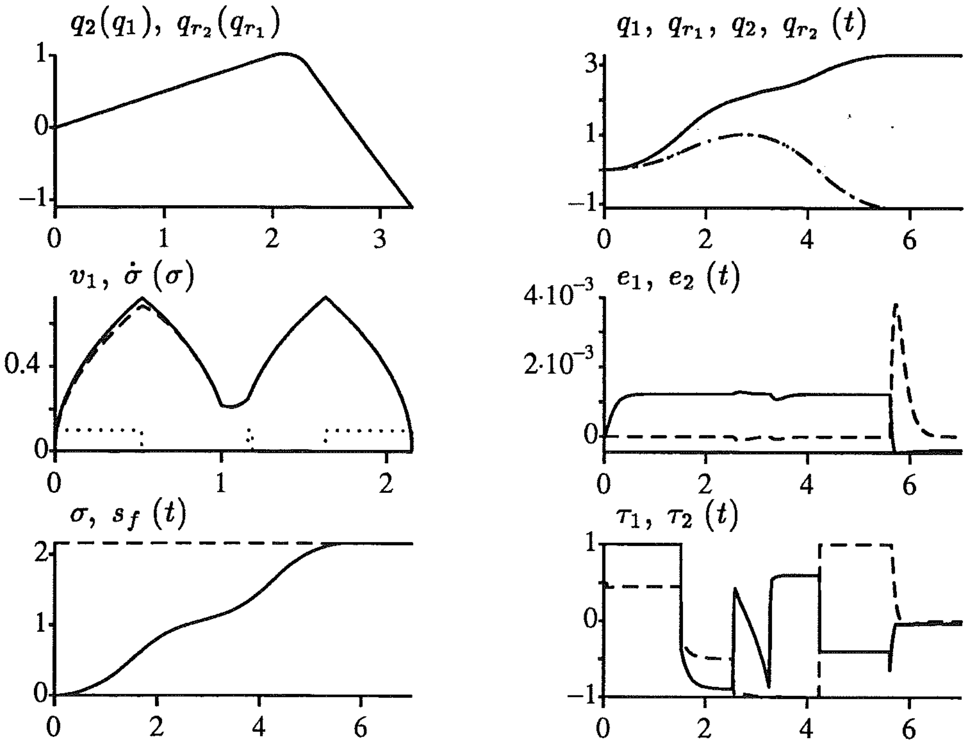


Figure 4.8 The result of using the basic algorithm on a model where friction has been added to joint one.

The upper left plot shows the desired and the actual path. As can be seen, the path deviation is not as large as for the elliptical path, shown in the upper left plot in Figure 4.5.

Using the basic algorithm for path velocity control (4.24) gives the result shown in Figure 4.8. The mid left plot shows the nominal velocity profile, solid line, and the actual velocity profile, dashed line. The acceleration and deceleration intervals are shown in Figure 3.3, where it is seen that joint one is limiting during the first acceleration interval. Since the model used in the simulation was obtained from the nominal model by adding friction to joint one, it is expected that the nominal acceleration is too high in this interval, and hence that the limits on $\ddot{\sigma}$ should be activated. The dotted line in the mid left plot in Figure 4.8 shows that this is indeed the case. The limits on $\ddot{\sigma}$ are activated during the first acceleration interval, and the velocity is reduced. For this simulation, the limits are also used for a short interval around $\sigma = 1.2$. From Figure 3.3, it is seen that the nominal τ_1 reaches the limit at this point. As can be seen from the dotted line in the mid left plot in Figure 4.8, the limits are also used during the final deceleration. From Figure 3.3, it is seen that joint two is limiting during this interval. Since the model for joint two is perfect, it is not expected that the limits on $\ddot{\sigma}$ are active during the final deceleration. There is however no visible difference in the velocity profiles during this interval, i.e. the acceleration modification done by the path velocity controller is small, indicating that the nominal acceleration, due to numerical inaccuracy, is slightly lower than the minimum possible.

The torques are shown in the lower right plot. As can be seen, the torques are now synchronized. Further, as was the case in the previous example, the motion is conservative during the first deceleration phase, as can be seen from τ_1 not being at the limit in this interval.

The upper right plot shows that the tracking is good. As a result of this, the path can be followed. This is seen in the upper left plot. The traversal time for this simulation was $t_f = 5.62$ seconds. The nominal traversal time, given in Example 3.2, was $t_{f_{nom}} = 5.60$ seconds.

A comparison with the previous example shows that the percentual increase in traversal time is smaller, and also that the path deviation when the nominal path parameter is used is smaller. Except that the shape of the path is different, in this example, joint one, i.e. the joint with increased friction, is limiting only during the first line segment. In the previous example, joint one was limiting for the entire path.

EXAMPLE 4.9—Friction compensation, nonlinear robot model

In this example, the nonlinear model (3.34) is used. The nominal velocity profile was computed in Example 3.3. The path is the cartesian circle (3.35). The path in joint space is shown in the upper plot in Figure 3.4. The nominal velocity profile and the nominal time functions are shown in Figure 4.4. The

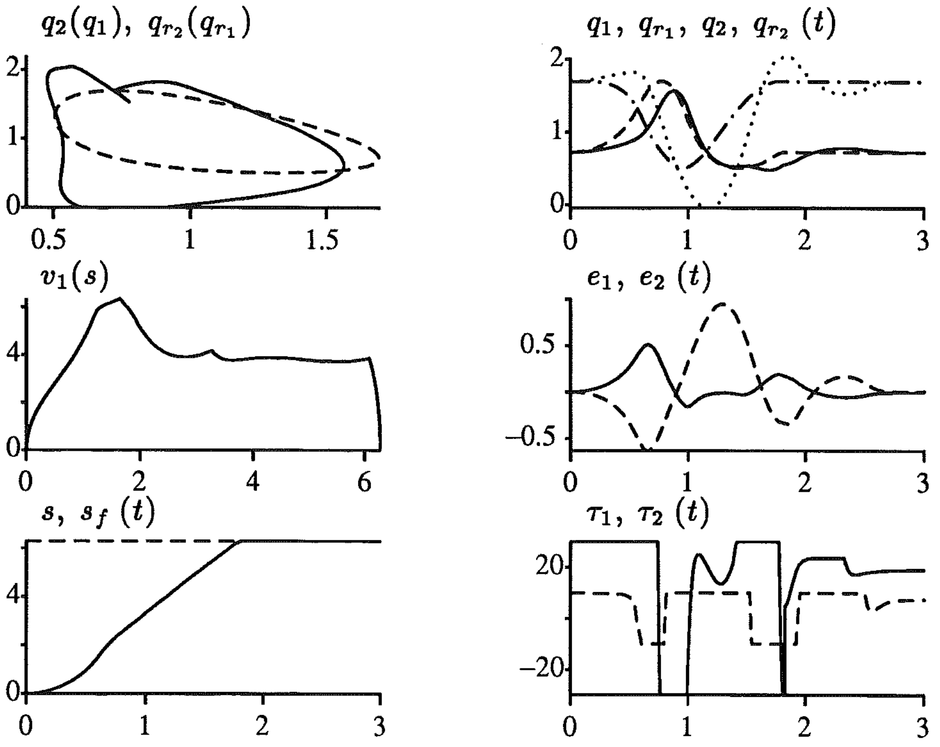


Figure 4.9 The result of using the nominal velocity profile on a nonlinear robot model where friction has been added to joint one. Due to the coupled dynamics, there are tracking errors in both joints.

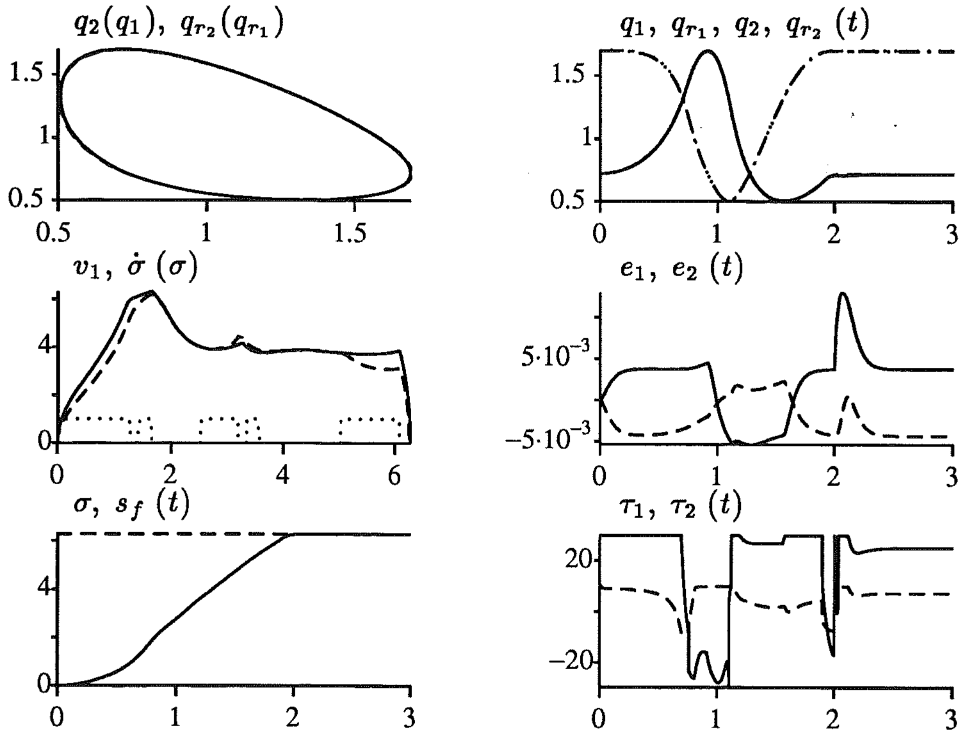


Figure 4.10 The result of using the basic algorithm on a nonlinear robot model where friction has been added to joint one.

robot model used in the simulation is obtained from (3.34) by adding nonlinear friction to joint one. This means that the term $d_{f_1} sign(\dot{q}_1)$ is added to the right side in the equation for τ_1 in (3.34). The parameter choice was $d_{f_1} = 3$. Compared with the previous examples, the friction level is also here 10 % of the maximum torque. The robot controller is the computed torque controller (4.5). The controller parameters K_v and K_p in (4.5) were chosen as diagonal matrices with diagonal elements 40 and 400, respectively.

The result of using the nominal path parameter is shown in Figure 4.9. The nominal velocity profile is shown in the mid left plot. The nominal path parameter is shown in the lower left plot. The nominal velocity profile and the nominal path parameter are also shown in Figure 4.4.

The reference trajectories and the actual trajectories are shown in the upper right plot. The reference trajectories $q_{r_1}(t)$, dashed line, and $q_{r_2}(t)$, dashed-dotted line, are also shown in the upper right plot in Figure 4.4. As can be seen in the upper right plot in Figure 4.9, the actual trajectories $q_1(t)$, solid line, and $q_2(t)$, dotted line, deviate from the reference trajectories. Note that there is only model error in joint one. Due to the coupled nonlinear dynamics (3.34), this results in tracking errors in both joints. This was not the case in the previous examples, as can be seen in Figures 4.5 and 4.7.

The lower right plot shows the torques. A comparison with the lower right plot in Figure 4.4 shows that both torques are now different from the nominal.

The upper left plot shows the desired and the actual path in joint space. As can be seen in this plot, the path cannot be followed.

The result of using the basic algorithm for path velocity control (4.24) is shown in Figure 4.10. The nominal and the actual velocity profiles are shown in the mid left plot. The acceleration and deceleration intervals are shown by the dotted line in the mid plot in Figure 3.4. From the dotted line in the mid left plot in Figure 4.10, it is seen that the limits on $\ddot{\sigma}$ are used during the first acceleration interval. The limits are also used during the final acceleration interval. This the same behavior as for the previous examples, where the acceleration phase was not feasible due to underestimation of friction. In this example, the limits are used also during the first deceleration phase, resulting in an actual velocity which is above the nominal velocity profile. This occurs around $\sigma = 3$, as can be seen in the mid left plot. It is however difficult to judge if the activation of the limits is a result of numerical inaccuracy, or if the deceleration is really unfeasible. The limits were also used during deceleration in Figure 4.8, even though the nominal minimum acceleration was feasible.

The reference trajectories and the actual trajectories are shown in the upper right plot. As can be seen in this plot, the result of using the path velocity controller is also in this example that the reference trajectory is modified so that the tracking is good. As a result of this, the path can be followed, as can be seen in the upper left plot. The traversal time for this simulation was $t_f = 2.04$ seconds. The nominal traversal time, given in Example 3.3, was $t_{f_{nom}} = 1.82$ seconds.

EXAMPLE 4.10—Errors in mass parameters, corner path
This simulation demonstrates how the path velocity controller can compensate for mass errors. The algorithm (4.29) is used. The path and the nominal velocity profile are the same as in Example 4.8. The model (4.48) is used in the simulation. The parameters were chosen as $m_1 = 1.1$, $m_2 = 1$, $d_1 = d_2 = 0$, $d_{f_1} = 0$, and $d_{f_2} = 0$, i.e. compared to the nominal model, the mass of joint one is increased.

A simulation where the nominal path parameter is used is shown in Figure 4.11. The mid left plot shows the nominal velocity profile, and the lower left plot shows the nominal path parameter. These plots are the same as the corresponding plots in Figure 4.7.

The upper right plot in Figure 4.11 shows the reference trajectories and the actual trajectories. As can be seen in this plot, the actual trajectory $q_1(t)$, solid line, deviates from the reference trajectory $q_{r_1}(t)$, dashed line. The deviation is also seen in the mid right plot, which shows the tracking errors.

The lower right plot shows the torques. The nominal torques are shown in the lower right plot in Figure 4.3. As can be seen in the lower right plot in Figure 4.11, the torque τ_1 remains at the limit after the first nominal switch. The same behavior is seen in Figure 4.7. The behavior for $t > 2$ seconds is however different. In Figure 4.11, τ_1 is saturated for the rest of the motion. This is not the case in Figure 4.7, where for $t > 4$ seconds, the torque τ_1 is not at the limit. This has the consequence that it is possible to achieve zero tracking error before the end of the path in Figure 4.7. This is not the case in Figure 4.11. By comparing the upper left plot in Figures 4.7 and 4.11, it is seen that the path deviation is also different for the two cases. The path deviation in Figure 4.11 is larger, i.e. for this specific path, the sensitivity to increasing the mass of joint one by 10% is larger than adding friction with a level of 10% of the maximum torque.

A simulation where the algorithm (4.29) is used is shown in Figure 4.12. The mid left plot shows the nominal velocity profile and the actual velocity profile. The dotted line in this plot shows that the limits are used during the first acceleration interval. This results in a velocity profile which is lower than the nominal. However, since the algorithm (4.29) is used, the scaling factor γ is simultaneously reduced. This is seen in the lower right plot, which shows γ as a function of σ. The parameter k in (4.29) was chosen as $k = 10$. This corresponds to a σ-time constant $\frac{1}{k} = 0.1$. As can be seen in the lower right plot, this seems to be the case for the actual γ obtained in the simulation.

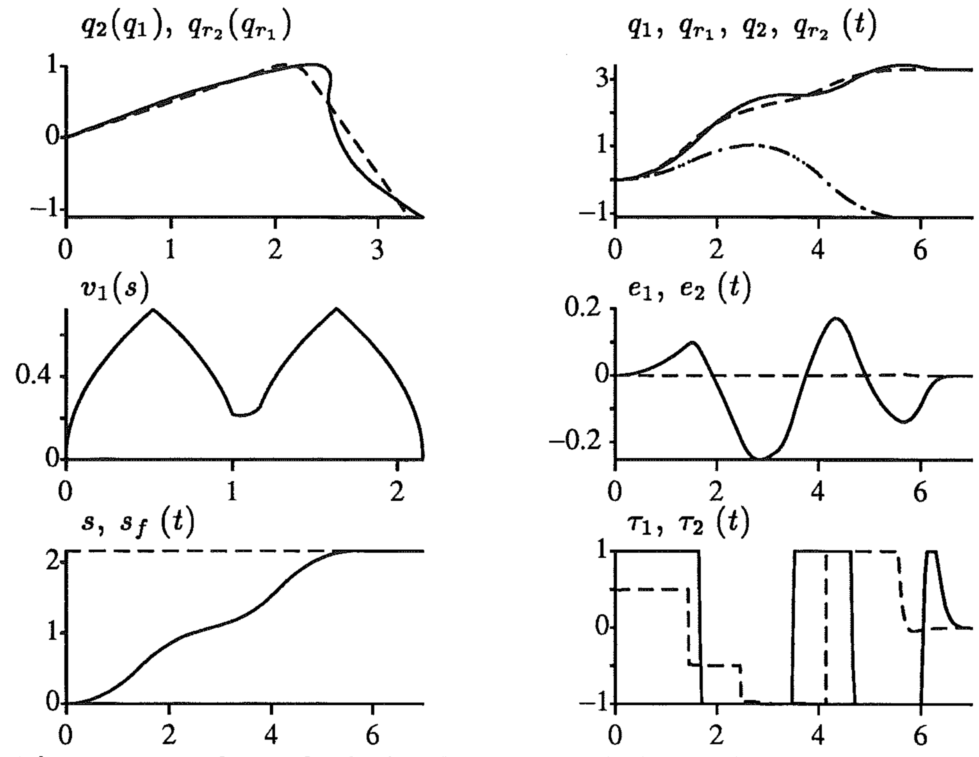


Figure 4.11 The result of using the nominal velocity profile on a model where the mass in joint one is increased. The result is path deviation, which is larger than for the case of increased friction, shown in Figure 4.7.

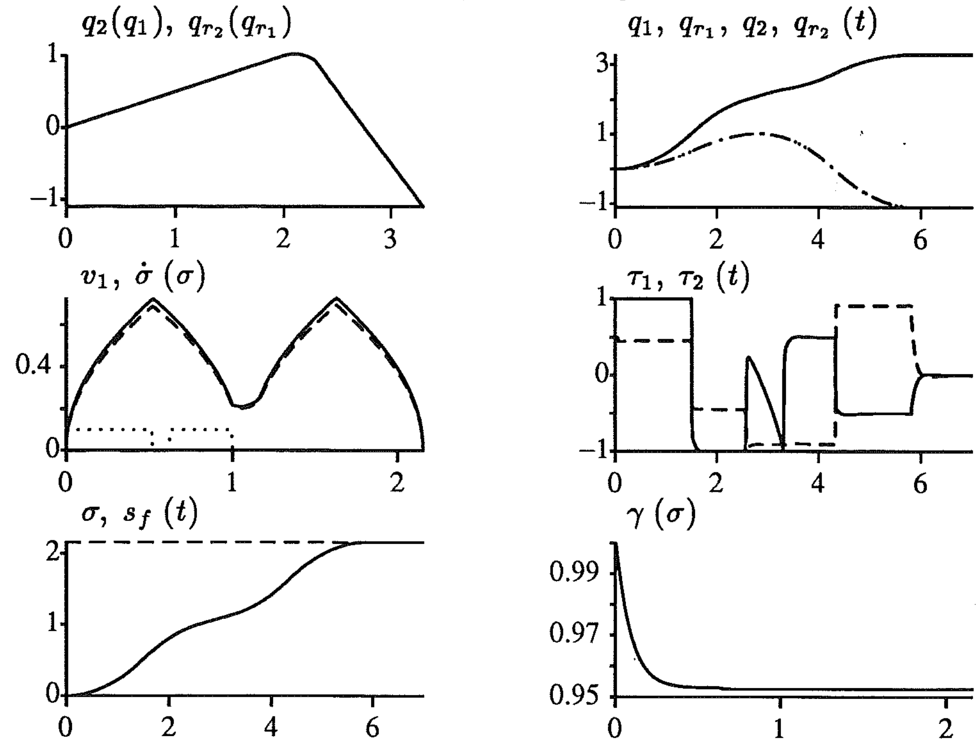


Figure 4.12 The result of using the algorithm with velocity profile scaling on a model where the mass in joint one is increased. The lower right plot shows the scaling factor γ, which is decreased during the first acceleration interval.

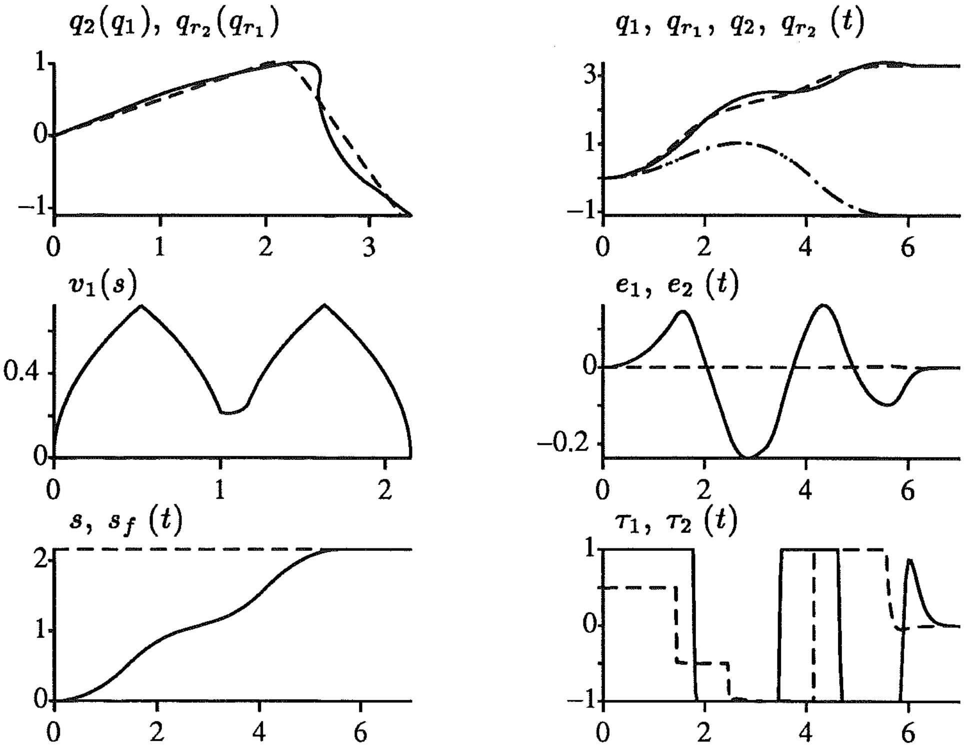


Figure 4.13 The result of using the nominal velocity profile on a model with increased mass and viscous friction. The path deviation is comparable to the case shown in Figure 4.11.

The final value of γ obtained from the simulation was 0.953. The theoretical γ, computed from (4.43), is $\gamma_a = 0.954$. Thus, the simulation agrees with the approximate analysis resulting in (4.27) and (4.43).

The algorithm (4.29) tries to estimate a suitable γ during acceleration such that the following deceleration is feasible. This was done succesfully in this simulation. However, the algorithm keeps the so obtained γ also during the rest of the motion. For this simulation, this results in a conservative motion during the second line segment of the path. This can be seen in the mid right plot in Figure 4.12, where during the second half of the motion, none of the torques are at the limit.

The conservative motion is here expected. The algorithm adjusts γ during the first line segment, and hence compensates for the model error in joint one. For the second line segment, as can be seen in Figure 3.3, joint two is limiting. Since there are no model errors in this joint it would have been possible to increase γ during the second line segment. An improved algorithm (4.29) would e.g. be to reset γ to one at the start of each acceleration phase.

EXAMPLE 4.11—Mass errors and friction compensation, corner path

This example demonstrates how the basic algorithm (4.24) can be used when the model used in the simulation has increased mass and viscous friction

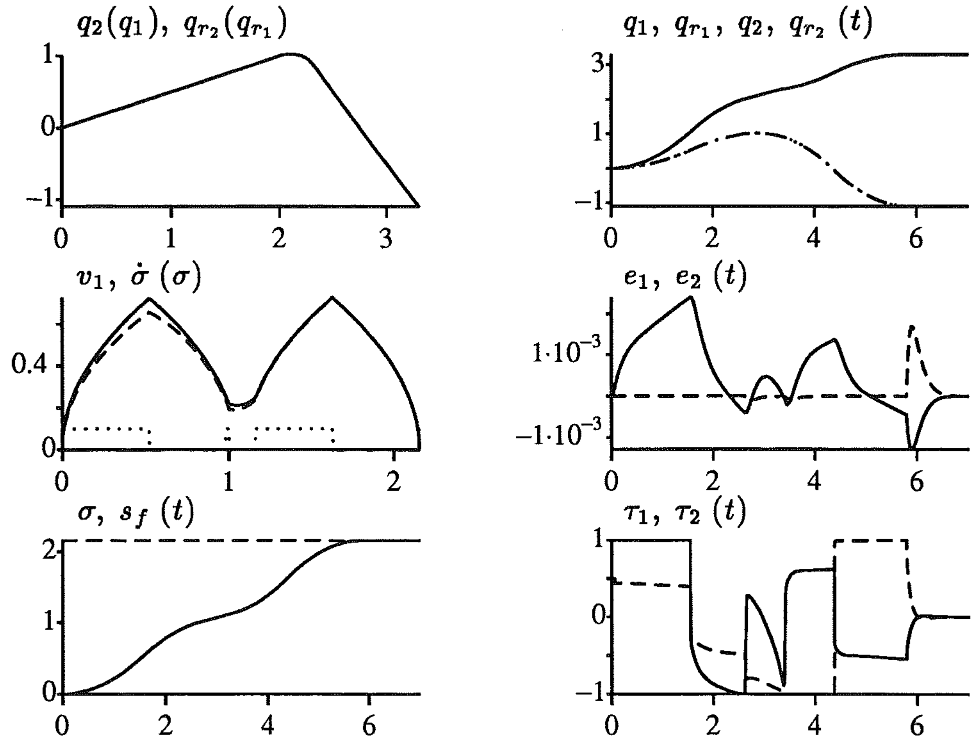


Figure 4.14 The result of using the basic algorithm with $\alpha = 5$ on a model with increased mass and viscous friction.

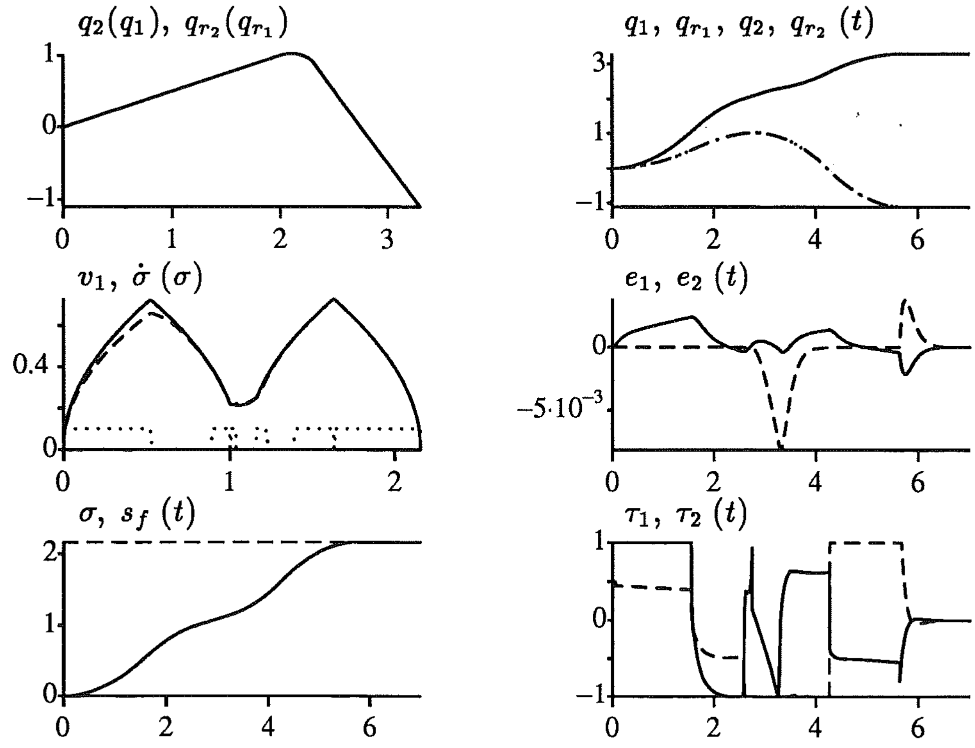


Figure 4.15 The result of using the basic algorithm with $\alpha = 10$ on a model with increased mass and viscous friction.

compared to the nominal model. The model (4.48) is used in the simulation. The parameters in (4.48) are chosen as $m_1 = 1.1$, $m_2 = 1$ and $d_1 = 0.1$, $d_2 = 0$, $d_{f_1} = 0$, and $d_{f_2} = 0$, i.e. compared to the nominal model, the mass of joint one is increased, and viscous friction is added. The path and the nominal velocity profile are the same as in Examples 4.8 and 4.10.

The result of using the nominal path parameter is shown in Figure 4.13. The nominal velocity profile is shown in the mid left plot, and the nominal path parameter is shown in the lower left plot. These plots are the same as the corresponding plots in Figures 4.7 and 4.11.

The reference trajetories and the actual trajectories are shown in the upper right plot. A comparison with the corresponding plot in Figure 4.11 shows that the behavior is similar. This is true also for the torques, shown in the lower right plot, and the path deviation, shown in the upper right plot, i.e. the added viscous friction does not change the qualitative behavior of the system. A difference in the torques can however be seen around the first nominal switch in τ_1. In Figure 4.13, the torque τ_1 remains at the limit for a longer time interval than in Figure 4.11, an effect probably caused by the added viscous friction.

The model used in this simulation is the same as was used in the simulation example in Chapter 2. Further, the first line segment of the path coincides with the path used in Chapter 2. A comparison with Figure 2.6 shows that the path deviation during the first line segment, shown in the upper left plot in Figure 4.13 is similar to the path deviation in Figure 2.6. The path deviation during the last line segment is however larger. The comparison shows that when the path is continued, the given model error gives larger path deviation than for the simple straight line path.

The result of using the basic algorithm for path velocity control (4.24) is shown in Figure 4.14. The mid left plot shows the nominal and the actual velocity profiles. As can be seen from the dotted line in this plot, the limits on $\ddot{\sigma}$ are used during the first acceleration interval.

The parameter α in the path velocity controller (4.24) was chosen as $\alpha = 5$. This means that, when the limits on $\ddot{\sigma}$ are not active, $\dot{\sigma}^2 \rightarrow v_1(\sigma)^2$ with a σ-time constant $\frac{1}{\alpha} = 0.2$. This was the parameter choice also in the simulation example in Chapter 2. As can be seen in the mid left plot in Figure 4.14, the limits are not used during the first deceleration phase. A comparison with Figure 2.9 shows that the actual velocity profiles are similar in this interval.

A simulation where α is increased to $\alpha = 10$ is shown in Figure 4.15. The mid left plot shows the nominal and the actual velocity profiles. As can be seen, the actual velocity profile now approaches the nominal velocity profile faster than in Figure 4.14. It is also seen that the limits are used more often than in Figure 4.14. Another difference is seen in the torques, shown in

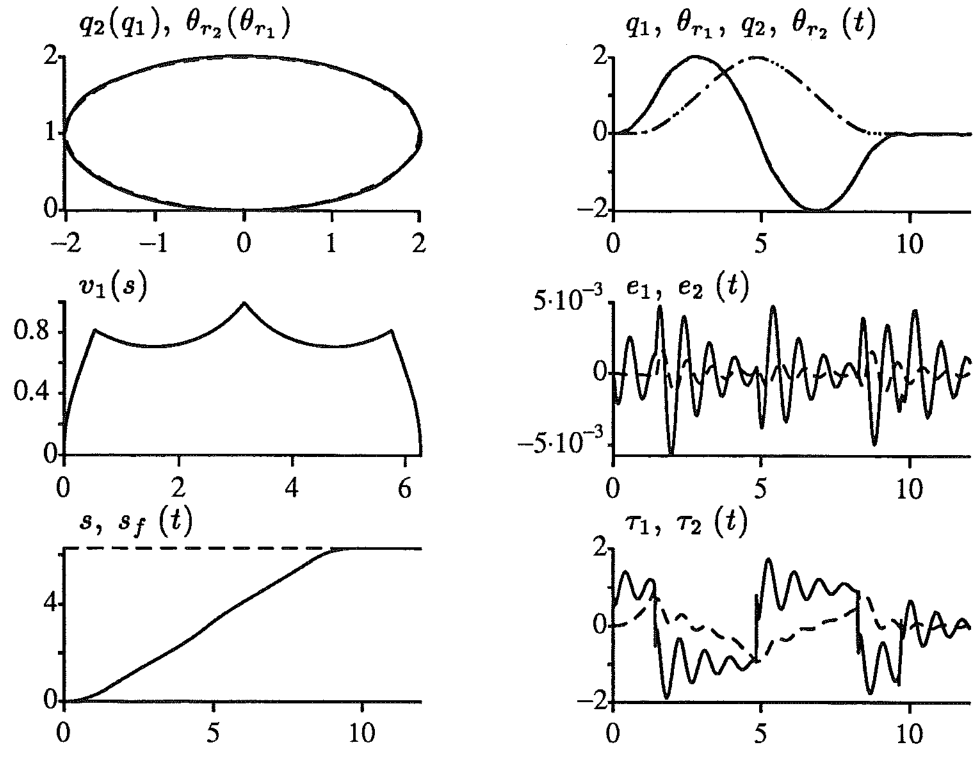


Figure 4.16 The result of using the nominal velocity profile on a model with joint flexibility. The torque limits have been removed.

the lower right plot. The time interval between the first and second switch in τ_1 is now smaller. This interval is the first deceleration phase, which thus is executed faster than in the previous simulation. The result of increasing α for this simulation is a decreased traversal time. The traversal time for the simulation in Figure 4.14 was 5.80 seconds. The traversal time for the simulation in Figure 4.15, where α was increased, was 5.64 seconds.

EXAMPLE 4.12—Joint flexibility
This example demonstrates how the path velocity controller can be used when the minimum time optimization is based on a rigid model, and the robot model used in the simulation has joint flexibility. The flexible joint model is chosen as the linear model (3.64) and (3.65). The parameters in (3.65) are chosen as

$$m_1 = m_2 = 0.5, \quad J_1 = J_2 = 0.5, \quad k_1 = k_2 = 30 \tag{4.49}$$

The controller is a linear second order controller. The controller measures the actuator angles θ in (3.64). The reference trajectory is now denoted θ_r and is given by $\theta_r(t) = f(\sigma(t))$. The controller is the same as used in previous simulations, i.e. (2.26). The controller is here written as

$$\tau = \hat{M}(\ddot{\theta}_r + K_v(\dot{\theta}_r - \dot{\theta}) + K_p(\theta_r - \theta)) \tag{4.50}$$

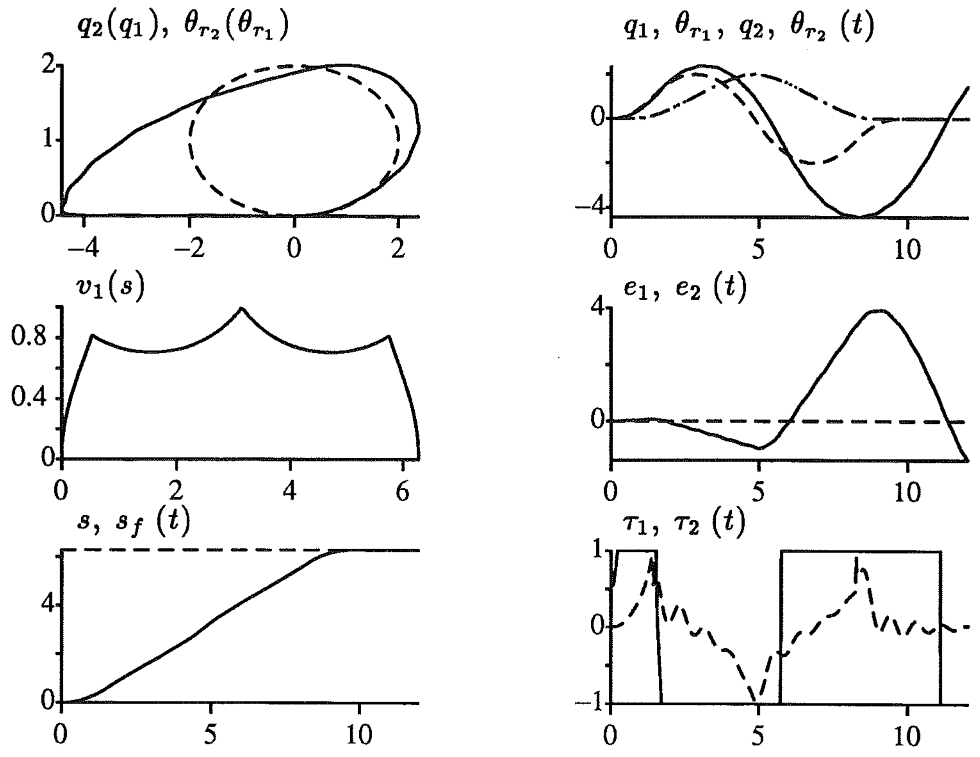


Figure 4.17 The result of using the nominal velocity profile on a model with joint flexibility when the torques are limited.

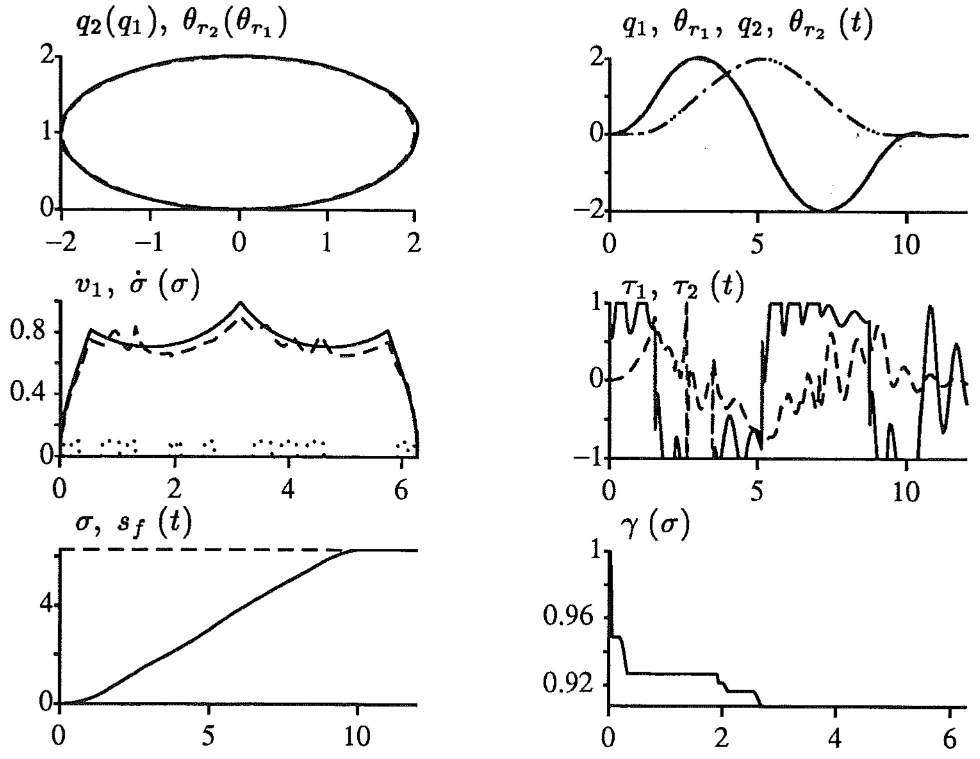


Figure 4.18 The result of using the algorithm with velocity profile scaling on a model with joint flexibility.

This example uses the nominal velocity profile from Example 3.1. A simulation where the nominal path parameter is used, and where the torque limits have been removed is shown in Figure 4.16. The nominal velocity profile and the nominal path parameter are shown in the mid left and in the lower left plots. These plots are the same as the corresponding plots in Figure 4.5. The reference trajectories $\theta_{r_i}(t)$, $i = 1, 2$, and the actual trajectories $q_i(t)$, $i = 1, 2$ are shown in the upper right plot. As can be seen in this plot, the tracking is good. This shows that the robot controller, which is designed for the rigid model (4.47), is robust with respect to the chosen flexible joint model. There are however small, oscillative tracking erorrs, as can be seen in the mid right plot. These errors are the actual tracking errors used by the robot controller (4.50), i.e. $e_i(t) = \theta_{r_i}(t) - \theta_i(t)$, $i = 1, 2$.

The lower right plot shows the torques. As can be seen in this plot, the torque τ_1 is outside the limits ± 1.

A simulation where the torques are limited is shown in Figure 4.17. The upper right plot shows that the reference trajectory of joint one cannot be followed. This is also seen as a large tracking error in the mid right plot. The lower right plot shows the torques. As can be seen, τ_1 is saturated. The result is that the path cannot be followed, as is seen in the upper left plot.

A simulation where the path velocity controller (4.29) is used is shown in Figure 4.18. The mid left plot shows the nominal and the actual velocity profiles. The acceleration and deceleration intervals are shown in Figure 3.2. The dotted line in the mid left plot in Figure 4.18 shows that the limits are used during the first acceleration interval. This has the effect that the scaling factor γ is reduced. This is shown in the lower right plot. The limits are also used during the first deceleration interval. In this interval γ is not modified, as can be seen in the lower right plot. The reason for this is the conditions on $\dot{\gamma}$ in (4.29), where γ is only modified when the actual velocity profile is below the scaled nominal velocity profile. The result of using the limits during the first deceleration interval is that the actual path velocity is for some intervals above the nominal velocity profile.

The upper right plot shows the reference trajectories $\theta_{r_i}(t)$, $i = 1, 2$, and the actual trajectories $q_i(t)$, $i = 1, 2$. As can be seen in this plot, the tracking is good.

The torques are shown in the mid right plot. As can be seen in this plot, the torque τ_1 occasionally reaches the limit. A comparison with the corresponding plot in Figure 4.16 shows that the torque obtained when the path velocity controller is used has the same oscillative behavior, but the oscillations that are outside the limits are cut off. Note that this effect is not seen when using the torque limits only, as in Figure 4.17. Instead, when the path velocity controller is used, the path acceleration is adjusted so that the oscillations that are outside the limits instead become variations in the

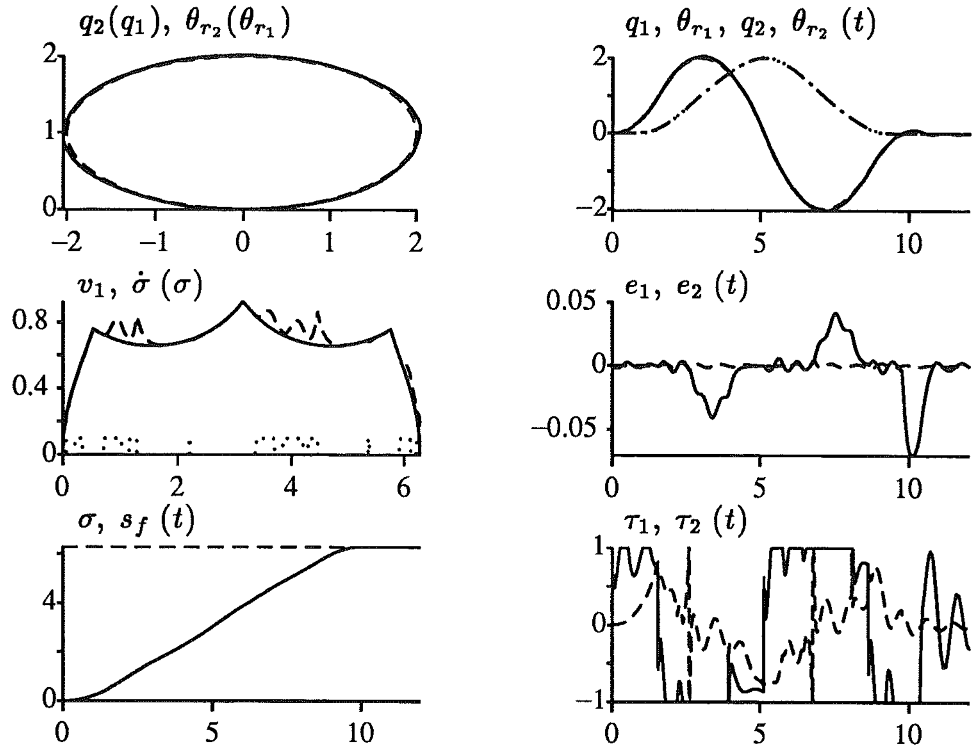


Figure 4.19 The result of using the basic algorithm and a constant scaling factor on a model with joint flexibility.

path acceleration. Thus, the path velocity controller "leaves room" for the variations in τ_1, by adjusting the path acceleration. Since the torque is oscillatory, the limits on acceleration are used only when the oscillation is outside the limits. In Figure 4.18, this is seen in the mid left plot, where the dotted line is more irregular than in previous simulations. This also has the effect that the adjustment of γ is more irregular, as can be seen in the lower right plot. This is because γ is only adjusted when the limits are used. Compare the lower right plots in Figures 4.18 and 4.12. The parameter choice $k = 10$ is used in both simulations, but the behavior of γ is different.

The result of using the path velocity controller in this simulation is however good tracking. This is seen in the upper right plot, and in the upper left plot, which shows that the desired path can now be followed. The traversal time when the path velocity controller is used was $t_f = 10.04$ seconds.

EXAMPLE 4.13—Constant velocity profile scaling

The algorithm (4.29) tries to estimate a scaling factor during acceleration, which also can be used during the following deceleration. Another application of the algorithm to use the resulting γ as a basis for choosing a constant scaling factor. As was seen in the previous simulation, γ was decreased in two steps. Picking the final value of γ is of course one solution, but since traversal time is to be minimized, one could try as large scaling as possible, while keeping the basic algorithm for path velocity control. Choosing $\gamma = 0.93$ by inspection in the lower right plot in Figure 4.18, and then using this γ as a constant scaling factor in the basic algorithm as in (4.26), gives the result shown in Figure 4.19. A comparison with the simulation in Figure 4.18, where the algorithm (4.29) was used, shows that the behavior is qualitatively the same. The traversal time has however decreased. The traversal time is now $t_f = 9.84$. The traversal time for the simulation in Figure 4.18, where the algorithm (4.29) was used, was $t_f = 10.04$ seconds.

4.6 Extension to Higher Order Systems

This section demonstrates how higher order path velocity controllers can be obtained. The controller (4.24) is generalized. The result is a controller of order p. The states in the controller are the path parameter σ and its time derivatives up to order $p-1$, i.e. $\sigma, \dot{\sigma}, \ldots, \sigma^{(p-1)}$. The feedback ψ in (4.16) is, as before, a nonlinear state feedback, now written as

$$\psi(\sigma, \dot{\sigma}, \ldots, \sigma^{(p-1)}) \tag{4.51}$$

The controller parametrization (4.1) is generalized to

$$\tau = \beta_1 \sigma^{(p)} + \beta_2 \tag{4.52}$$

An example of the controller parametrization (4.52) for the case $p = 4$ was given in Example 4.4.

The controller parametrization (4.52) can be used to compute limits on $\sigma^{(p)}$ in the same way as the limits on $\ddot{\sigma}$ (4.14) are computed by the computational procedure (4.11)–(4.14). The resulting limits are denoted $\sigma^{(p)}_{min}(\beta_1, \beta_2)$ and $\sigma^{(p)}_{max}(\beta_1, \beta_2)$. The basic algorithm (4.16) is then generalized as

$$\begin{aligned}
\frac{d\sigma}{dt} &= \dot{\sigma} \\
\frac{d\dot{\sigma}}{dt} &= \ddot{\sigma} \\
&\vdots \\
\frac{d\sigma^{(p-1)}}{dt} &= \sigma^{(p)} \\
u_r &= \psi(\sigma, \ldots, \sigma^{(p-1)}) \\
\sigma^{(p)} &= sat(u_r, \sigma^{(p)}_{min}(\beta_1, \beta_2), \sigma^{(p)}_{max}(\beta_1, \beta_2))
\end{aligned} \tag{4.53}$$

The purpose of the feedback ψ in (4.53) is to make $\dot{\sigma} \to v_1(\sigma)$ when the limits on $\sigma^{(p)}$ are not active. The feedback includes, as before, the nominal velocity profile $v_1(\sigma)$ and the nominal acceleration profile $v_2(\sigma)$, but now also higher order derivatives are used. These are represented by the functions $v_3, \ldots, v_p$, defined together with v_1 and v_2 as

$$\begin{aligned} \dot{s}(t) &= v_1(s(t)) \\ \ddot{s}(t) &= v_2(s(t)) \\ &\vdots \\ s^{(p)}(t) &= v_p(s(t)) \end{aligned} \tag{4.54}$$

The definitions of v_1 and v_2 are also given in (2.4) and (2.5).

The path velocity controller (4.53) limits $\sigma^{(p)}$. When the limits on $\sigma^{(p)}$ are not active, the path velocity controller (4.53) reduces to the dynamic system

$$\begin{aligned} \frac{d\sigma}{dt} &= \dot{\sigma} \\ \frac{d\dot{\sigma}}{dt} &= \ddot{\sigma} \\ &\vdots \\ \frac{d\sigma^{(p-1)}}{dt} &= \sigma^{(p)} \\ \sigma^{(p)} &= \psi(\sigma, \ldots, \sigma^{(p-1)}) \end{aligned} \tag{4.55}$$

The feedback ψ is below chosen such that the dynamic system (4.55) becomes a linear system when σ is interpreted as time variable.

A Dynamic system in σ

The feedback ψ is derived from a nonlinear system of order $p-1$. This system was also derived in Chapter 3, section 3.5, and is given in (3.54). We will here use the same system, but with s replaced by σ. Introduce therefore

$$x_1 = \frac{\dot{\sigma}^p}{p} \tag{4.56}$$

which is (3.46) with s replaced by σ and

$$x_k = \frac{dx_{k-1}}{d\sigma}, \quad k = 2, \ldots, p-1 \tag{4.57}$$

which is (3.49) with s replaced by σ. Introducing the notation $u = \sigma^{(p)}$, the nonlinear dynamic system is given by (3.54) with s replaced by σ as

$$\begin{aligned} \frac{dx_1}{d\sigma} &= x_2 \\ \frac{dx_2}{d\sigma} &= x_3 \\ &\vdots \\ \frac{dx_{p-1}}{d\sigma} &= u - F_p(x_1, \dots, x_{p-1}) \end{aligned} \tag{4.58}$$

Feedback linearization

The feedback ψ is obtained by choosing u in (4.58) as a feedback linearizing controller. A feedback linearizing controller for the system (4.58) is given by

$$u = v + F_p(x_1, \dots, x_{p-1}) \tag{4.59}$$

where v is considered as a new input. Introduce a reference value $x_r(\sigma)$ by

$$\begin{aligned} x_r(\sigma) &= \begin{pmatrix} x_{r_1}(\sigma) & \dots & x_{r_{p-1}}(\sigma) \end{pmatrix} \\ x &= \begin{pmatrix} x_1 & \dots & x_{p-1} \end{pmatrix} \end{aligned}$$

A linear state feedback for v in (4.59) is then given by

$$v = \frac{dx_{r_{p-1}}(\sigma)}{d\sigma} + L(x_r(\sigma) - x) \tag{4.60}$$

The controller (4.59) with v given by (4.60) gives a linear system in σ. The system has the property that x_1 approaches $x_{r_1}(\sigma)$ with a σ-time constant depending on the feedback gains in the vector L.

Assuming $\dot{\sigma}$ is always positive, i.e. the reference trajectory is always moving forward along the path, we can then, since x_1 and $\dot{\sigma}$ are related as in (4.56), achieve $\dot{\sigma} \to v_1(\sigma)$ by choosing $x_{r_1}(\sigma)$ as

$$x_{r_1}(\sigma) = \frac{v_1(\sigma)^p}{p} \tag{4.61}$$

and

$$x_{r_k}(\sigma) = \frac{dx_{r_{k-1}}(\sigma)}{d\sigma}, \quad k = 1, \dots, p-1 \tag{4.62}$$

The feedback ψ

The final feedback law (4.51) is now obtained from (4.59) and (4.60) by expressing $x_1, \ldots, x_{p-1}$ as functions of $\dot{\sigma}, \ldots, \sigma^{(p-1)}$ and by expressing $x_{r_1}(\sigma), \ldots, x_{r_{p-1}}(\sigma)$, and $\frac{dx_{r_{p-1}}(\sigma)}{d\sigma}$ as functions of $v_1(\sigma), \ldots, v_p(\sigma)$. This is done as follows.

From (4.56) and (4.57), it is seen that (3.51) can be written, with s replaced by σ, as

$$\begin{aligned} \dot{\sigma} &= g(x_1) \\ \ddot{\sigma} &= g'(x_1)g(x_1)x_2 \end{aligned} \tag{4.63}$$

Using (3.50) and (4.57), higher order derivaties of σ can be expressed as

$$\begin{aligned} \sigma^{(3)} &= g'(x_1)g(x_1)^2 x_3 + F_3(x_1, x_2) \\ &\vdots \\ \sigma^{(p-1)} &= g'(x_1)g(x_1)^{p-2} x_{p-1} + F_{p-1}(x_1, \ldots, x_{p-2}) \\ \sigma^{(p)} &= g'(x_1)g(x_1)^{p-1} \frac{dx_{p-1}}{d\sigma} + F_p(x_1, \ldots, x_{p-1}) \end{aligned} \tag{4.64}$$

Introduce the function $\Phi(x_1, \ldots, x_{p-1}, \frac{dx_{p-1}}{d\sigma})$ by combining equations (4.63) and (4.64), i.e.

$$\begin{pmatrix} \dot{\sigma} \\ \vdots \\ \sigma^{(p)} \end{pmatrix} = \Phi(x_1, \ldots, x_{p-1}, \frac{dx_{p-1}}{d\sigma}) \tag{4.65}$$

Inversion of (4.65) now gives

$$\begin{pmatrix} x_1 \\ \vdots \\ x_{p-1} \\ \frac{dx_{p-1}}{d\sigma} \end{pmatrix} = \Phi^{-1}(\dot{\sigma}, \ldots, \sigma^{(p)}) \tag{4.66}$$

Using (4.66), $x_1, \ldots, x_{p-1}$ can be expressed as functions of $\dot{\sigma}, \ldots, \sigma^{(p-1)}$.

Equation (4.66) actually gives also $x_{r_1}(\sigma), \ldots, x_{r_{p-1}}(\sigma)$, and $\frac{dx_{r_{p-1}}(\sigma)}{d\sigma}$ as functions of $v_1(\sigma), \ldots, v_p(\sigma)$, i.e.

$$\begin{pmatrix} x_{r_1}(\sigma) \\ \vdots \\ x_{r_{p-1}}(\sigma) \\ \frac{dx_{r_{p-1}}(\sigma)}{d\sigma} \end{pmatrix} = \Phi^{-1}(v_1(\sigma), \ldots, v_p(\sigma)) \tag{4.67}$$

This is seen as follows. Introduce $\bar{x}_1, \ldots, \bar{x}_{p-1}$ by replacing σ by s in (4.56) and (4.57), i.e.

$$\bar{x}_1 = \frac{\dot{s}^p}{p}, \quad \bar{x}_k = \frac{d\bar{x}_{k-1}}{ds}, \quad k = 2, \ldots, p-1$$

Then (4.65) can be written, with σ replaced by s, as

$$\begin{pmatrix} \dot{s} \\ \vdots \\ s^{(p)} \end{pmatrix} = \Phi(\bar{x}_1, \ldots, \bar{x}_{p-1}, \frac{d\bar{x}_{p-1}}{ds}) \tag{4.68}$$

Using (4.54), and replacing σ by s in (4.61) and (4.62), then gives that (4.68) can be written as

$$\begin{pmatrix} v_1(s) \\ \vdots \\ v_p(s) \end{pmatrix} = \Phi(x_{r_1}(s), \ldots, x_{r_{p-1}}(s), \frac{dx_{r_{p-1}}(s)}{ds}) \tag{4.69}$$

Inverting (4.69) and replacing s by σ now gives (4.67).

Derivation of ψ The following algorithm can be used for derivation of the feedback law (4.51).

ALGORITHM 4.1—Derivation of ψ

1. Define $x_1, \ldots x_{p-1}$ as in (4.56) and (4.57).
2. Express $\dot{\sigma}, \ldots, \sigma^{(p)}$ as functions of $x_1, \ldots, x_{p-1}, \frac{dx_{p-1}}{d\sigma}$. This gives the function Φ in (4.65). It also gives the function F_p in (4.64), and hence the dynamic system (4.58) with $u = \sigma^{(p)}$ as input.
3. Choose u as a feedback linearizing controller (4.59) and (4.60). This gives $\sigma^{(p)}$ as a function of $x_1, \ldots, x_{p-1}$, $x_{r_i}(\sigma), \ldots, x_{r_{p-1}}(\sigma)$, and $\frac{dx_{r_{p-1}}(\sigma)}{d\sigma}$.
4. Invert (4.65) to to obtain (4.66). This gives $x_1, \ldots, x_{p-1}$ as functions of $\dot{\sigma}, \ldots, \sigma^{(p-1)}$
5. Replace $\dot{\sigma}, \ldots, \sigma^{(p)}$ in (4.66) by by $v_1(\sigma), \ldots, v_p(\sigma)$. Using (4.67), this gives $x_{r_i}(\sigma), \ldots, x_{r_{p-1}}(\sigma)$, and $\frac{dx_{r_{p-1}}(\sigma)}{d\sigma}$ as functions of $v_1(\sigma), \ldots, v_p(\sigma)$.

Path velocity control

Algorithm 4.1 is exemplified by deriving the path velocity controller (4.53) for the cases $p = 2$ and $p = 4$.

EXAMPLE 4.14
For $p = 2$ we get

$$x_1 = \frac{\dot{\sigma}^2}{2}$$

From the derivation of (3.55), we get, with s replaced by σ,

$$\begin{aligned}\dot{\sigma} &= (2x_1)^{1/2}\\ \ddot{\sigma} &= \frac{dx_1}{d\sigma}\end{aligned} \tag{4.70}$$

The dynamic system (4.58) is then given as (3.55) with s replaced by σ, i.e.

$$\frac{dx_1}{d\sigma} = u$$

where $u = \ddot{\sigma}$. The feedback ψ is in this case a linear feedback

$$u = \frac{dx_{r_1}(\sigma)}{d\sigma} + \alpha(x_{r_1}(\sigma) - x_1) \tag{4.71}$$

Inversion of (4.70) gives

$$\begin{aligned}x_1 &= \frac{\dot{\sigma}^2}{2}\\ \frac{dx_1}{d\sigma} &= \ddot{\sigma}\end{aligned} \tag{4.72}$$

Replacing $\dot{\sigma}$ and $\ddot{\sigma}$ in (4.72) by $v_1(\sigma)$ and $v_2(\sigma)$, gives, using (4.67),

$$\begin{aligned}x_{r_1}(\sigma) &= \frac{v_1(\sigma)^2}{2}\\ \frac{dx_{r_1}(\sigma)}{d\sigma} &= v_2(\sigma)\end{aligned}$$

From (4.71), the feedback ψ is now obtained as

$$\psi(\sigma, \dot{\sigma}) = v_2(\sigma) + \alpha(\frac{v_1(\sigma)^2}{2} - \frac{\dot{\sigma}^2}{2})$$

The resulting path velocity controller (4.53) then becomes

$$\begin{aligned}&\frac{d\sigma}{dt} = \dot{\sigma}\\ &\frac{d\dot{\sigma}}{dt} = \ddot{\sigma}\\ &u_r = v_2(\sigma) + \frac{\alpha}{2}(v_1(\sigma)^2 - \dot{\sigma}^2)\\ &\ddot{\sigma} = sat(u_r, \ddot{\sigma}_{min}(\beta_1, \beta_2), \ddot{\sigma}_{max}(\beta_1, \beta_2))\end{aligned}$$

which is the same as (4.24)

EXAMPLE 4.15
For $p = 4$ we get

$$x_1 = \frac{\dot{\sigma}^4}{4}$$

From the derivation of (3.56), we get, with s replaced by σ,

$$\begin{aligned}
\dot{\sigma} &= (4x_1)^{1/4} \\
\ddot{\sigma} &= x_2(4x_1)^{-1/2} \\
\sigma^{(3)} &= x_3(4x_1)^{-1/4} - 2x_2^2(4x_1)^{-5/4} \\
\sigma^{(4)} &= \frac{dx_3}{d\sigma} + 5(\frac{x_2^3}{8x_1^2} - \frac{x_2x_3}{4x_1})
\end{aligned} \tag{4.73}$$

The dynamic system (4.58) is then given as (3.56) with s replaced by σ, i.e.

$$\begin{aligned}
\frac{dx_1}{d\sigma} &= x_2 \\
\frac{dx_2}{d\sigma} &= x_3 \\
\frac{dx_3}{d\sigma} &= u - 5(\frac{x_2^3}{8x_1^2} - \frac{x_2x_3}{4x_1})
\end{aligned}$$

where $u = \sigma^{(4)}$. The feedback ψ is then given by the feedback linearizing controller (4.59) and (4.60) as

$$\begin{aligned}
u &= v + 5(\frac{x_2^3}{8x_1^2} - \frac{x_2x_3}{4x_1}) \\
v &= \frac{dx_{r_3}}{d\sigma} + \sum_{i=1}^{3} l_i(x_{r_i}(\sigma) - x_i)
\end{aligned} \tag{4.74}$$

Inversion of (4.73) gives

$$\begin{aligned}
x_1 &= \frac{\dot{\sigma}^4}{4} \\
x_2 &= \dot{\sigma}^2\ddot{\sigma} \\
x_3 &= \dot{\sigma}\sigma^{(3)} + 2\ddot{\sigma}^2 \\
\frac{dx_3}{d\sigma} &= \sigma^{(4)} + 5\frac{\ddot{\sigma}\sigma^{(3)}}{\dot{\sigma}}
\end{aligned} \tag{4.75}$$

Replacing $\dot{\sigma}, \ldots, \sigma^{(4)}$ in (4.75) by $v_1(\sigma), \ldots v_4(\sigma)$, gives, using (4.67),

$$\begin{aligned} x_{r_1}(\sigma) &= \frac{v_1(\sigma)^4}{4} \\ x_{r_2}(\sigma) &= v_1(\sigma)^2 v_2(\sigma) \\ x_{r_3}(\sigma) &= v_1(\sigma) v_3(\sigma) + 2v_2(\sigma) \\ \frac{dx_{r_3}(\sigma)}{d\sigma} &= v_4(\sigma) + 5\frac{v_2(\sigma)v_3(\sigma)}{v_1(\sigma)} \end{aligned} \tag{4.76}$$

From (4.74), the feedback ψ is now obtained as

$$\begin{aligned} \psi(\sigma, \dot{\sigma}, \ddot{\sigma}, \sigma^{(3)}) =& v_4(\sigma) + 5(\frac{v_2(\sigma)v_3(\sigma)}{v_1(\sigma)} - \frac{\ddot{\sigma}\sigma^{(3)}}{\dot{\sigma}}) \\ &+ l_1(\frac{v_1(\sigma)^4}{4} - \frac{\dot{\sigma}^4}{4}) + l_2(v_1(\sigma)^2 v_2(\sigma) - \dot{\sigma}^2\ddot{\sigma}) \\ &+ l_3(v_1(\sigma)v_3(\sigma) + 2v_2(\sigma) - \dot{\sigma}\sigma^{(3)} - 2\ddot{\sigma}^2) \end{aligned} \tag{4.77}$$

The resulting path velocity controller (4.53) then becomes

$$\begin{aligned} \frac{d\sigma}{dt} &= \dot{\sigma} \\ \frac{d\dot{\sigma}}{dt} &= \ddot{\sigma} \\ \frac{d\ddot{\sigma}}{dt} &= \sigma^{(3)} \\ \frac{d\sigma^{(3)}}{dt} &= \sigma^{(4)} \\ u_r &= \psi(\sigma, \dot{\sigma}, \ddot{\sigma}, \sigma^{(3)}) \\ \sigma^{(4)} &= sat(u_r, \sigma^{(4)}_{min}(\beta_1, \beta_2), \sigma^{(4)}_{max}(\beta_1, \beta_2)) \end{aligned}$$

where ψ is given by (4.77).

4.7 Conclusions

A path velocity controller can be used to utilize the available torque range by feedback modification of the reference trajectory. The modification is done such that the modified reference trajectory defines the same path as the nominal, which results in motion along the path, at the expense of increased traversal time.

The robot controller is parametrized in the scalar path parameter, but otherwise unchanged, i.e. a well tuned control behavior is kept. A basic algorithm (4.24) uses limits on acceleration together with feedback using the nominal velocity profile. The limits on acceleration are computed from the controller parametrization. When the acceleration saturates, the maximum or minimum acceleration that gives admissible torques is used. When the acceleration is not saturated, the feedback (4.20) gives a linear system with the path parameter σ as independent variable. The parameter α in the path velocity controller then determines a σ-time constant $\frac{1}{\alpha}$, and can be tuned by inspection of the nominal velocity profile. An extension to higher order systems shows that the feedback can be chosen such that the resulting system becomes linear also for higher order path velocity controllers. A constant time scaling of the nominal path parameter can be introduced as a constant scaling of the velocity profile as in (4.26). An extended algorithm (4.29) uses feedback for modification of the scaling factor. An approximate analysis of constant scaling gives insight into what type of model errors that can be handled by the path velocity controller, and it was e.g. exemplified how underestimation of friction can be handled, Examples 4.5 and 4.7.

The design of the path velocity controller reflects properties of the minimum time solution. The nominal minimum time solution has maximum and minimum path acceleration. Modeling errors may then result in the nominal acceleration being inadmissible. The basic algorithm (4.24) handles this problem by limiting the acceleration, using a controller parametrization (4.1), of the same type as the parametrization (3.11) of the robot model used in the minimum time optimization. The minimum time velocity profile generally touches the maximum velocity curve. Model errors may then result in inadmissible path velocities around the touching points. The extended algorithm (4.29) handles the added problem of inadmissible velocities using velocity profile scaling.

The path velocity controller was shown to work well in simulation examples. It was shown how modeling errors result in path deviation, when the path velocity controller is not present, e.g. Figure 4.5, and also how the path velocity controller can reduce the path deviation, e.g. Figure 4.6. The use of path velocity control thus adds robustness to the minimum time solution, in the sense that the sensitivity to modeling errors is reduced.

5 Experimental Evaluation

The path velocity controller is experimentally evaluated on an industrial robot. The original robot control system has been reconfigured, and an external computer is used for the control. Evaluation measures are discussed and proposed, and experiments illustrating the performance of the path velocity controller are presented.

The purpose of path velocity control is to achieve fast motion along a path. A measure of path deviation therefore has to be considered. The paths used in the experiments are measured using only joint positions, and there are no external sensors, e.g. for measuring the actual position of the end-effector. The path deviation is measured using visual inspection, either in joint space or in cartesian space, where the paths in cartesian space are obtained from the joint space paths using forward kinematics.

One also has to consider a measure of fast motion. A measure of the utilization of the torque range is proposed for this purpose. The measure is denoted τ_u, and satisfies $0 \leq \tau_u \leq 1$. The torque utilization τ_u achieves its maximum value when the motion has bang-bang character, i.e. $\tau_u = 1$ when one torque is always at the limit.

Another aspect of the evaluation is the possibility to evaluate the path velocity controller separately from the robot controller. In the simulation in Figure 2.7, the torque limits were removed, and it was possible to investigate the effect of the modeling error on the robot controller. The torque limits can however not be removed in experiments. Instead, two different levels of torque limits are used. The lower level is used in the minimum time optimization, and when using the path velocity controller. The upper level represents the condition of removed torque limits, and is used to check the performance of the robot controller. The purpose is that the torques should never reach the upper limit. The path velocity controller is then evaluated by comparing the path deviation when the path velocity controller is used, with the path deviation that occurs when the torque limits are set to the upper level. The performance of the path velocity controller is considered acceptable, if the path deviation for the two cases is comparable. The path velocity controller has then adjusted the reference trajectory such that the path error is of the same magnitude as when there are no torque limits, and the remaining path error is due to the robot controller.

Having decided upon evaluation strategy, one has to select proper experiments. One type of experiments is to first compute a nominal velocity profile by minimum time optimization, and then investigate the effect of using this nominal velocity profile without the path velocity controller. The path velocity controller is then used, and the result is evaluated. Three experiments of this type are presented. The first experiment uses a path where only two joints move. This shows more clearly effects which also can be seen in simulation. The second and third paths are specified in cartesian space, and three joints are involved in the motion. The two paths result in different behavior with respect to both path deviation and path velocity control performance.

Another effect is time varying robot dynamics. An experiment where the path velocity controller is used to handle this is presented. The experiment is done by repeated motion along a path. The motion is started after having the robot kept at zero velocity for approximately 5–10 minutes. This has the effect that as time elapses, the robot is able to move faster, probably due to decreased friction. The result of using path velocity control is then that the velocity of the reference trajectory increases, while path following is maintained.

In the simulations, model errors were introduced such that the nominal motion specification should result in inadmissible torques, e.g. by increasing mass or friction in the model used in the simulation. The nominal motion specification in the previously discussed experiments is also of this type. It could however also happen that the nominal model is too conservative, i.e. the minimum time velocity profile computed from the model results in good path following, but unsatisfactory utilization of the available torque range. One alternative to increase the performance in such a situation is to use velocity profile scaling in the path velocity controller as in (4.26) to increase the velocity, i.e. the opposite to what was discussed in Chapter 4, where the algorithm (4.29) decreases the scaling factor. It is demonstrated in an experiment how this results in smaller traversal time compared to the nominal. The low utilization of the torque range can however be seen as an indication that the dynamic model can be improved. It is shown in an experiment that if the nominal velocity profile is computed from a different model, better performance is achieved, in the sense that the same traversal time as when the velocity profile scaling was used is obtained, but with both smaller path deviation and larger torque utilization.

The path velocity controllers (4.24) and (4.26) are used in the experiments. The nominal velocity profile is computed by minimum time optimization using Algorithm 3.1. The model used in the optimization is a decoupled linear robot model, obtained from experiments on the robot using system identification for estimation of the model parameters. A short description of the experimental environment is given in Section 5.1. The evaluation measures are described in Section 5.2, and the experimental results are presented in Section 5.3.

5.1 Experimental Environment

The experiments were done on an industrial robot of the type ABB Irb-6. The robot control system has been reconfigured, and the resulting system provides general interfaces that allow control using external computers [Braun *et al.*, 1990]. The computer system used in the experiments is a VME-based system consisting of a Motorola M68030 processor and an AT&T digital signal processor (DSP). The VME-system is connected to a Sun workstation using Ethernet. The software used in the experiments is written mainly in C++ and Modula-2. The path velocity controller is written in C++, and runs in the DSP. The implementation is described in Appendix A. The M68030 processor handles the operator interface, and the data that should be sent to the Sun for plotting and/or storage in files. A facility to connect Matlab running on the Sun to the VME system is used for the data transfer [Nilsson, 1992b]. The system software for the DSP is described in [Mattsson, 1991]. The software for the Motorola processor is based on a real time kernel developed at the Department of Automatic Control [Andersson and Blomdell, 1991]. A system for real time implementation of control algorithms [Dahl, 1991] is used as a software tool for the other parts of the Motorola software, e.g. for parameter editing in the running controller in the DSP. A schematic diagram of the system is shown in Figure 5.1. The robot is shown in Figure 5.2.

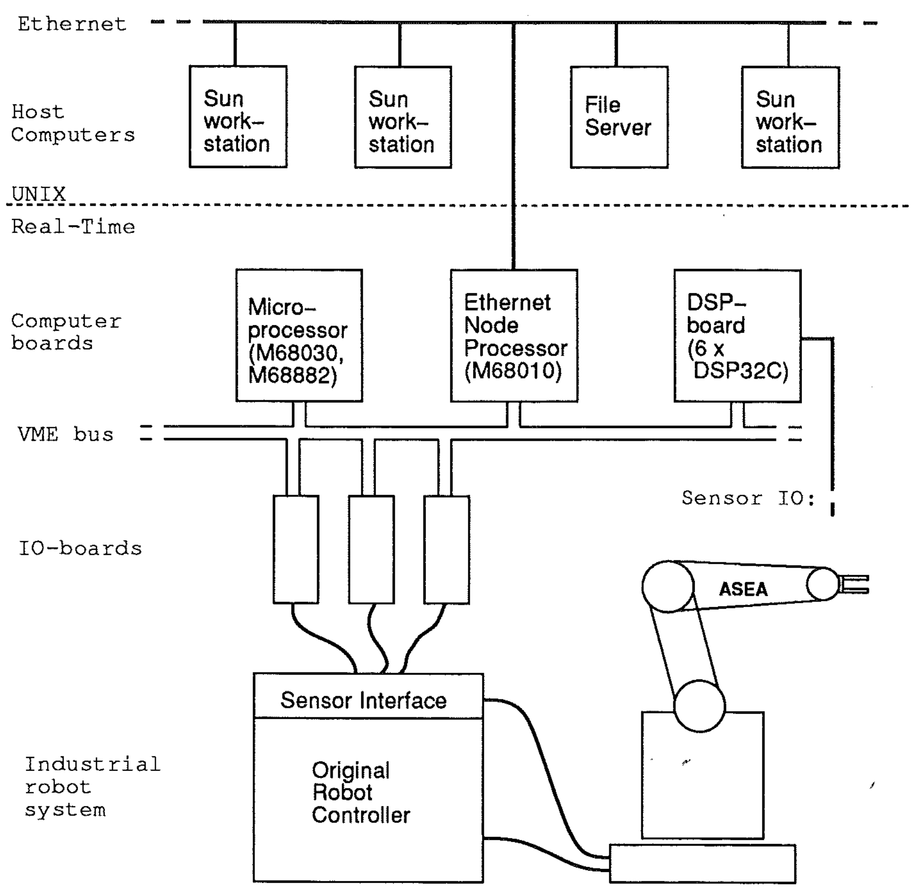


Figure 5.1 A schematic description of the experimental environment.

5.2 Experimental Evaluation

This section describes the chosen criteria for evaluation. It also describes the robot models used, and the robot controller.

Evaluation measures

The purpose of using the path velocity controller is to achieve fast motion along the path. Our main criteria is therefore path deviation, which is measured using visual inspection. As a measure of the closeness to the unknown minimum time, we measure the utilization of the torque range. The torque utilization τ_u is computed as

$$\tau_u = 1 - \frac{1}{N}\sum_{t=1}^{N} \min_{1\leq i\leq n} \left| \frac{\tau_i(t) - \tau_i^m}{\tau_i^m} \right| \tag{5.1}$$

where N is the number of data points, and n is the number of joints. For the robot used here, $n = 5$. The notation τ_i^m denotes here the torque limit

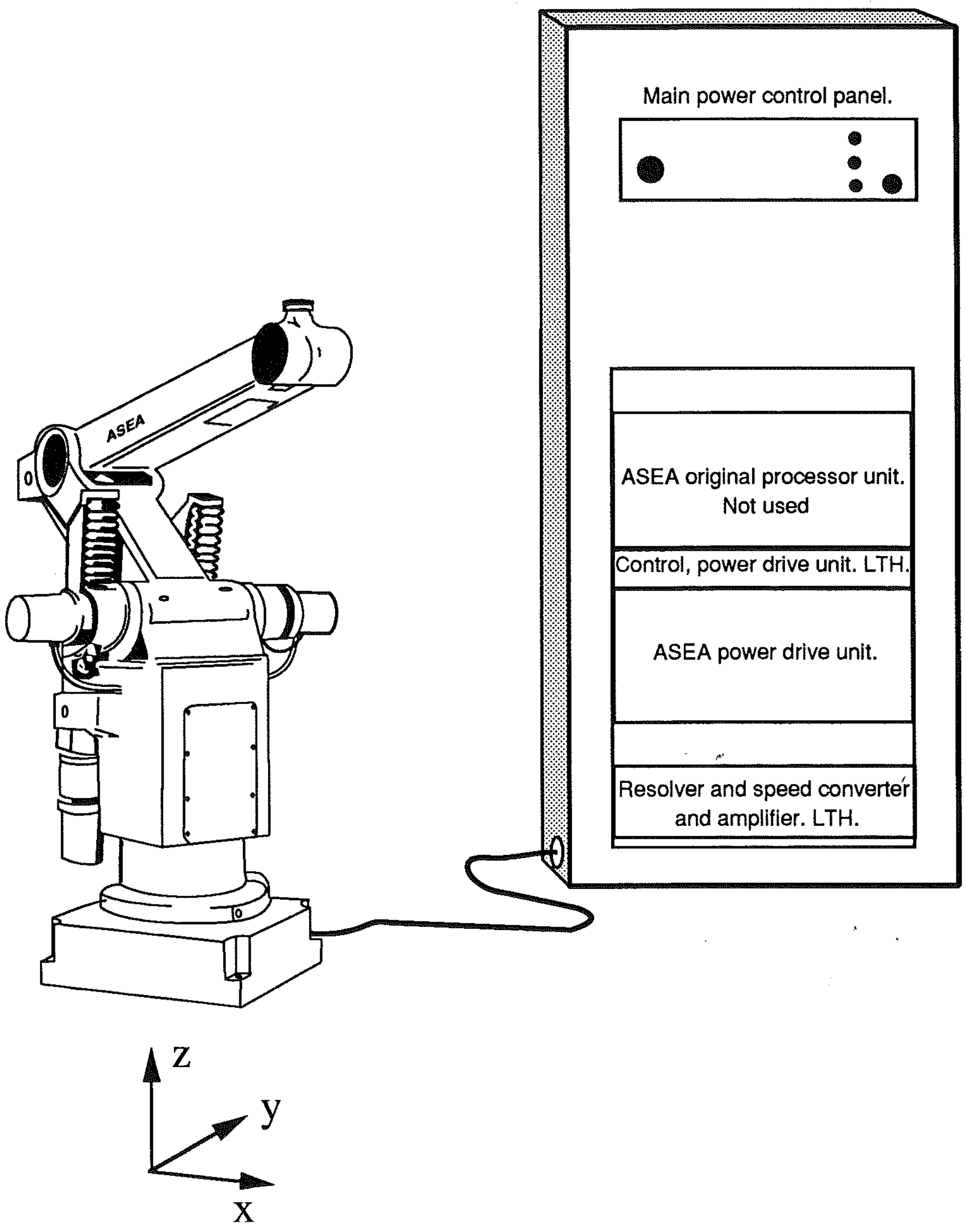

Main power control panel.
ASEA original processor unit. Not used
Control, power drive unit. LTH.
ASEA power drive unit.
Resolver and speed converter and amplifier. LTH.
ASEA
z
y
x

Figure 5.2 The robot used in the experiments. The coordinate axes in cartesian space are also shown. The origin of the cartesian space is at the base of the robot.

which is closest to τ_i. Note that for minimum time motion, where one joint is always on the limit, $\tau_u = 1$.

An indirect measure of the path deviation is found by computing the mean square tracking error. The mean square tracking error in joint space is denoted *mse*, and is computed as

$$mse = \frac{1}{N}\sum_{t=1}^{N}\sum_{i=1}^{n}(q_{r_i}(t) - q_i(t))^2 \tag{5.2}$$

The corresponding quantity in cartesian space, denoted *cmse*, is computed as

$$cmse = \frac{1}{N}\sum_{t=1}^{N}(x_{r_i}(t) - x_i(t))^2 + (y_{r_i}(t) - y_i(t))^2 + (z_{r_i}(t) - z_i(t))^2 \tag{5.3}$$

where the cartesian values are obtained using forward kinematics. The cartesian coordinate system is defined in Figure 5.2. The quantities *mse* (5.2) and *cmse* (5.3) are indirect measures of path deviation, in the sense that small values of *mse* and *cmse* result in small path deviation, but not vice versa, e.g. when the motion is along a line and the robot is behind the reference trajectory, we get large tracking errors but zero path deviation.

In the experiments, the traversal time is also computed. The traversal time is computed as the time when the reference trajectory reaches the end of the path. The traversal time t_f is thus given from $\sigma(t_f) = s_f$.

Evaluation from limits on path acceleration Another evaluation measure is obtained using properties of the minimum time solution. When the limits on $\ddot{\sigma}$ (4.14) are inadmissible, the path velocity controller (4.24) cannot adjust the path acceleration such that all torques are admissible, which then may result in path deviations. From the characterization of the admissible region in Section 3.3, it is seen that for velocities above the maximum velocity curve, with the exception of critical points, the limits on the nominal path acceleration (3.26) are inadmissible, i.e. $\ddot{s}_{min}(s,\dot{s}) > \ddot{s}_{max}(s,\dot{s})$. Inadmissibility of the limits on the actual path acceleration (4.14), i.e. $\ddot{\sigma}_{min}(\beta_1,\beta_2) > \ddot{\sigma}_{max}(\beta_1,\beta_2)$ can therefore be seen as an indication that the velocity is too high.

An evaluation of the result of using path velocity control can therefore be done by checking the admissibility of the limits afterwards. If the limits were inadmissible, this indicates that the nominal velocity was too high. Moreover, the inadmissibility can also be seen as a possible explanation of path deviations caused by a too high nominal velocity profile, i.e. path deviations that can be eliminated if the nominal velocity profile is modified, e.g. by using the path velocity controller (4.26) with a scaling factor $\gamma < 1$.

In the presentation of the experiments in Section 5.3, the admissibility of the limits is checked by plotting the limits on path acceleration (4.14) together with the actual path acceleration $\ddot{\sigma}$. This also makes it possible to investigate when the limits were used, and also which of the limits that were activated.

Robot Controller

The robot controller is an individual joint PD controller, written as

$$\tau_i = \hat{m}_i \ddot{q}_{r_i} + \hat{d}_i \dot{q}_{r_i} + k_{v_i} \dot{e}_i + k_{p_i} e_i, \quad 1 \leq i \leq 5 \tag{5.4}$$

where $e_i = q_{r_i} - q_i$. The feedback gains k_{p_i} and k_{v_i}, $1 \leq i \leq 5$, in (5.4) were empirically tuned. There were some small differences in the feedback gains between different experiments, but a typical choice is

$$\begin{aligned} &k_{p_1} = 0.14,\ k_{p_2} = 0.15,\ k_{p_3} = 0.14,\ k_{p_4} = 0.05,\ k_{p_5} = 0.05 \\ &k_{v_1} = 0.02,\ k_{v_2} = 0.015,\ k_{v_3} = 0.02,\ k_{v_4} = 0.01,\ k_{v_5} = 0.01 \end{aligned}$$

Robot models

In the experiments, only joints 1 to 3 are active in the motion. The reference values for joints 4 and 5 are zero. System identification [Ljung, 1991] was used to obtain the parameters in a linear decoupled model on the form

$$m_i \ddot{q}_i + d_i \dot{q}_i = \tau_i, \quad i = 1, 2, 3 \tag{5.5}$$

This model is used in the minimum time optimization. It also gives the feedforward parameters $\hat{m}_i$ and $\hat{d}_i$, $1 \leq i \leq 3$, in (5.4). The feedforward parameters for joints 4 and 5 were set to zero.

The identification was done in closed loop. A prbs test signal was first used as reference value. The identification was done by parametric identification of a first order discrete time transfer function for each joint. The discrete model was then converted to a continuous time model. The sampling interval used in the identification was 10 ms. The identification results were

$$\begin{aligned} &m_1 = 6.6840 \cdot 10^{-4}, \quad m_2 = 4.8197 \cdot 10^{-4}, \quad m_3 = 0.0011 \\ &d_1 = 0.0027, \quad d_2 = 0.0034, \quad d_3 = 0.0060 \end{aligned} \tag{5.6}$$

As a first test, a model with the parameters (5.6) was used to compute a minimum time velocity profile for some test paths. The result was a conservative motion, with unsatisfactory torque utilization. A new model was then identified, using the minimum time velocity profile obtained from (5.6) as reference trajectory. The result of this identification was

$$\begin{aligned} &m_1 = 6.0429 \cdot 10^{-4}, \quad m_2 = 4.0817 \cdot 10^{-4}, \quad m_3 = 6.7348 \cdot 10^{-4} \\ &d_1 = 0.0019, \quad d_2 = 0.0024, \quad d_3 = 0.0059 \end{aligned} \tag{5.7}$$

Experiment 1	Nominal	With limits	With PVC
Path following	Figure 5.4	Figure 5.5	Figure 5.6
τ_u	-	0.99	0.94
t_f	1.59	1.59	1.83
mse	2.33	145	1.98
$cmse$	32.2	5040	31.2

Table 5.1 The result of experiment 1

Experiment 2	Nominal	With limits	With PVC
Path following	Fig. 5.12–13	Fig. 5.14–15	Fig. 5.16–17
τ_u	-	0.92	0.90
t_f	4.57	4.57	4.95
mse	3.35	94.1	3.16
$cmse$	46.6	3160	38.3

Table 5.2 The result of experiment 2

5.3 Experimental Results

The experimental results are presented in Experiments 1–5 in this section. The nominal velocity profiles in Experiments 1–4 were computed by minimum time optimization using the robot model (5.7). In Experiment 5, a nominal velocity profile based on the model (5.6) is also used.

For all experiments, the traversal times are measured in seconds, the joint variables are given in radians for the motor shaft, and the cartesian values are given in millimeters. The lower level of torque limits were chosen as ± 0.5 for all joints. The upper level of torque limits were ± 1 for all joints. The parameter choice $\alpha = 10$ in the path velocity controller (4.24) was used in all experiments. The sampling frequency for the robot controller and the path velocity controller was 1 Khz, and the data logging was done with 100 Hz. Experiments 1 to 3 are summarized in Tables 5.1–5.3. The first row refer to the Figures that are used for evaluation of the path following by visual inspection. The following rows show the torque utilization τ_u, computed as in (5.1), the traversal times, and the mean square tracking errors mse and $cmse$, computed as in (5.2) and (5.3). The first column shows the result of using the nominal velocity profile "without" the torque limits, i.e. with the torque limits ± 1. The torque utilization (5.1) is not defined for this case. In column 2, the nominal velocity profile is used together with the torque limits ± 0.5. Column 3 shows the result when the path velocity controller is used.

Experiment 3	Nominal	With limits	With PVC
Path following	Fig. 5.22–23	Fig. 5.24–25	Fig. 5.26–27
τ_u	-	0.95	0.93
t_f	3.40	3.40	3.78
mse	3.16	40.8	3.36
$cmse$	27.0	604.8	32.5

Table 5.3 The result of experiment 3

Experiment 1 – Ellips in joint space

The first experiment is evaluated in joint space. The path is an ellips in joint space for joints 1 and 2, parametrized as

$$f_1(s) = 50(1 - \cos(s)), \quad f_2(s) = 20\sin(s), \quad 0 \le s \le 2\pi \tag{5.8}$$

The nominal velocity profile was computed using minimum time optimization. The model parameters are given in (5.7). The path and the minimum time solution are shown in Figure 5.3. The path (5.8) is shown in the upper plot. The mid plot shows the minimum time velocity profile, solid line, and the maximum velocity curve, dashed line. The dotted line in the mid plot is 2 if $\ddot{s}$ is maximum, and 0 if $\ddot{s}$ is minimum. The lower plot shows the torques τ_1, solid line, and τ_2, dashed line, as functions of s.

An experiment where the nominal velocity profile is used, and where the torque limits are removed (actually set to ± 1) is shown in Figure 5.4. The upper right plot shows the path velocity $\dot{s}(t)$, dashed line, and the path parameter $s(t)$, solid line. The path parameter $s(t)$ is reset to zero at the end of the path. The path velocity $\dot{s}(t)$ in this plot is the time function corresponding to the velocity profile $v_1(s)$ in Figure 5.3. The lower left plot shows the reference trajectories $q_{r_i}(t) = f_i(s(t))$, $i = 1, 2$, and the corresponding actual trajectories $q_i(t)$.

The lower right plot shows the torques $\tau_1(t)$, solid line, and $\tau_2(t)$, dashed line. As can be seen from the plot, the torques required to track the nominal trajectory are outside the limits ± 0.5. It can also be seen that the torque τ_1 is for some intervals actually limited by the limits ± 1, i.e. the assumption of an experiment where the torque limits are removed is violated, however only for a small part of the motion.

The upper left plot shows the desired path, solid line, and the actual path, dashed line.

The result of using the nominal velocity profile and the torque limits ± 0.5 is shown in Figure 5.5. The upper right plot shows the path velocity $\dot{s}(t)$, dashed line, and the path parameter $s(t)$, solid line. The lower left plot shows the reference trajectories $q_{r_i}(t) = f_i(s(t))$, $i = 1, 2$, and the corresponding

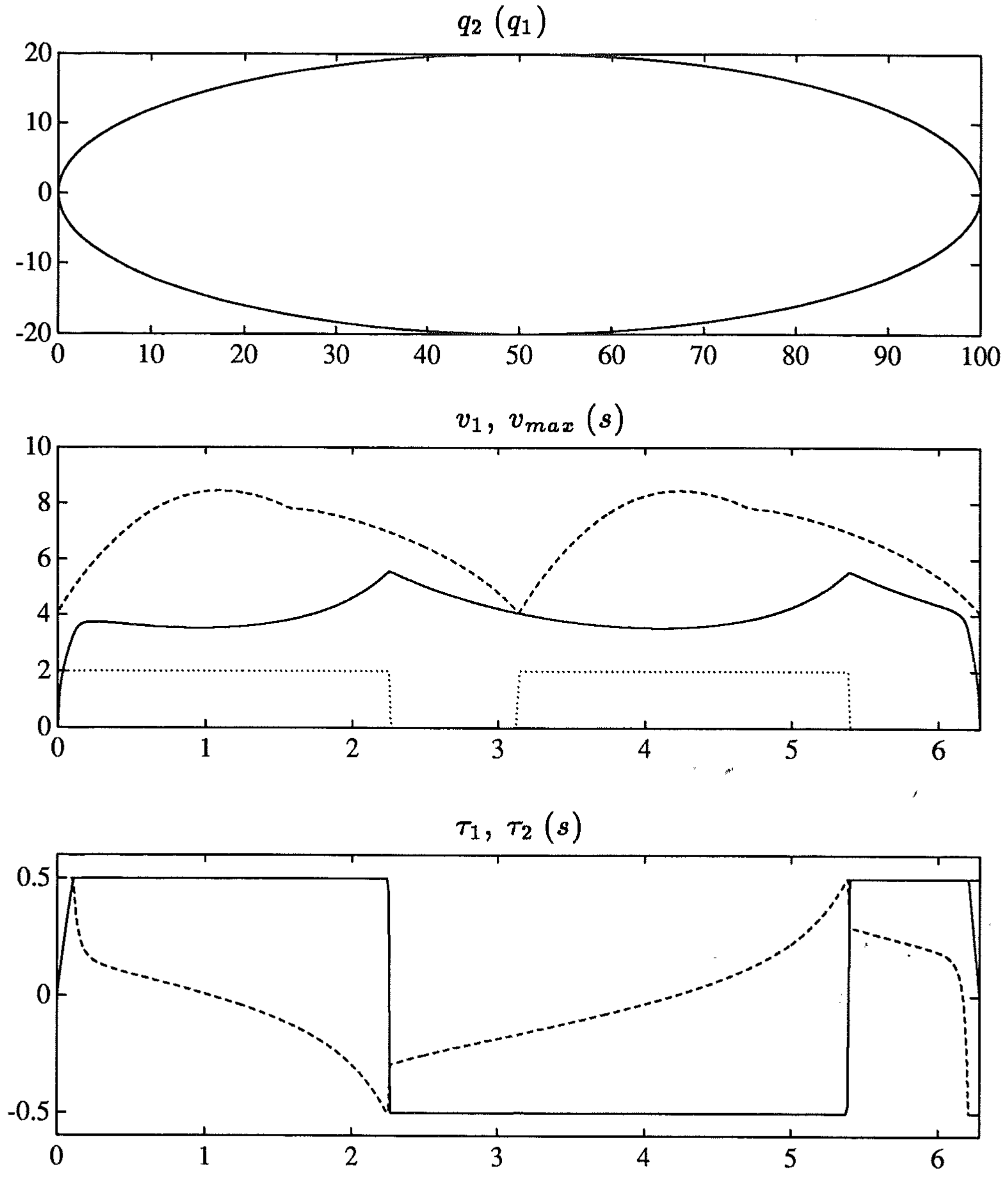


Figure 5.3 The minimum time solution for Experiment 1, mid plot, and the corresponding torques, lower plot. The path in joint space is shown in the upper plot.

actual trajectories $q_i(t)$. The solid line is the reference trajectory $q_{r_1}(t)$, and the dashed line is the actual trajectory $q_1(t)$. As can be seen in the plot, the trajectory $q_1(t)$ deviates from the reference trajectory $q_{r_1}(t)$.

The lower right plot shows the torques $\tau_i(t)$, $i = 1, 2$. As can be seen in the plot, the torque τ_1, solid line is saturated for most parts of the motion. The torque τ_2 is also saturated, but for shorter time intervals. From Figure

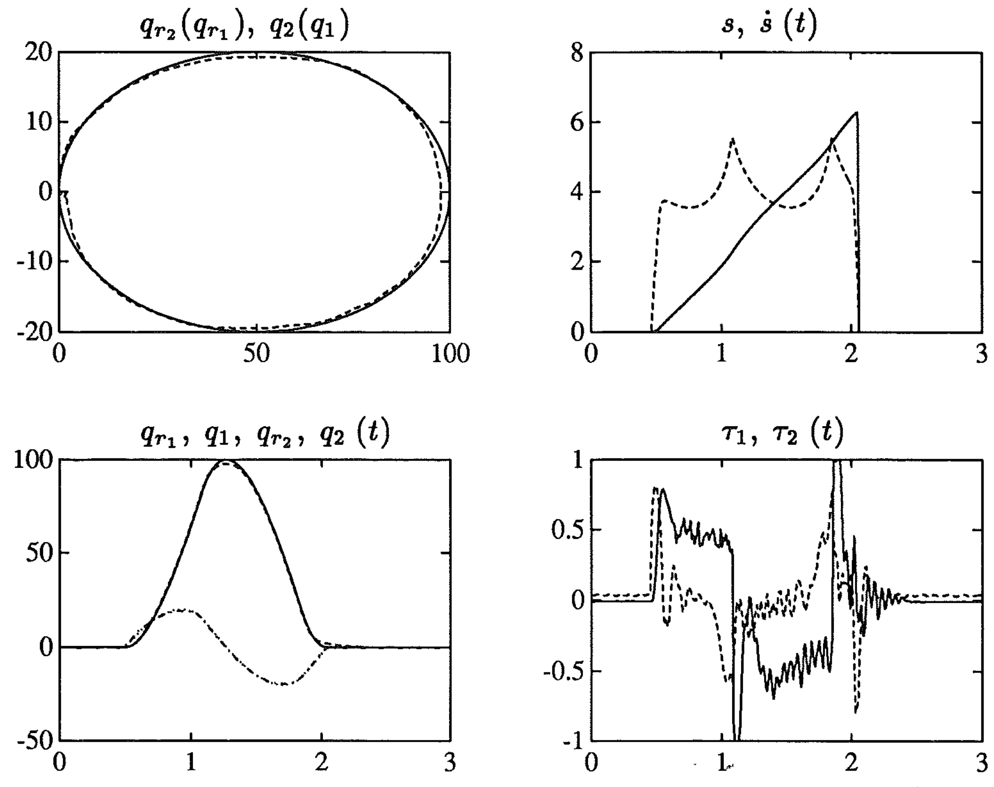


Figure 5.4 The result of using the nominal velocity profile without the torque limits. The traversal time was $t_f = 1.59$ and the mean square tracking error was $mse = 2.33$.

5.3 it is seen that for the initial part of the motion ($s \approx 0 - 0.2$), the nominal τ_1 and τ_2 are synchronized in the sense that τ_2 leaves the upper torque limit as soon as τ_1 reaches the limit. In the lower right plot in Figure 5.4, $t \approx 0.8$, it is seen that τ_2 remains saturated after τ_1 reaches the limit. A similar phenomenon is seen around $t = 1.3$, where τ_1 remains saturated after τ_2 leaves the lower limit. This is not the case for the nominal torques, shown in Figure 5.3. This type of behavior was also observed in simulation, see e.g. the lower right plot in Figure 4.5.

The upper left plot shows the desired path, solid line, and the actual path, dashed line. As can be seen in the figure, the path cannot be followed, a result of the large tracking error in joint one.

Using the path velocity controller (4.24) gives the result shown in Figure 5.6. The upper right plot shows the actual path velocity $\dot{\sigma}(t)$, dashed line, and the actual path parameter $\sigma(t)$, solid line. A comparison with the corresponding plot in Figure 5.5 shows that the path velocity is reduced, resulting in increased traversal time.

The lower left plot shows the reference trajectories $q_{r_i}(t) = f_i(\sigma(t))$,

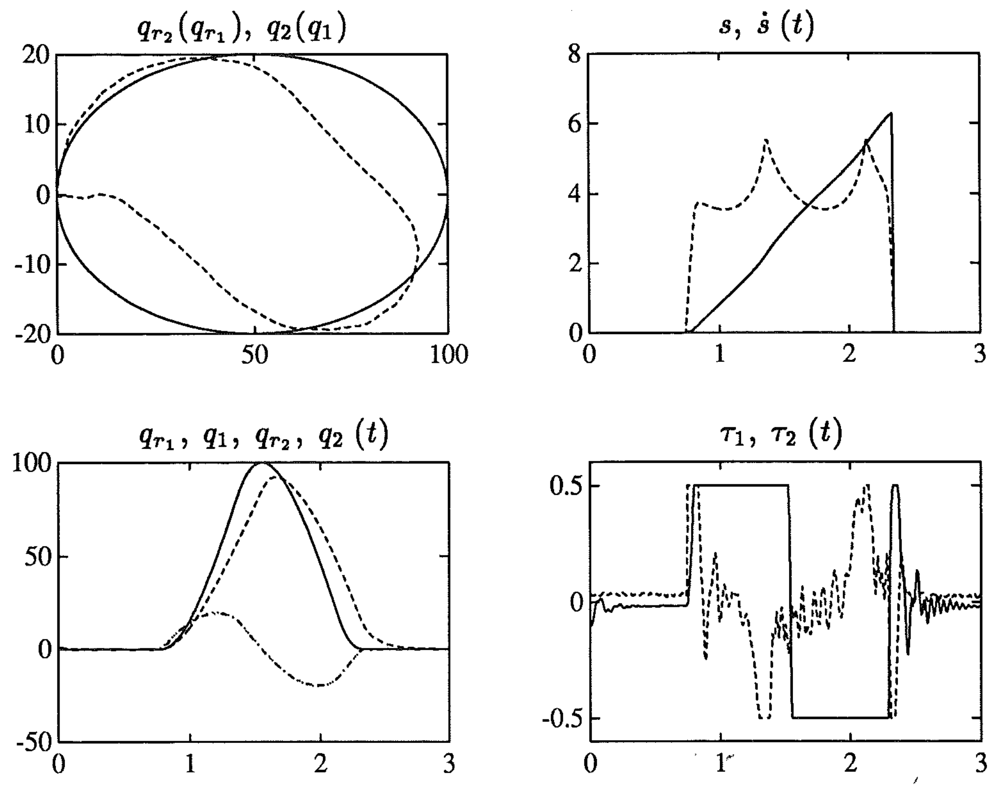


Figure 5.5 The result when the torques are limited. The torque utilization was $\tau_u = 0.99$, the traversal time was $t_f = 1.59$, and the mean square tracking error was $mse = 145$.

$i = 1, 2$, and the corresponding actual trajectories $q_i(t)$. A comparison with the corresponding plot in Figure 5.4 shows that the tracking performance when the path velocity controller is used is similar as for the case of removed torque limits.

The torques are shown in the lower right plot. A comparison with the nominal torques, shown in Figure 5.3, shows that the torques are now synchronized in the initial path of the motion, i.e. τ_2 leaves the upper limit, when τ_1 reaches the upper limit. It is also seen that the torques are synchronized for $t \approx 1.5$, where τ_1 leaves the upper limit when τ_2 changes direction. This was not the case when the nominal velocity profile was used, as can be seen in the corresponding plot in Figure 5.5. There is also a time interval, around $t \approx 1.6$, where none of the torques are at the limit.

The upper left plot shows the desired and the actual paths. As can be seen from the figure, the path following performance is comparable to Figure 5.4. The numerical results corresponding to Figure 5.6 are given in Table 5.1, the right column. As can be seen in Table 5.1, the *mse* and the *cmse* are of the same magnitude as for the nominal motion.

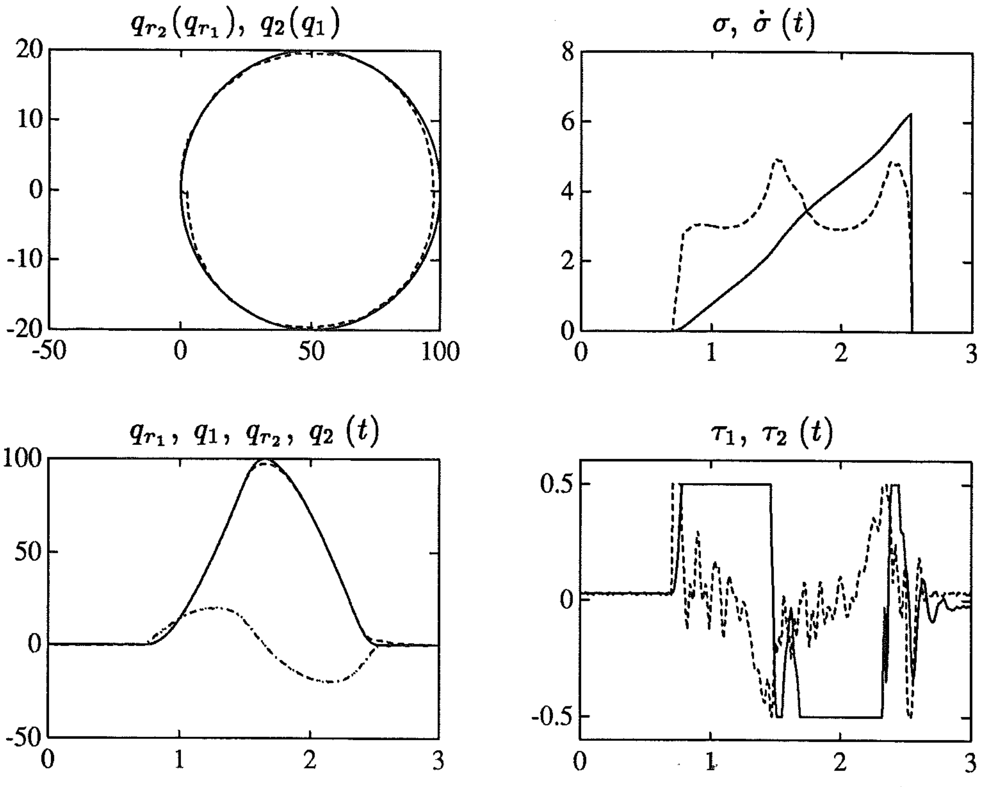


Figure 5.6 The result when the path velocity controller is used. The torque utilization was $\tau_u = 0.94$, the traversal time was $t_f = 1.83$, and the mean square tracking error was $mse = 1.98$

Figure 5.7 shows the actual paths from Figures 5.4 and 5.6. The solid line is the desired path, the dashed line is the path obtained when the path velocity controller is used, and the dotted line is the path obtained from the experiment in Figure 5.4 where the torque limits were removed. As can be seen in the figure, both the dashed line and the dotted line deviates from the desired path. However, the deviations are similar. This shows that when the path velocity controller is used, the behavior with respect to path deviation is qualitatively the same as for the case when the torque limits are removed. This can to some extent be expected, since the path velocity controller has the property that when the robot controller requires more torque than what is available, this is compensated by adjusting the path acceleration. Thus, as long as the limits on $\ddot{\sigma}$ are admissible, then from the robot controller's point of view, there are no torque limits.

An evaluation plot for the case when the path velocity controller is used is given in Figure 5.8. The dashed line in the upper plot is the nominal velocity profile $v_1(\sigma)$. The solid line in the upper plot is the actual velocity profile, i.e. $\dot{\sigma}$ as a function of σ. The dotted line in the upper plot indicates if the

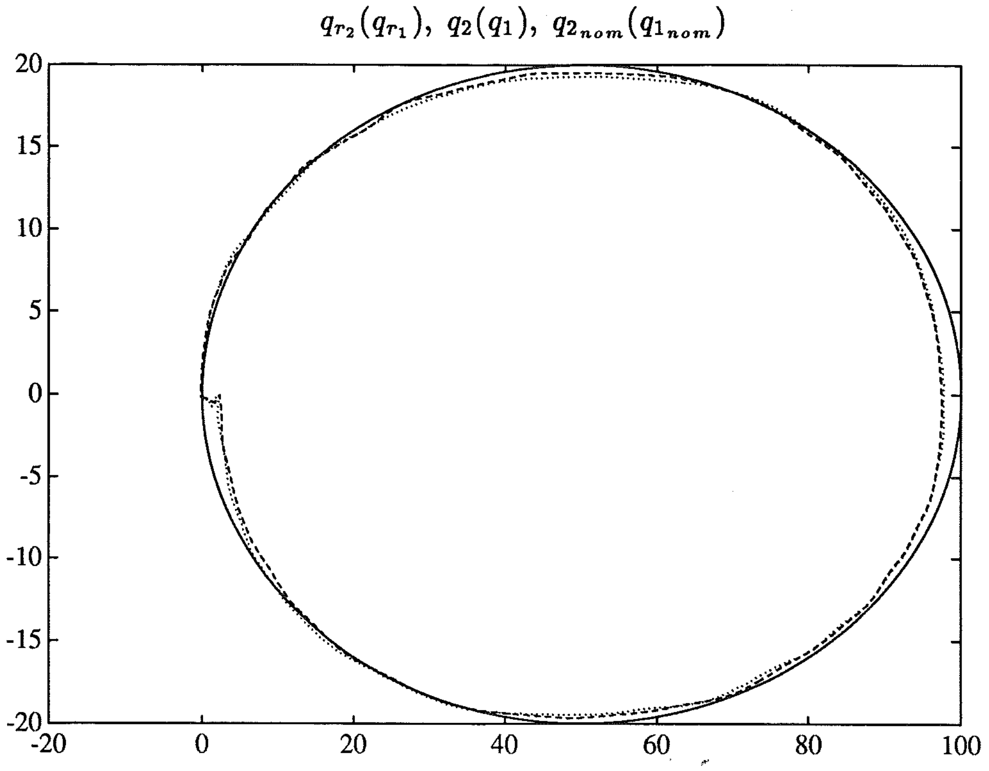


Figure 5.7 A comparison showing the actual path from the experiment without the torque limits, dotted line, and the actual path when the path velocity controller is used, dashed line. The solid line is the desired path.

limits on $\ddot{\sigma}$ are used or not, i.e. the dotted line is one if the limits are used, and zero otherwise. The lower plot shows the limits on path acceleration (4.14), and the path acceleration $\ddot{\sigma}$, as functions of σ. The solid line is $\ddot{\sigma}_{max}$ and the dashed line is $\ddot{\sigma}_{min}$. The dotted line is the path acceleration $\ddot{\sigma}$. The nominal acceleration and deceleration intervals are shown in Figure 5.3 by the dotted line in the mid plot. From the dotted line in the upper plot in Figure 5.8, it is seen that the limits are used during the acceleration intervals. From the lower plot in Figure 5.8, it is seen that the upper limit is active, i.e. $\ddot{\sigma} = \ddot{\sigma}_{max}$ in these intervals. It can also be seen that the lower limit $\ddot{\sigma}_{min}$ is active during parts of the deceleration intervals. For this experiment, the limits are admissible, i.e. $\ddot{\sigma}_{min} < \ddot{\sigma}_{max}$, for the complete motion. When the limits are admissible, the lower plot in Figure 5.8 is related to the torques, shown in the lower right plot in Figure 5.6, in the sense that in the intervals where the limits are active, i.e. when either $\ddot{\sigma} = \ddot{\sigma}_{min}$ or $\ddot{\sigma} = \ddot{\sigma}_{max}$, at least one of the torques are at the limit, and, conversely, when $\ddot{\sigma}_{min} < \ddot{\sigma} < \ddot{\sigma}_{max}$, then none of the torques are at the limit.

Figure 5.9 shows the nominal path acceleration $v_2(\sigma)$, dashed line, and

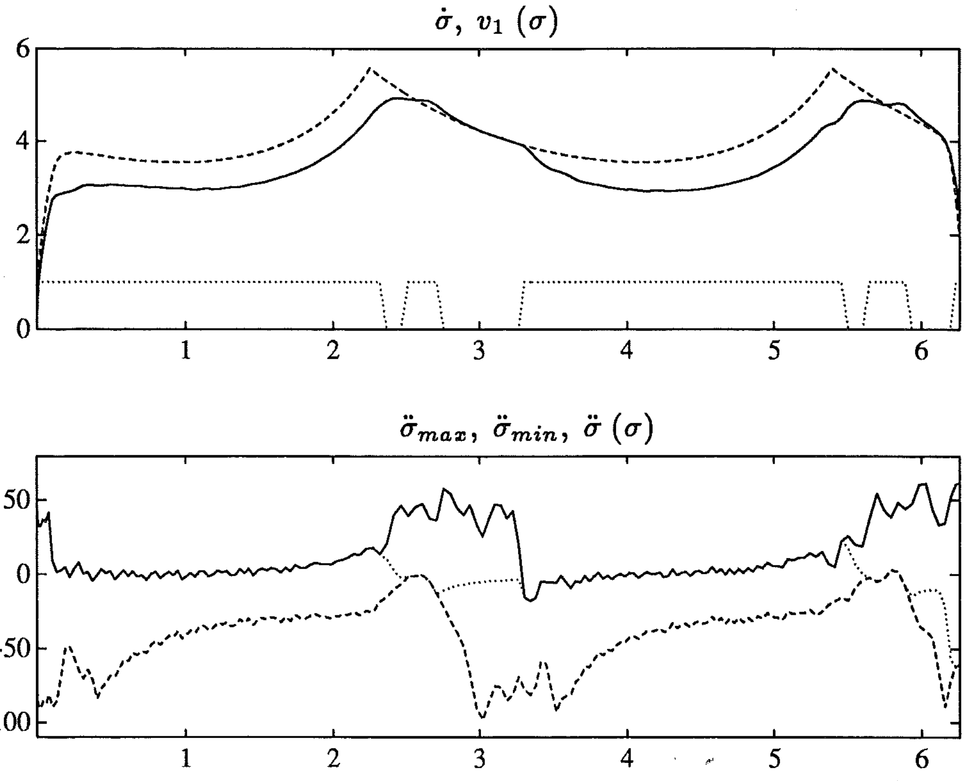


Figure 5.8 Evaluation of Experiment 1. The nominal and actual velocity profiles are shown in the upper plot. The path acceleration $\ddot{\sigma}$ and the limits $\ddot{\sigma}_{min}$ and $\ddot{\sigma}_{max}$ are shown in the lower plot.

the actual path acceleration $\ddot{\sigma}$ as a function of σ, solid line. Consider e.g. the interval $1 \leq \sigma \leq 2$. From the lower plot in Figure 5.8, it is seen that the upper limit on acceleration is active in this interval. In Figure 5.9, it is seen that the actual acceleration $\ddot{\sigma}$ is rapidly varying, but the nominal acceleration is smooth. This is expected, since the activation of the upper limit has the effect that the torque is kept at maximum, and the variations that would have been variations in the torque in the case of no torque limits, are instead seen as variations in $\ddot{\sigma}$. Thus, the variations in $\ddot{\sigma}$ in Figure 5.9 correspond to variations in the torques in Figure 5.4.

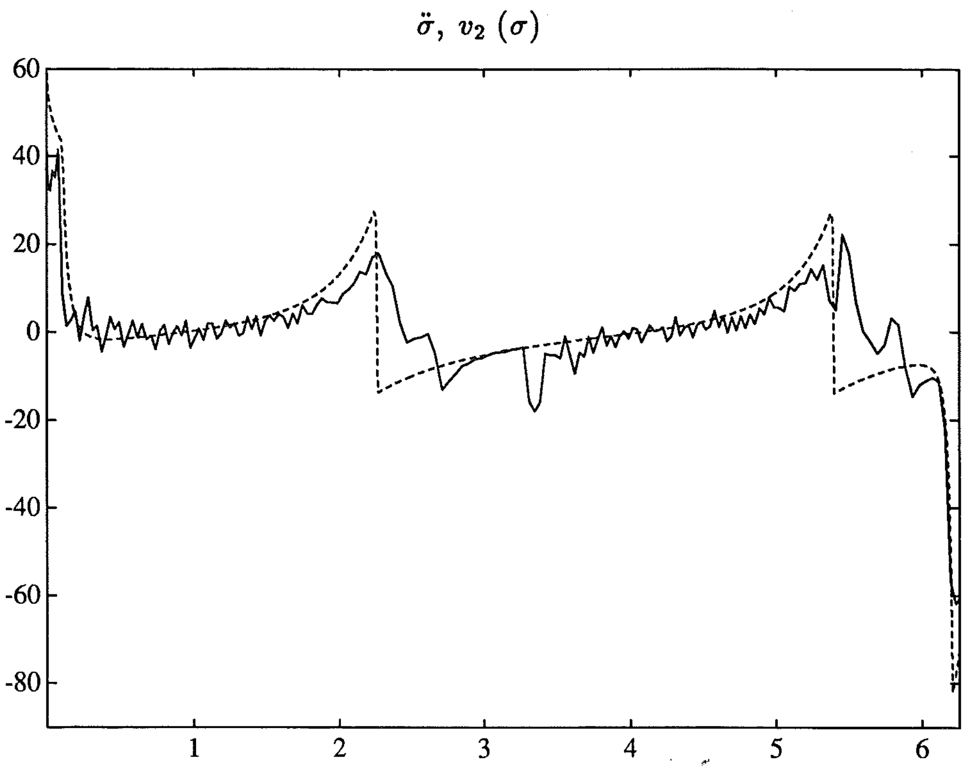


Figure 5.9 The nominal and actual path accelerations.

Experiment 2 – y-z-square

The path used in this experiment is specified in cartesian space. The path is a square with rounded corners, aligned with the y and z-axis of the cartesian coordinate system. The cartesian coordinate system is shown in Figure 5.2. The path specification is done as follows. B-splines [de Boor, 1978] are used to represent the cartesian path. Inverse kinematics is then used for transformation of the cartesian positions to joint positions. The transformed joint values are approximated by a B-spline function in joint space. This gives the functions $f(\sigma)$, $f'(\sigma)$, and $f''(\sigma)$. Figure 5.10 shows the path in joint space, and the corresponding cartesian path. The path parameter s is for this path defined in the interval $0 \leq s \leq s_f$ where $s_f = 56.3$. The starting point $s = 0$ and the points $s = 0.25s_f$, $s = 0.5s_f$, $s = 0.75s_f$ are marked in the upper right plot, and in the lower right plot. As can be seen in the upper right plot, the path for joints 1 and 3 is almost aligned with the coordinate axes. The path is traversed counterclockwise in the q_1-q_3-plane, upper right plot, and clockwise in the y-z-plane, lower right plot.

The minimum time solution is shown in Figure 5.11. The upper plot shows the maximum velocity curve, dashed line, and the minimum time ve-

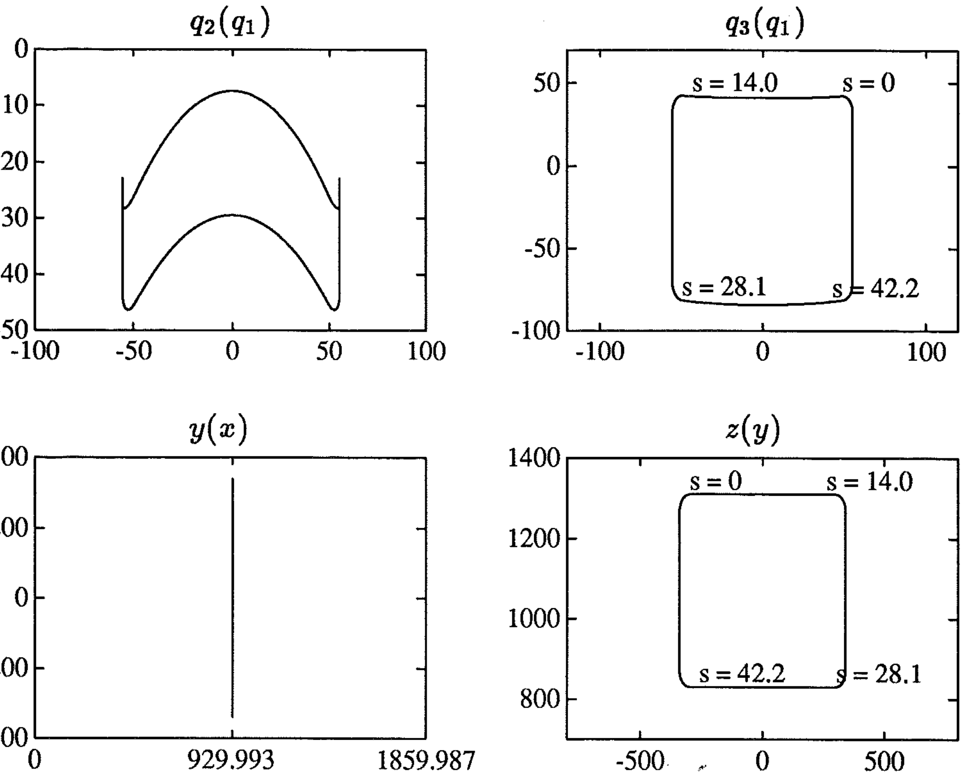


Figure 5.10 The path in joint space for Experiment 2, upper plots, and the corresponding cartesian path, lower plots.

locity profile, solid line. The dotted line is nonzero when $\ddot{s}$ is maximum and zero when $\ddot{s}$ is minimum. The lower plot shows the torques τ_1, solid line, τ_2, dashed line, and τ_3, dotted line, as functions of s. For $0 \leq s \leq 15$, which corresponds to the first straight line segment, shown in the upper right plot in Figure 5.10, τ_1 is at the limit, except for a small interval around $s \approx 12$, where τ_2 is at the limit. For $17 \leq s \leq 27$, which corresponds to the next straight line segment, τ_3 is at the limit. This behavior repeats itself for the third and fourth line segments, i.e. for the third straight line segment, τ_1 is at the limit, and for the final straight line segment τ_3 is at the limit.

The result of using the nominal velocity profile without the torque limits is shown in Figure 5.12. The upper plot shows the path parameter $s(t)$ and the path velocity $\dot{s}(t)$. The mid left plot shows the desired and actual paths in joint space for joints 1 and 2, i.e. q_{r_2} as a function of q_{r_1}, and q_2 as a function of q_1. The mid right plot is the corresponding plot for the desired and actual paths for joints 3 and 1. The lower left plot shows the reference trajectories and the actual trajectories, i.e. $q_{r_i}(t)$ and $q_i(t)$ where $i = 1, 2, 3$. At $t = 0$, the trajectories are, in order of decreasing values, q_1, q_3, and q_2.

The lower right plot shows the torques $\tau_1(t)$, solid line, $\tau_2(t)$, dashed line,

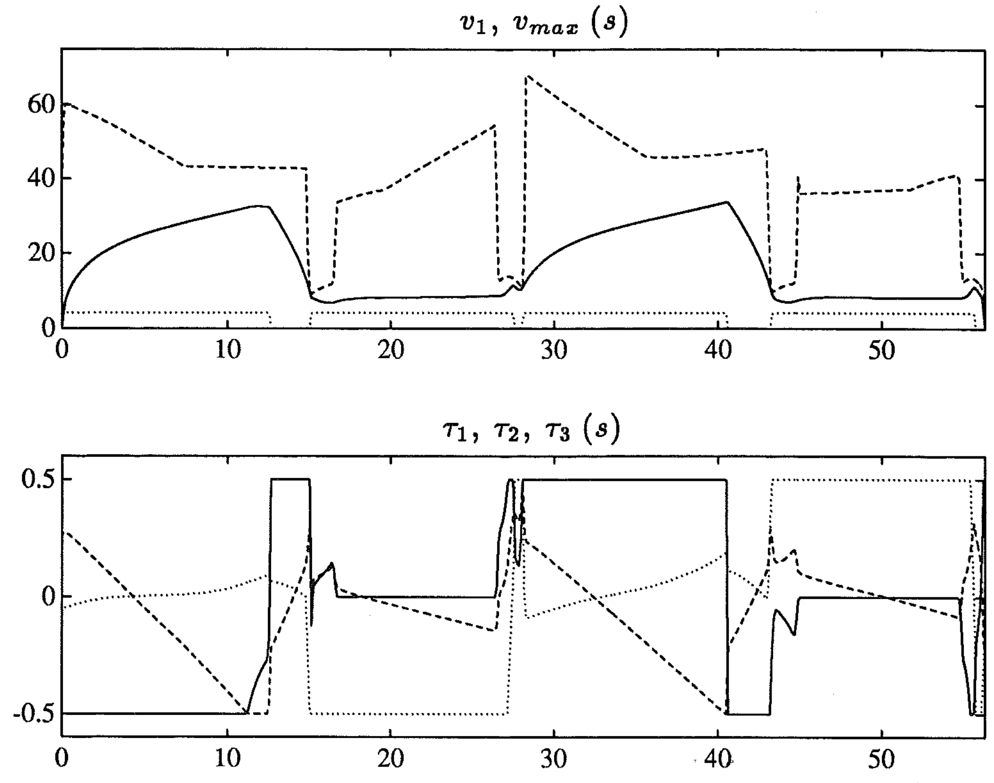


Figure 5.11 The minimum time solution for Experiment 2, and the corresponding torques.

and $\tau_3(t)$, dotted line. As can be seen from this plot, the torques are outside the limits ± 0.5. The corresponding motion in cartesian space is shown in Figure 5.13. As can be seen in the lower plot, the cartesian path deviation in the x-direction is approximately 6mm.

The result of using the nominal velocity profile and the torque limits is shown in Figure 5.14. The corresponding plot in cartesian space is shown in Figure 5.15. As can be seen from the plots, the path cannot be followed. In cartesian space, the path deviation in the y-z-plane is concentrated to the corners. The path deviation in the x-y-plane has also increased, compare the lower plots in in Figures 5.13 and 5.15. In joint space in the q_1-q_3-plane, the path deviation is also concentrated to the corners, as can be seen in Figure 5.14, the mid right plot. The explanation for this is as follows. From Figure 5.10, the upper right plot, it is seen that the path in the q_1-q_3-plane starts in the upper right corner. During the first line segment, joint one is limiting the motion, and the nominal reference trajectory cannot be followed, see the lower left plot in Figure 5.14. This means that when joint 3 starts to move, joint 1 has not reached the corner, which results in deviation from the path. During the next line segment, joint 3 is limiting, but the tracking error is

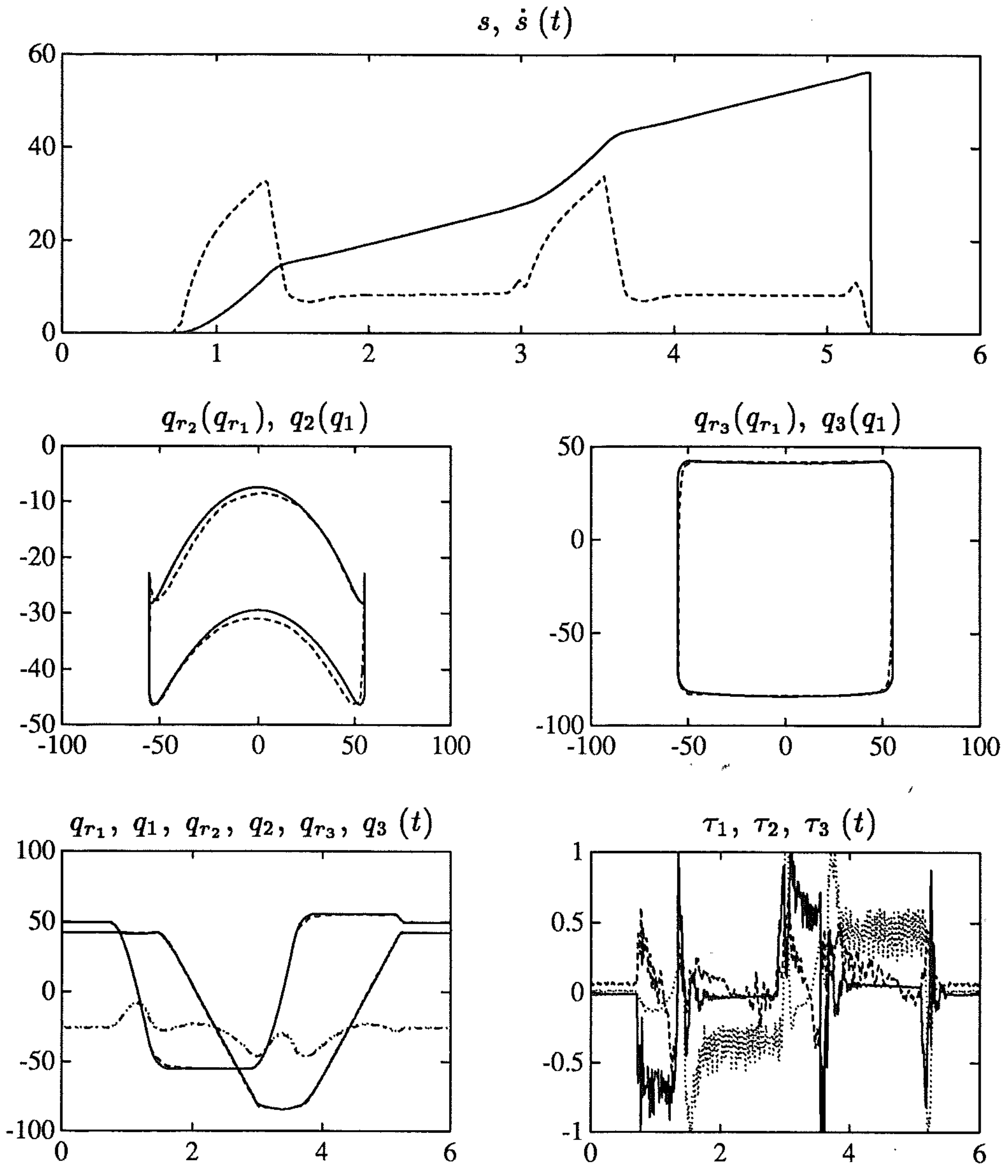


Figure 5.12 The result of using the nominal velocity profile without the torque limits. The traversal time was $t_f = 4.57$ and the mean square tracking error was $mse = 3.35$.

now smaller, see the lower left plot in Figure 5.14. This has the effect that the path following through the next corner, i.e. the lower left corner in the mid right plot in Figure 5.14, is good. During the third line segment, joint one is again limiting, which results in path deviation in the lower right corner in the mid right plot in Figure 5.14.

The result of using the path velocity controller (4.24) is shown in Figures 5.16 and 5.17. As can be seen from the plots, the path following is good.

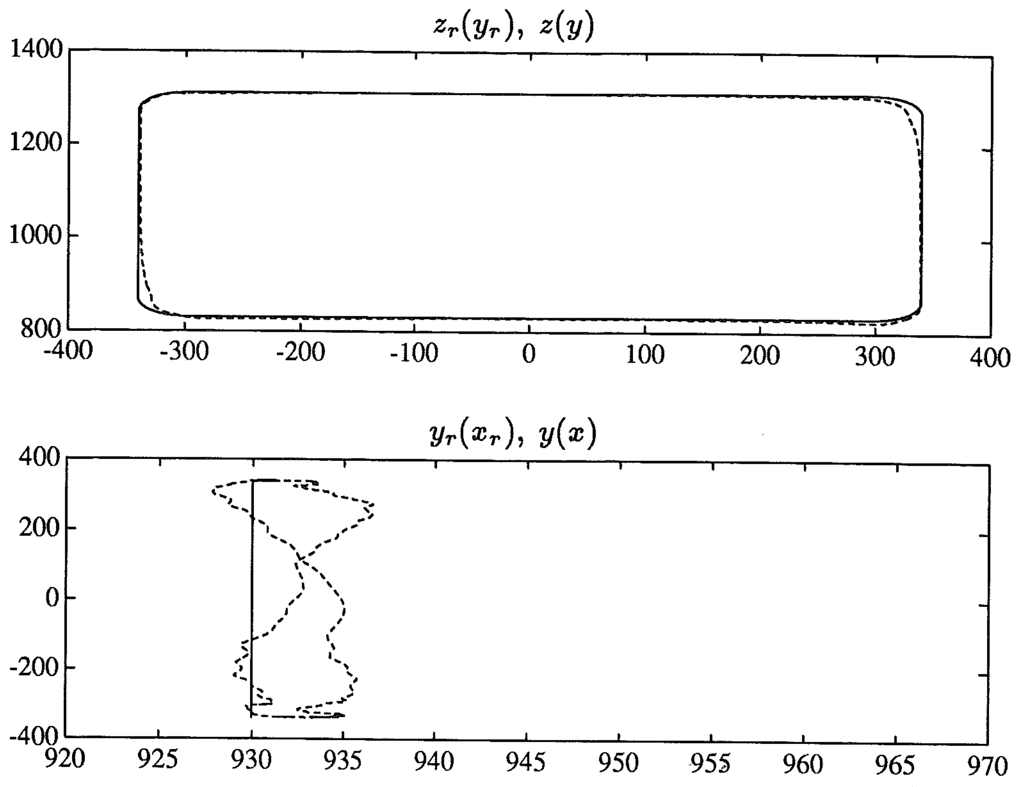


Figure 5.13 The nominal motion in cartesian space. The axes are scaled in millimeters.

The path deviation in the corners has been reduced. The motion through the corners has been synchronized, in the sense that during the first line segment, the velocity of the reference trajectory has been reduced so that when joint 3 starts to move, there is no tracking error in joint 1, and motion along the path through the corner is possible. The path deviation in the x-y-plane is also reduced, see the lower plot in Figure 5.17, which shows that the path deviation is now comparable to the case when the torque limits are removed, see the lower plot in Figure 5.13.

Figure 5.18 shows the actual cartesian paths from Figures 5.13 and 5.17. The solid line is the desired path, the dashed line is the path obtained when the path velocity controller is used, and the dotted line is the path from Figure 5.13. As can be seen in the figure, the path velocity controller gives the same qualitative behavior with respect to path deviation as for the case when the torque limits are removed. The same observation could be done in Figure 5.7. In Figure 5.18 it is also seen that the path deviation when the path velocity controller is used is actually smaller than for the artificial experiment. An explanation for this could be that the velocity is reduced compared to the experiment where the torque limits were removed.

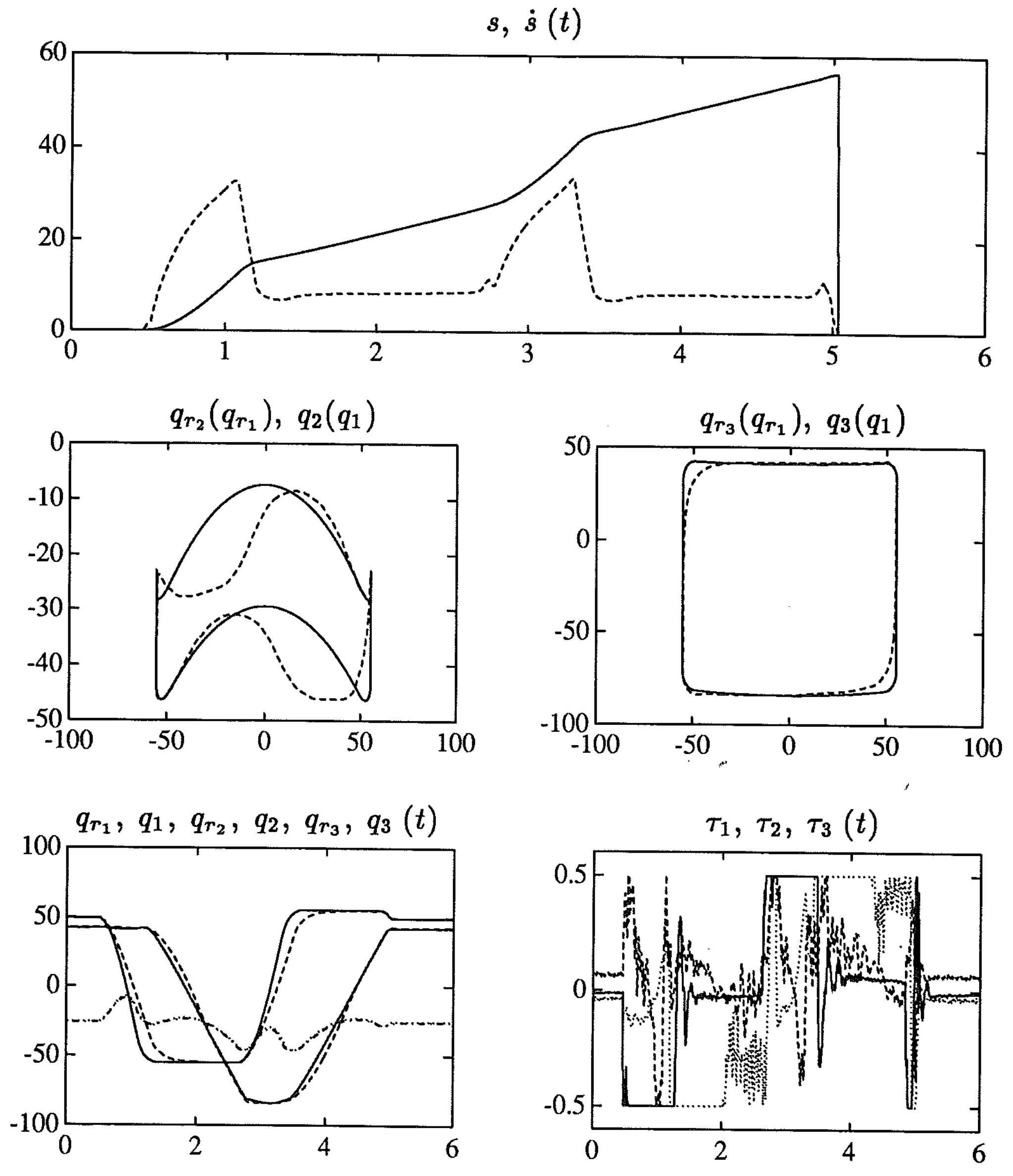


Figure 5.14 The result when the torques are limited. The torque utilization was $\tau_u = 0.92$, the traversal time was $t_f = 4.57$, and the mean square tracking error was $mse = 94.1$.

The evaluation plot for this experiment is shown in Figure 5.19. As can be seen from the lower plot, the limits on $\ddot{\sigma}$ are inadmissible for short intervals. This happens e.g. at the end of the deceleration phase for the first and the third line segments. As can be seen from the nominal minimum time velocity profile in Figure 5.11, the points where the optimal velocity profile touches the maximum velocity curve corresponds roughly to the points where the bounds on $\ddot{\sigma}$ are inadmissible, see the lower plot in Figure 5.19. For velocities above

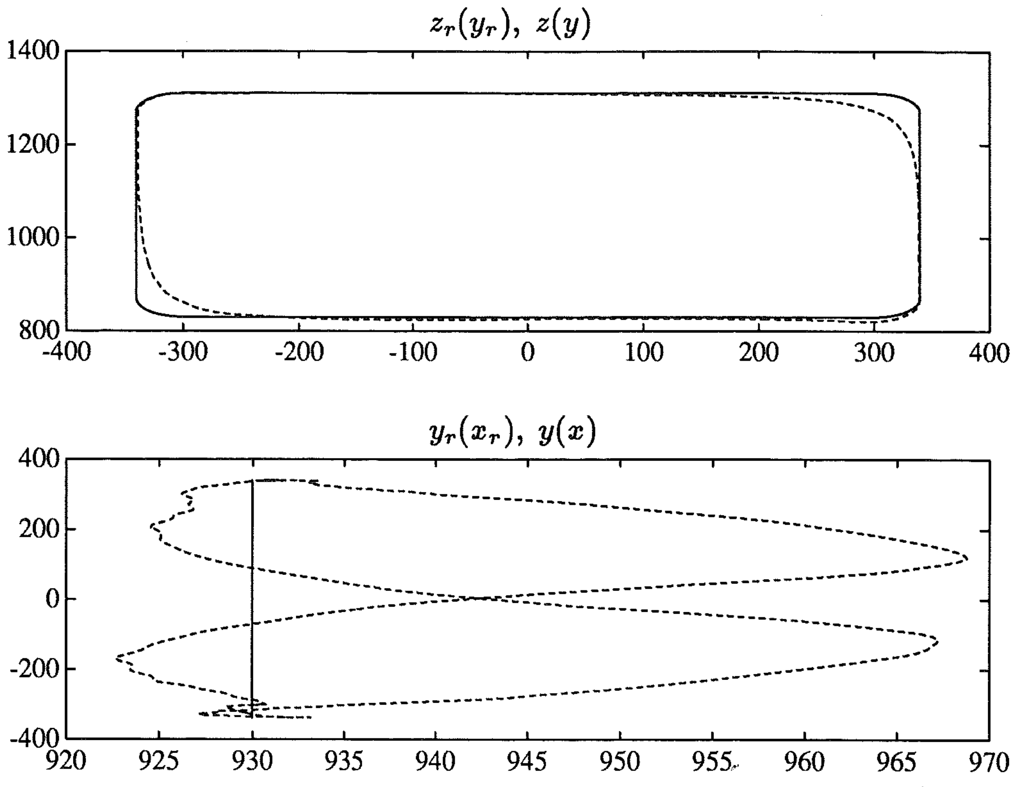


Figure 5.15 The torque limited motion in cartesian space.

the maximum velocity curve, with the exception of critical points, the limits on the nominal path acceleration (3.26) are inadmissible, i.e. $\ddot{s}_{min}(s, \dot{s}) > \ddot{s}_{max}(s, \dot{s})$. It can therefore be expected that close to the touching points, the limits on $\ddot{\sigma}$ also become inadmissible, i.e. $\ddot{\sigma}_{min}(\beta_1, \beta_2) > \ddot{\sigma}_{max}(\beta_1, \beta_2)$, which is the case for this experiment. Note however that in spite of the inadmissible bounds, the path following is comparable to the experiment where the torque limits were removed, as is seen in Figure 5.18. The numerical results for this experiment are given in Table 5.2. As can be seen in Table 5.2, both the *mse* and the *cmse* are of the same magnitude as for the nominal motion. For this experiment, the *mse* and the *cmse* are smaller than for the nominal motion. This agrees with the visual inspection of Figure 5.18, where the path deviation when the path velocity controller is used is smaller than for the experiment where the torque limits were removed.

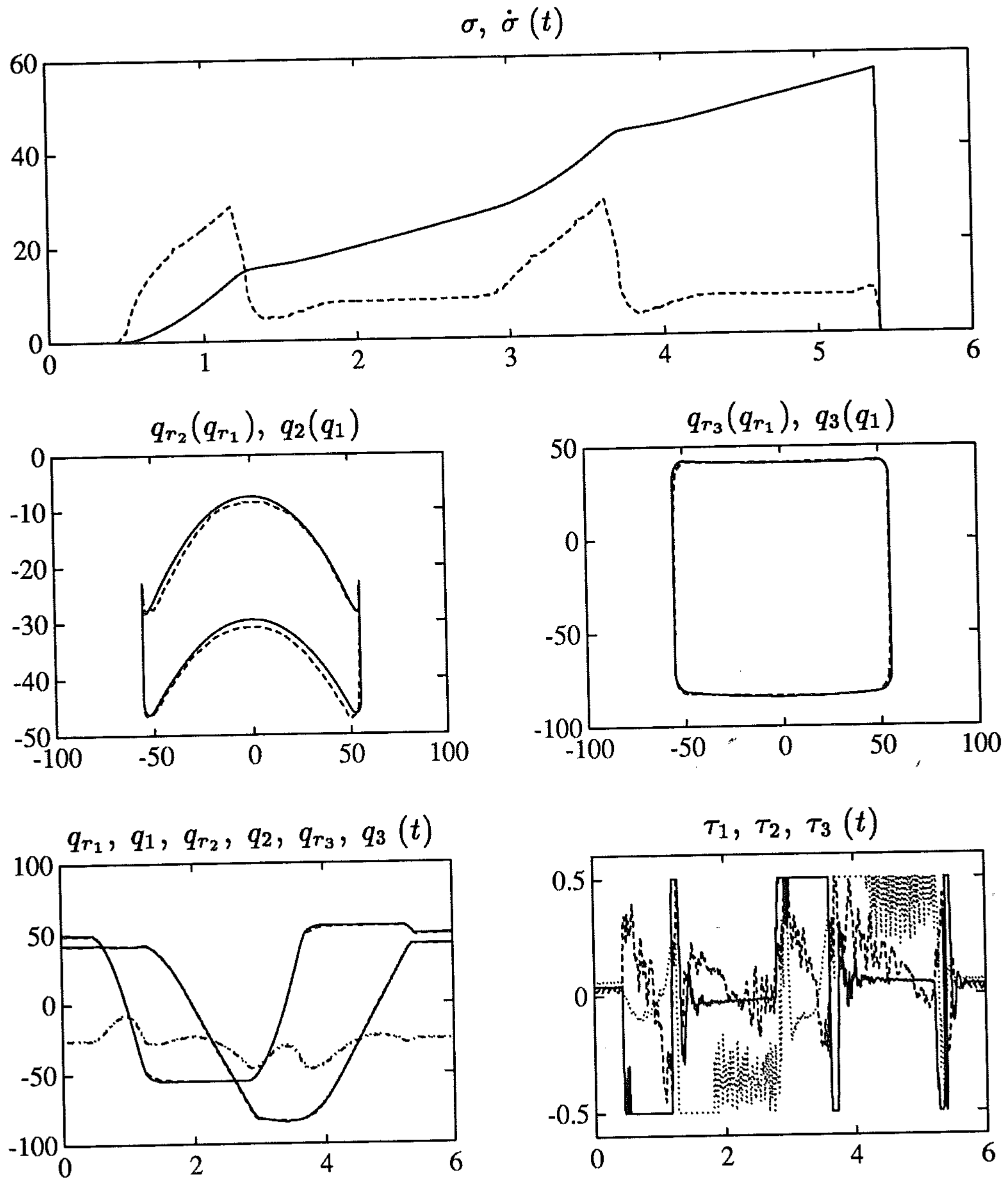


Figure 5.16 The result when the path velocity controller is used. The torque utilization was $\tau_u = 0.90$, the traversal time was $t_f = 4.95$, and the mean square tracking error was $mse = 3.16$.

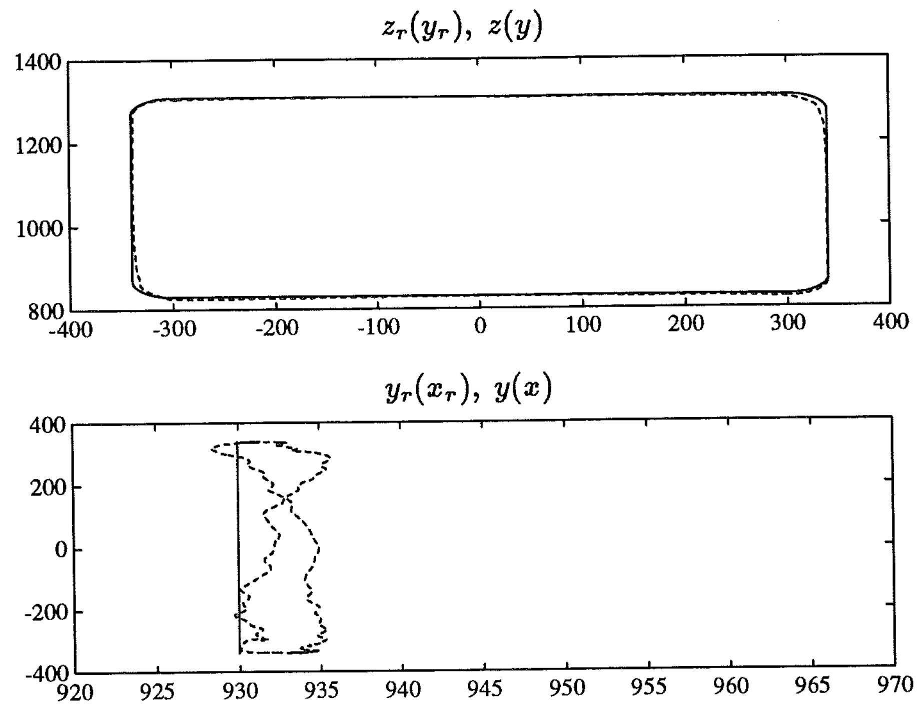


Figure 5.17 The motion in cartesian space when the path velocity controller is used.

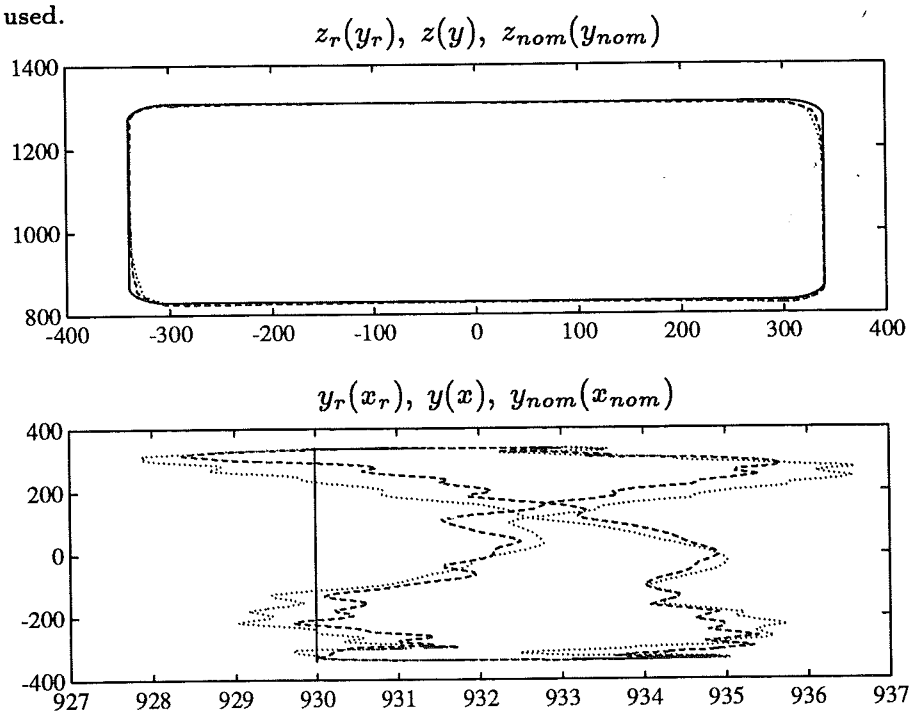


Figure 5.18 A comparison in cartesian space, showing the actual path from the experiment where the torque limits were removed, dotted line, and the actual path when the path velocity controller is used, dashed line. The solid line is the desired path.

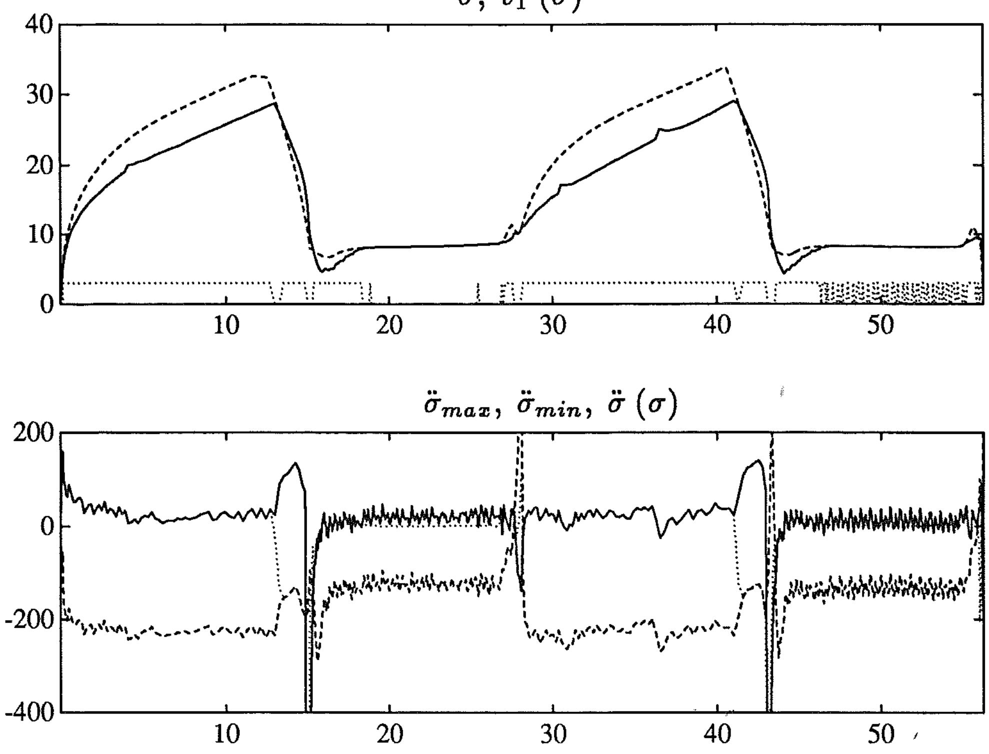


Figure 5.19 Evaluation of Experiment 2. The nominal and actual velocity profiles are shown in the upper plot and the path acceleration $\ddot{\sigma}$ and the limits on $\ddot{\sigma}$ are shown in the lower plot.

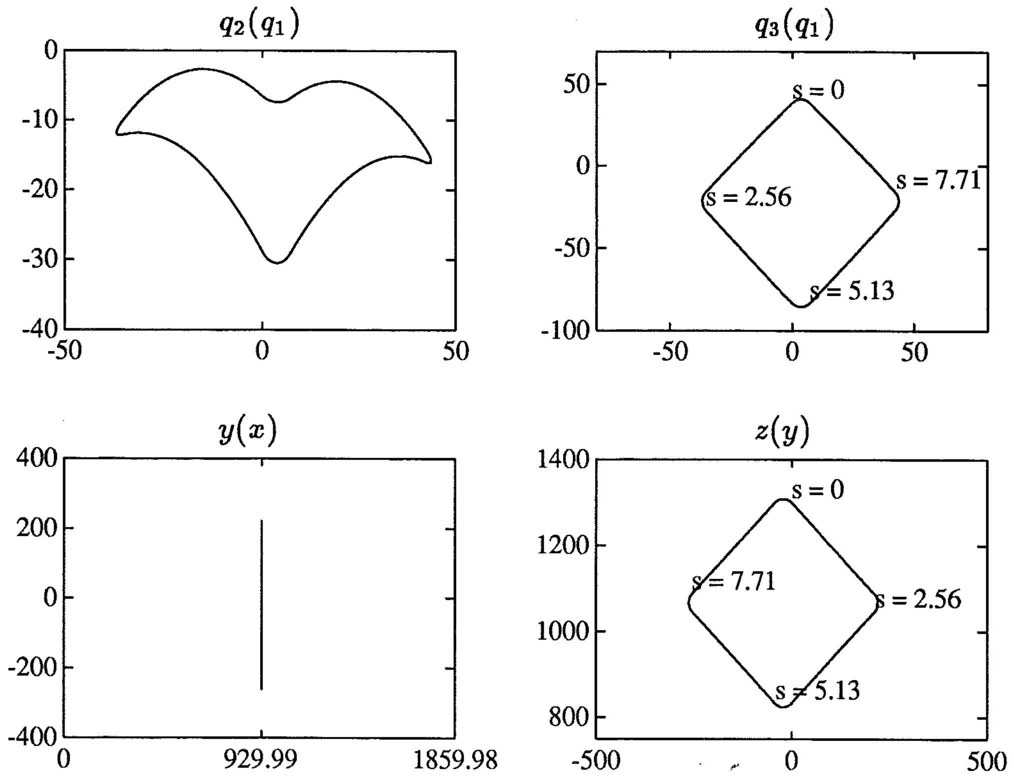


Figure 5.20 The path in joint space for Experiment 3, upper plots, and the corresponding cartesian path, lower plots.

Experiment 3 – Rotated y-z-square

The path used in this experiment is a rotated square with rounded corners. The path is obtained by 45 degrees rotation of a square which is aligned with the y and z-axis of the cartesian coordinate system. Figure 5.20 shows the path in joint space and the corresponding cartesian path. The path in the q_1-q_3-plane starts at the upper corner and is traversed counterclockwise, as can be seen in the upper right plot in Figure 5.20. In cartesian space, the path in the y-z-plane starts in the upper corner and proceeds clockwise.

The minimum time velocity profile and the corresponding torques are shown in Figure 5.21. As can be seen in the lower plot, the torque τ_3, dotted line, is at the limit, except for small intervals where τ_1, solid line, is at the limit. This happens in the left and right corners of the path in the q_1-q_3-plane, shown in the upper right plot in Figure 5.20. The result of using the nominal velocity profile is shown in Figure 5.22. As can be seen in the lower right plot, the torque τ_3, dotted line, is outside the limits ± 0.5. The lower left plot shows the reference trajectories and the actual trajectories. At $t = 0$, the trajectories are, in order of decreasing values, q_3, q_1, and q_2. The

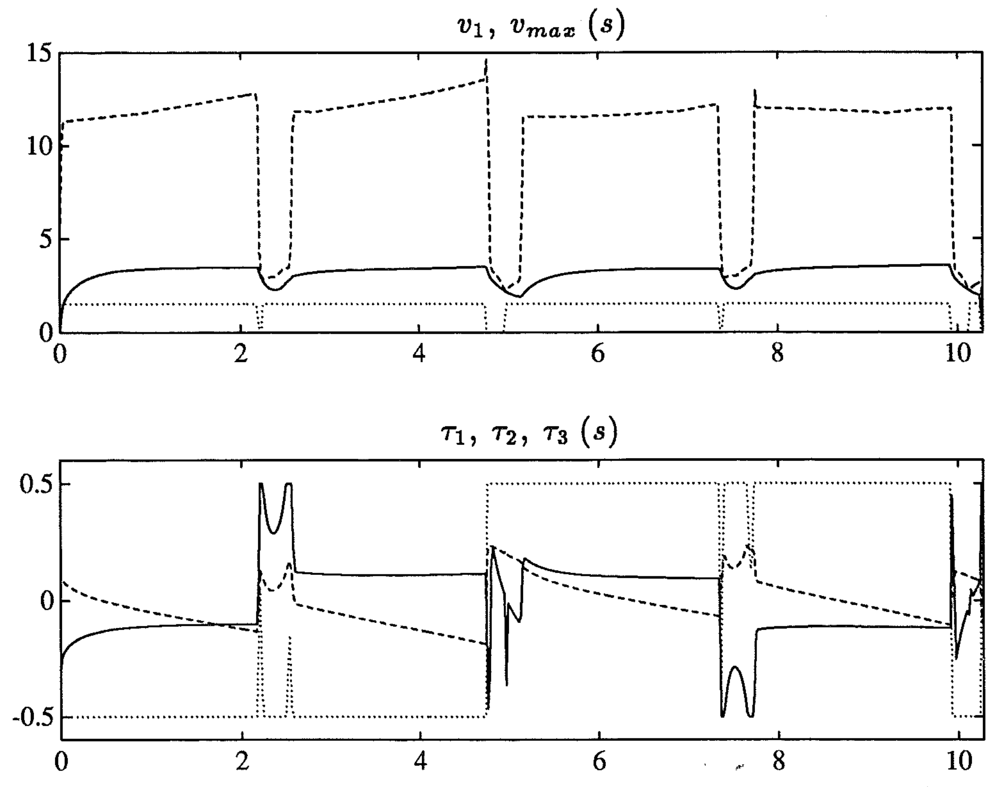


Figure 5.21 The minimum time solution for Experiment 3, and the corresponding torques.

corresponding motion in cartesian space is shown in Figure 5.23. As can be seen in the lower plot, the path deviation in the x-direction is approximately 5 mm.

The result of using the nominal velocity profile and the torque limits is shown in Figure 5.24. The corresponding motion in cartesian space is shown in Figure 5.25. As can be seen from the plots, the desired path cannot be followed. For this path, the path deviation occurs along the entire path, as opposed to the previous experiment, where the path deviation was concentrated to the corners. The reason is that the path in the q_1-q_3-plane is not aligned with the coordinate axes, as was the case in the previous experiment, compare the upper right plots in Figures 5.10 and 5.20. This means that tracking errors in one joint, in this case joint 3, see the lower left plot in Figure 5.24, result in path deviations since the other joints move simultaneously. This was not the case in the previous experiment where the tracking error in joint 1 did not result in path deviations during the straight line segments. The path deviation then occurred at the corner when joint 3 started to move, see the lower left plot in Figure 5.14.

Using the path velocity controller (4.24) gives the result shown in Figure

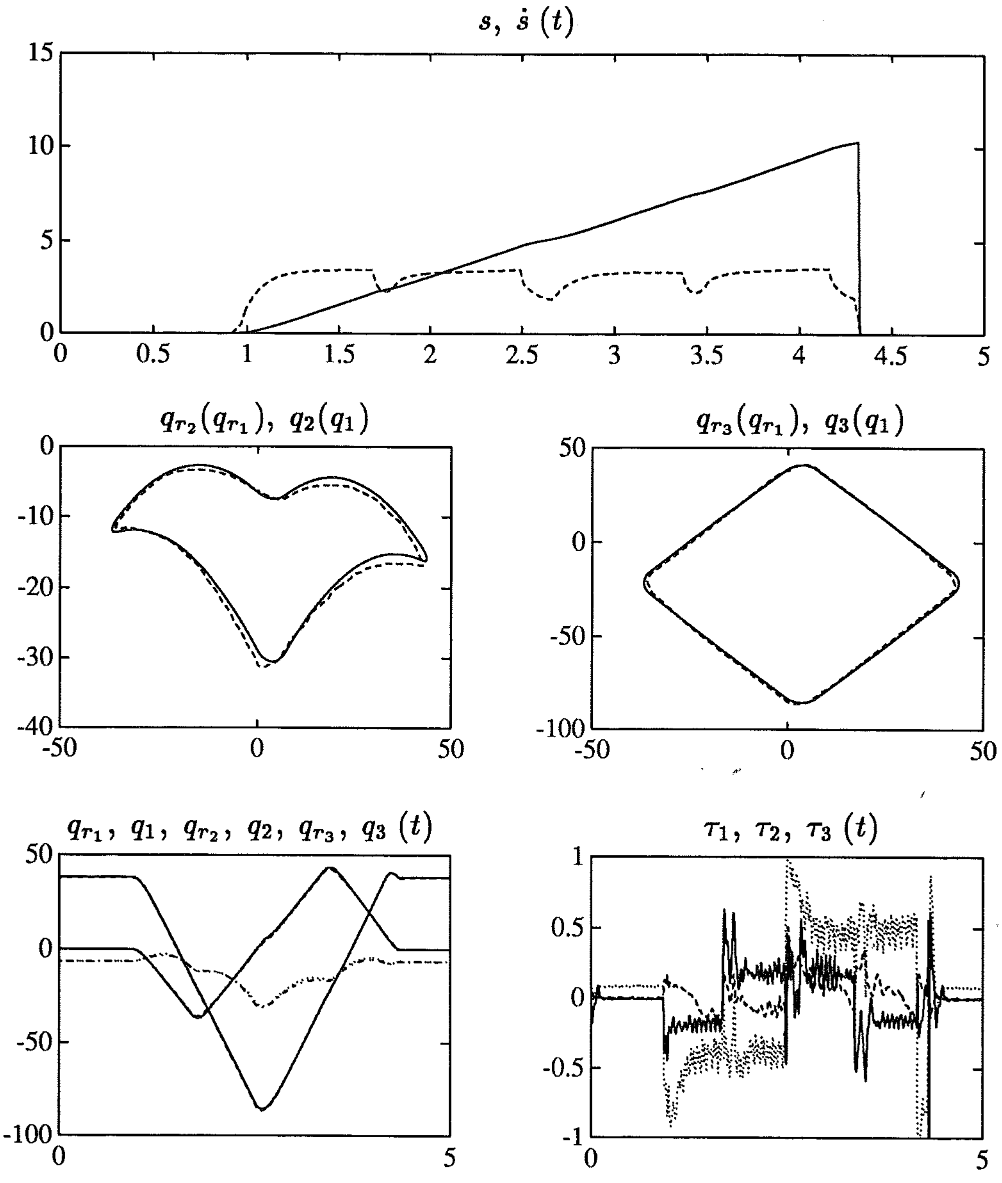


Figure 5.22 The result of using the nominal velocity profile without the torque limits. The traversal time was $t_f = 3.4$ and the mean square tracking error was $mse = 3.16$.

5.26. The corresponding motion in cartesian space is shown in Figure 5.27. As can be seen from the plots, the path following is comparable to the experiment where the torque limits were removed, shown in Figures 5.22 and 5.23. The path deviation in the x-direction is however larger when the path velocity controller is used, as can be seen by comparing the lower plots in Figures 5.23 and 5.27. This is also seen in the lower plot in Figure 5.28, which shows the path from the experiment where the torque limits were removed together

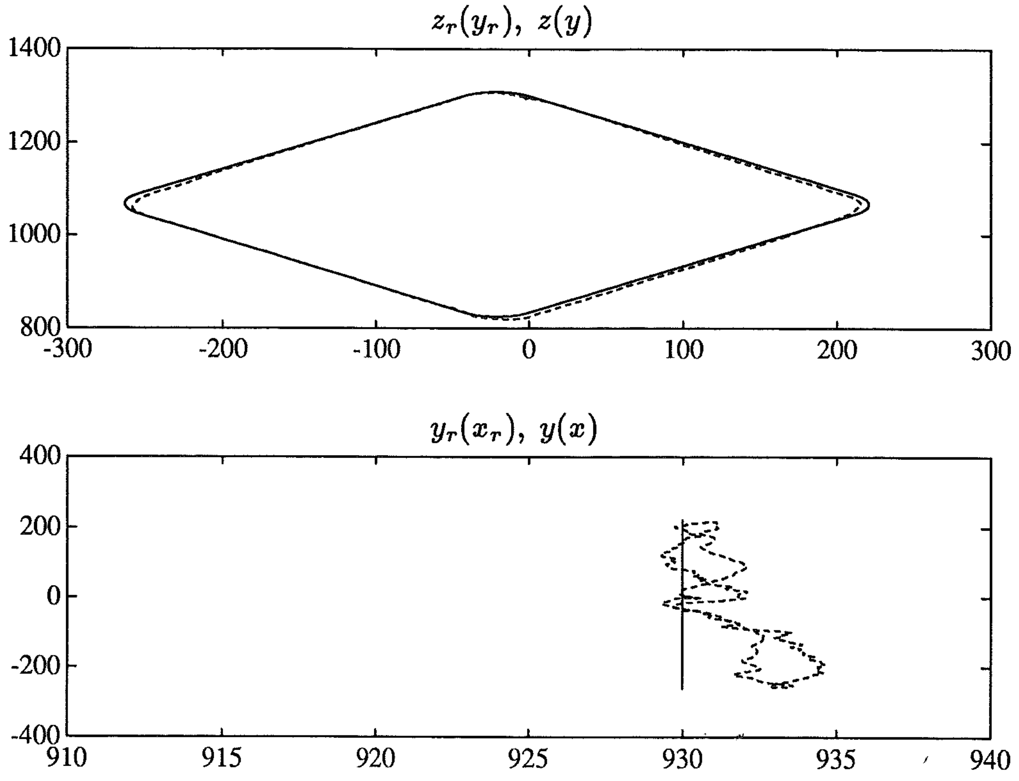


Figure 5.23 The nominal motion in cartesian space.

with the path obtained when the path velocity controller is used.

An evaluation plot is shown in Figure 5.29. As can be seen the lower plot, the limits on $\ddot{\sigma}$ are oscillative. This has the effect that the limits are used only occasionally, as can be seen in the dotted line in the upper plot. The oscillations in the limits on $\ddot{\sigma}$ are also seen as oscillations in the limiting torque τ_3, the dotted line in the lower right plot in Figure 5.22. It can also be seen in the lower plot in Figure 5.29 that the limits on $\ddot{\sigma}$ are inadmissible around $\sigma \approx 5$. This is probably the explanation for the increased error in the x-direction, which can be seen in the lower plot in Figure 5.28. This error can also be seen as a tracking error in joint 3 in the lower left plot in Figure 5.26, for $t \approx 2.5$, where $q_3 \approx -85$. A comparison with the corresponding plot in Figure 5.22 shows that the tracking error is smaller for the experiment where the torque limits were removed. The inadmissibility of the limits on $\ddot{\sigma}$ can thus be related to the path deviation, and is hence an indication that the nominal velocity is too high around $\sigma \approx 2.5$. One should however note that the limits were inadmissible also in the previous experiment, where the path following when the path velocity controller was used was comparable to the experiment where the torque limits were removed, as can be seen in Figure

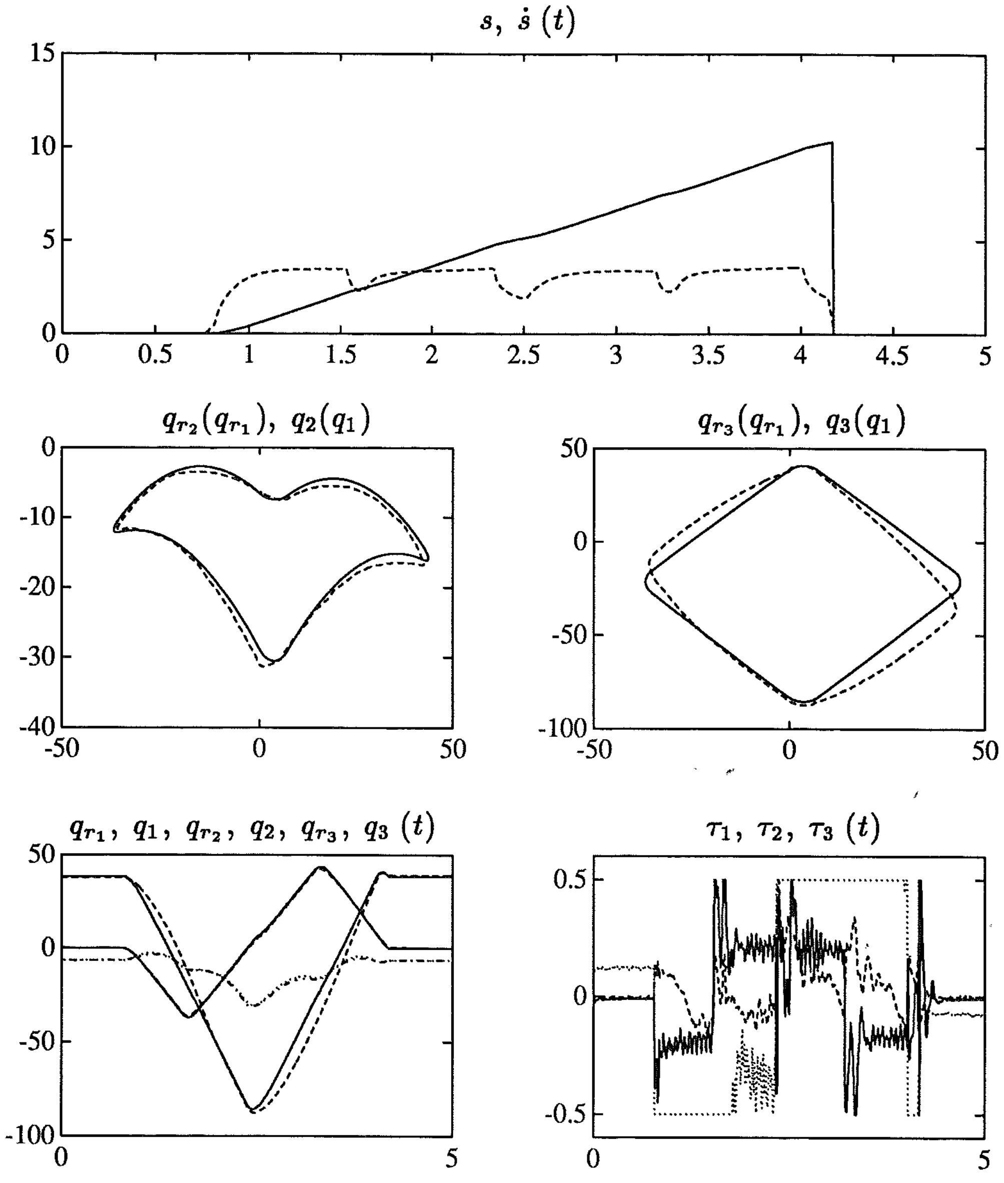


Figure 5.24 The result when the torques are limited. The torque utilization was $\tau_u = 0.95$, the traversal time was $t_f = 3.40$, and the mean square tracking error was $mse = 40.8$.

5.18.

A possible explanation for the oscillations in $\ddot{\sigma}_{min}$ and $\ddot{\sigma}_{max}$ in the lower plot in Figure 5.29 could be joint flexibility. The path acceleration $\ddot{\sigma}$ and the limits on $\ddot{\sigma}$ obtained from the simulation in Figure 4.18 are shown in Figure 5.30. As can be seen in the figure, the limits on $\ddot{\sigma}$ are oscillative, and the limits are used only occasionally, a behavior which is also seen in the lower plot in Figure 5.29.

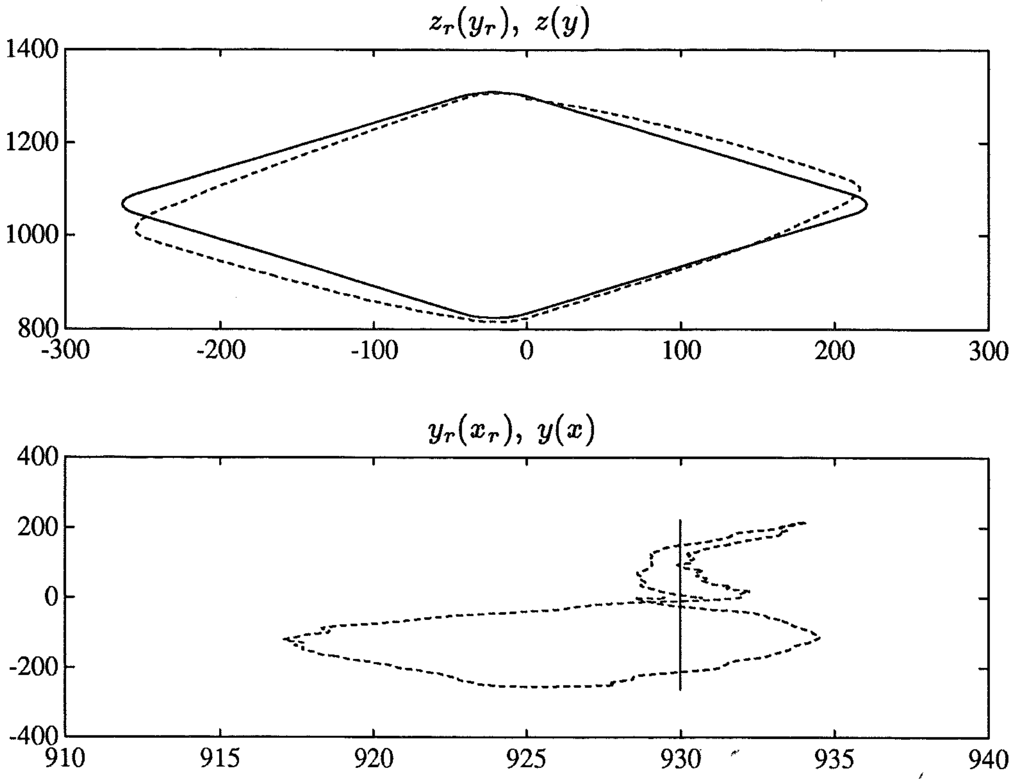


Figure 5.25 The torque limited motion in cartesian space.

The numerical results for this experiment are shown in Table 5.3. The errors *mse* and *cmse* are slightly larger for the case when the path velocity controller is used, right column, compared to the experiment where the torque limits were removed, shown in the left column. This agrees with the increased path deviation when the path velocity controller is used, as can be seen in the lower plot in Figure 5.28.

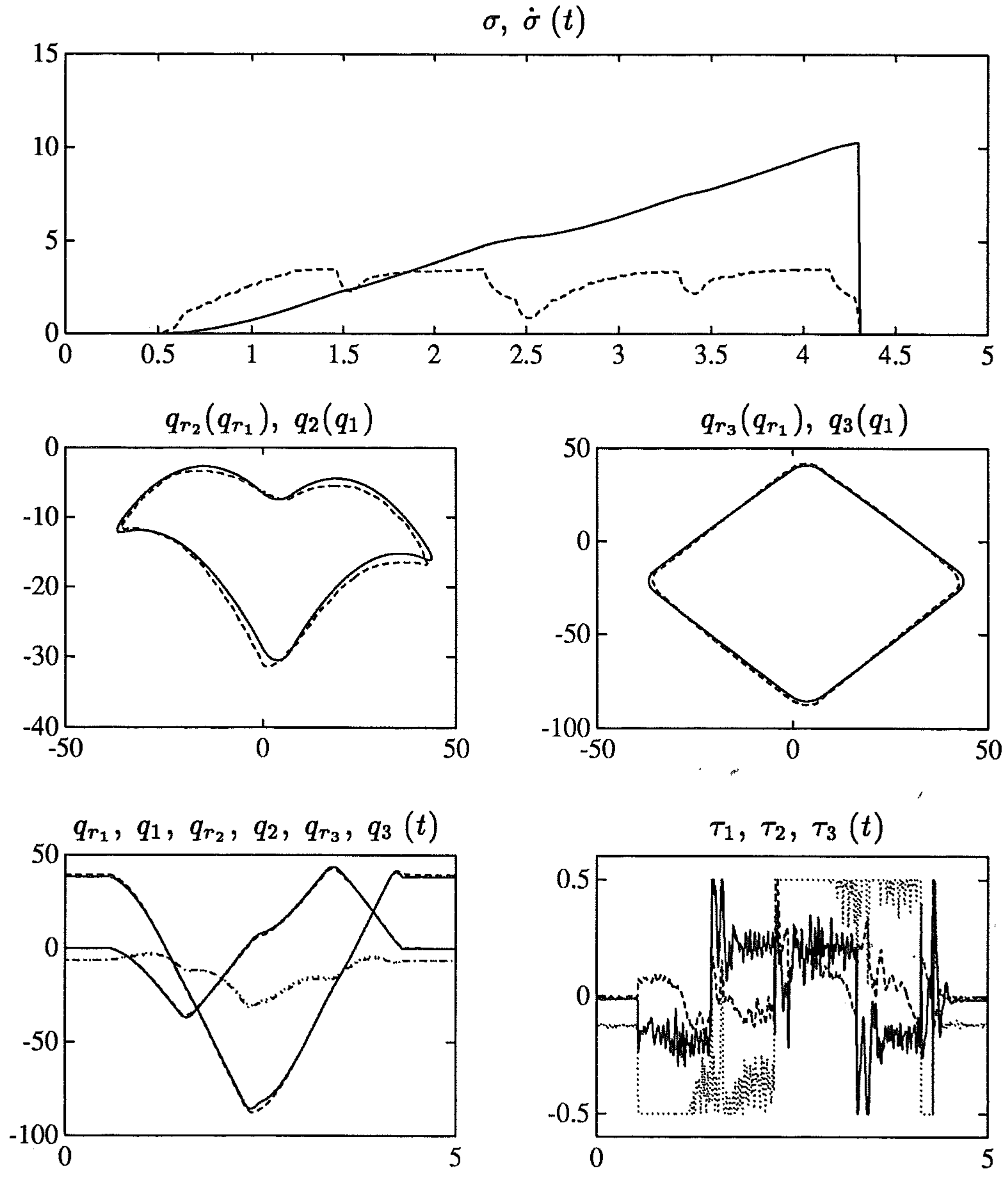


Figure 5.26 The result when the path velocity controller is used. The torque utilization was $\tau_u = 0.93$, the traversal time was $t_f = 3.78$, and the mean square tracking error was $mse = 3.36$.

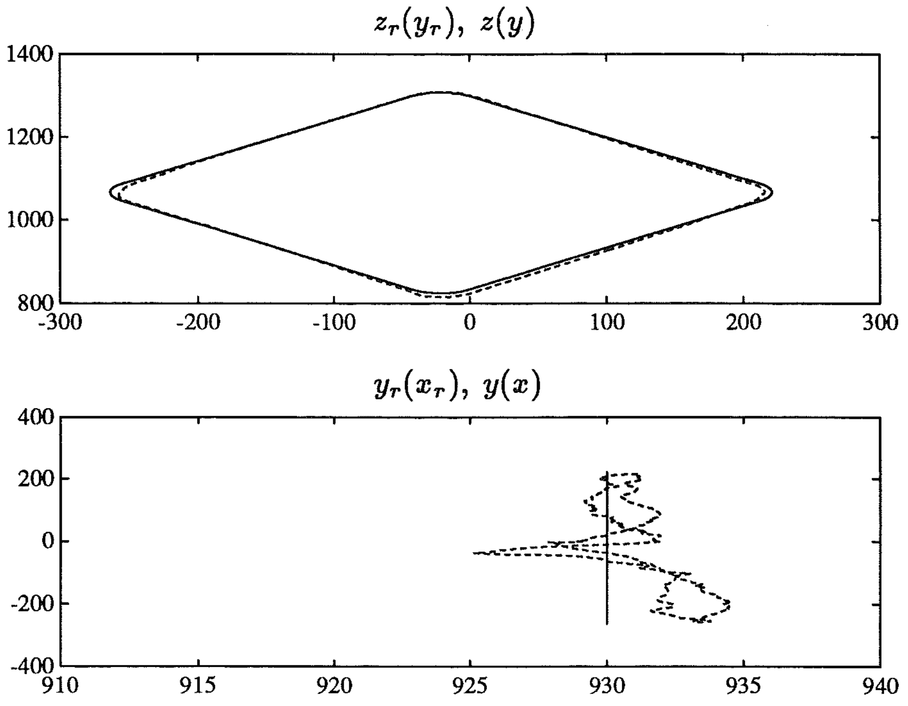


Figure 5.27 The result in cartesian space when the path velocity controller is used.

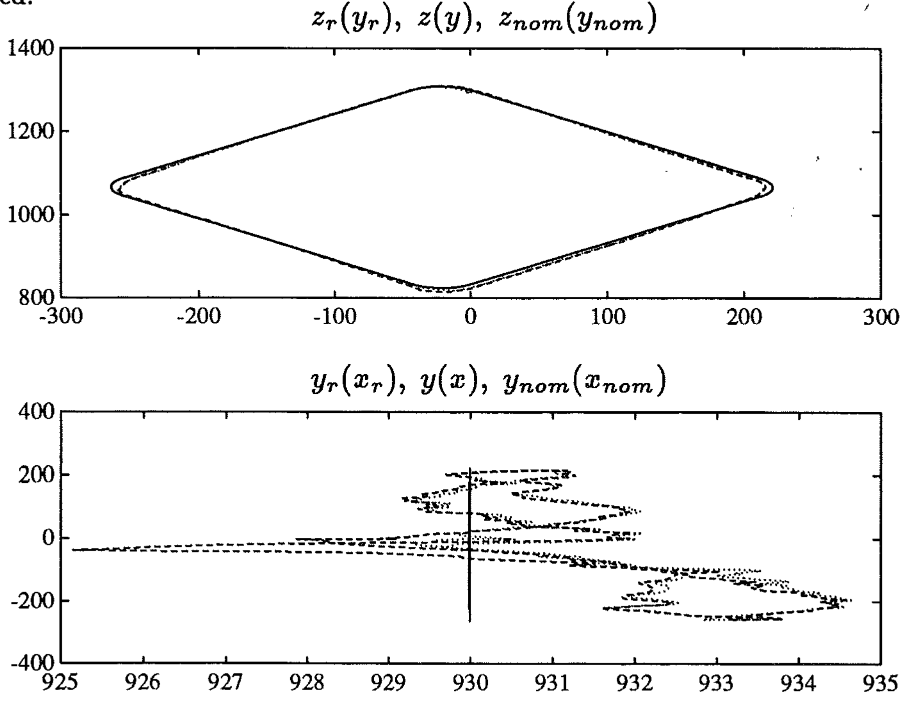


Figure 5.28 A comparison in cartesian space, showing the actual path from the experiment where the torque limits were removed, dotted line, and the actual path when the path velocity controller is used, dashed line. The solid line is the desired path.

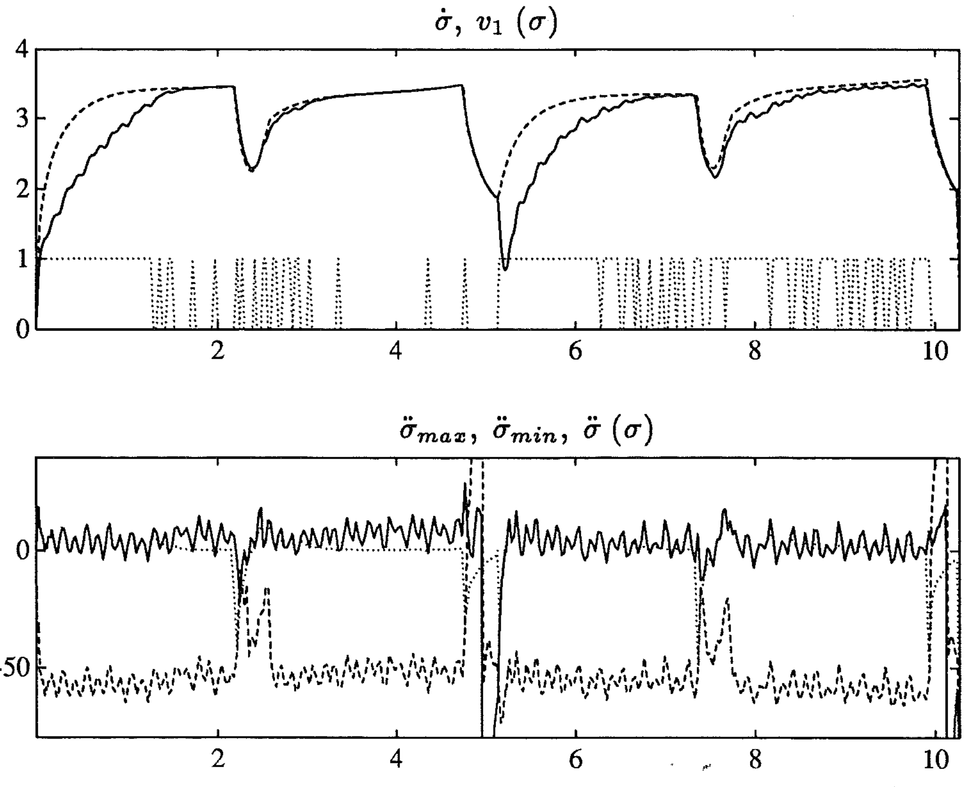


Figure 5.29 Evaluation of Experiment 3. The nominal and actual velocity profiles are shown in the upper plot, and the path acceleration $\ddot{\sigma}$ and the limits on $\ddot{\sigma}$ are shown in the lower plot.

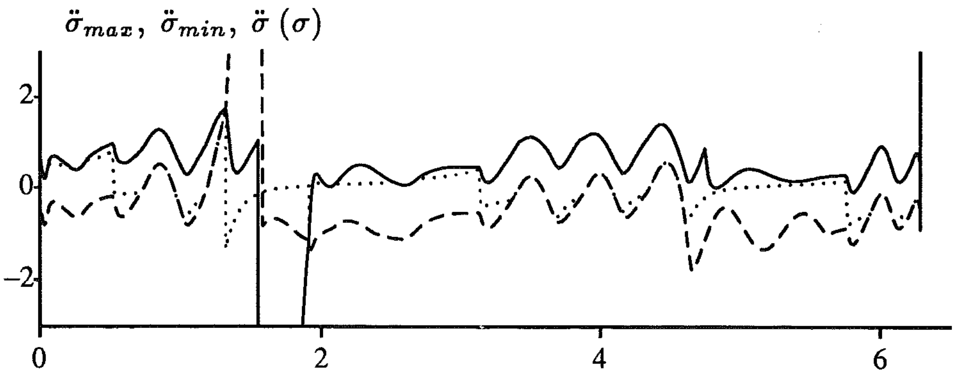


Figure 5.30 The path acceleration $\ddot{\sigma}$ and the limits on $\ddot{\sigma}$ from a simulation where the path velocity controller is used on a robot with joint flexibility.

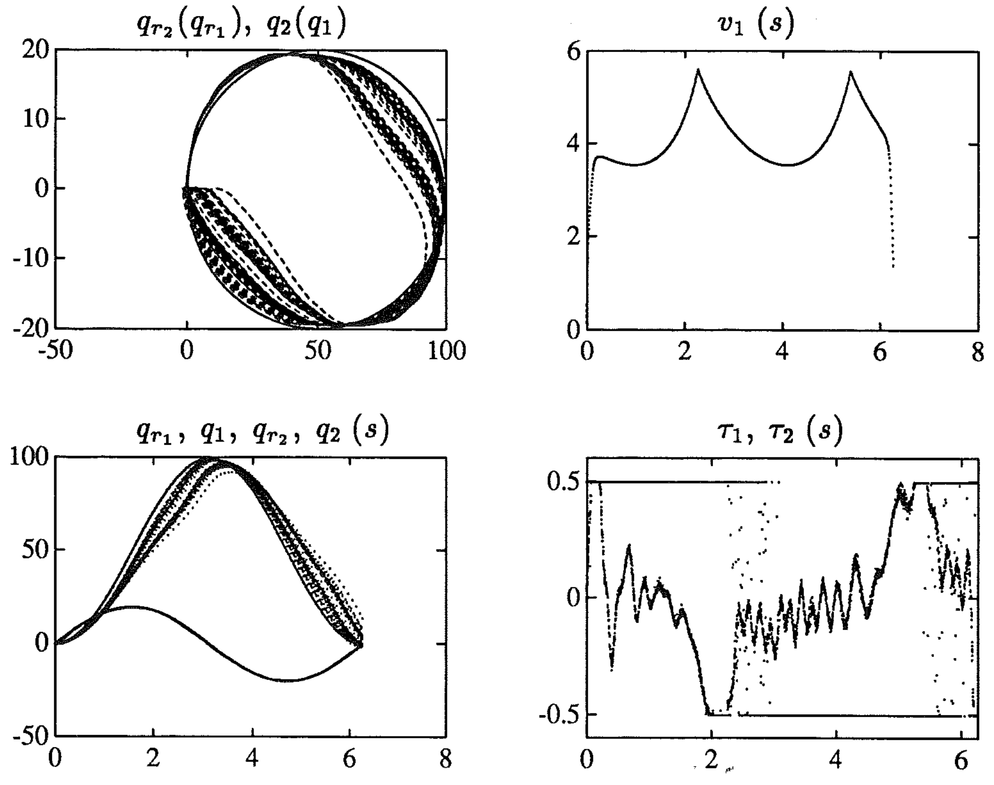


Figure 5.31 Time variations when the torques are limited. The upper left plot shows the desired path, and the actual paths obtained from repeated motion, while keeping the nominal velocity profile, shown in the upper right plot. The lower right plot shows the torques as functions of the path parameter.

Experiment 4 – Adaptation to time varying dynamics

This experiment demonstrates how the path velocity controller is able to adapt to time varying dynamics. The path and the nominal minimum time velocity profile are shown in Figure 5.3. The experiment was done by repeated motion along the path. The motion started after having the robot kept at the initial position for approximately 5–10 minutes. The result of repeated motion along the path when the nominal velocity profile and the torque limits ± 0.5 are used is shown in Figure 5.31. As can be seen from the upper left plot the path deviation is varying. The time variability is also seen in the lower right plot which shows the torques τ_1 and τ_2 as functions of s. From this plot it is seen that the switches in τ_1 occur irregularly. The experiment was repeated, but with the path velocity controller (4.24). The result is shown in Figure 5.32. As can be seen from the upper left plot, the path deviation is kept small. The path velocity is however varying, as can be seen from the upper right plot. Compared to Figure 5.31, the time variations now appear

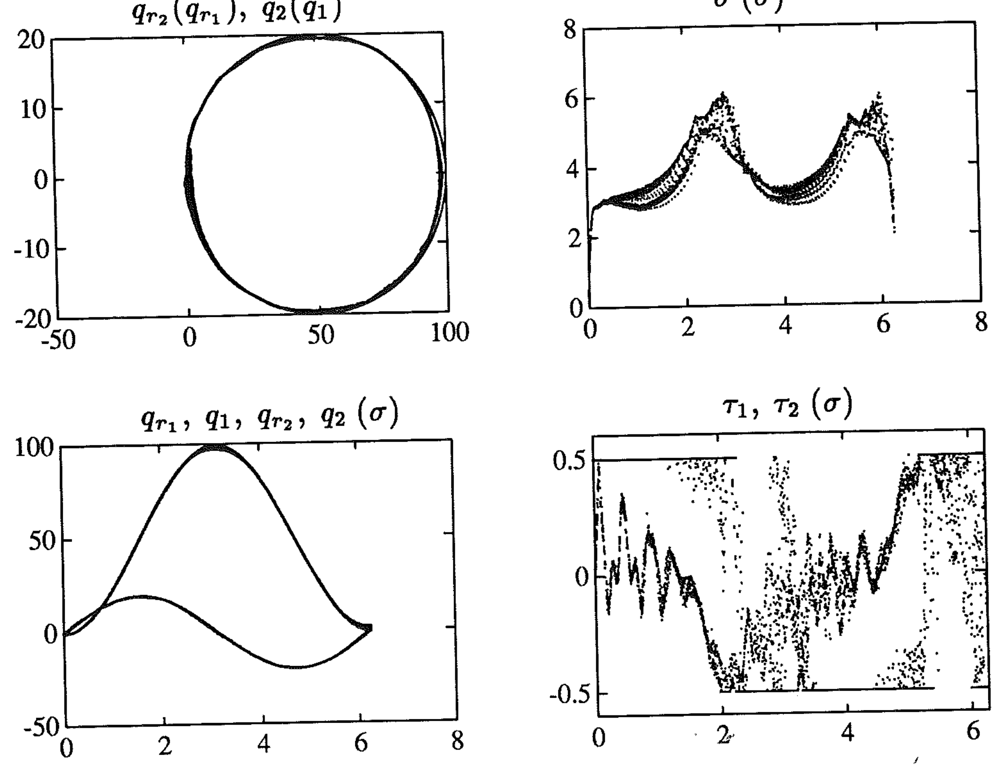


Figure 5.32 Time variations when the path velocity controller is used. The upper left plot shows that the path deviation is low, while the path velocity is varying, as is seen in the upper right plot.

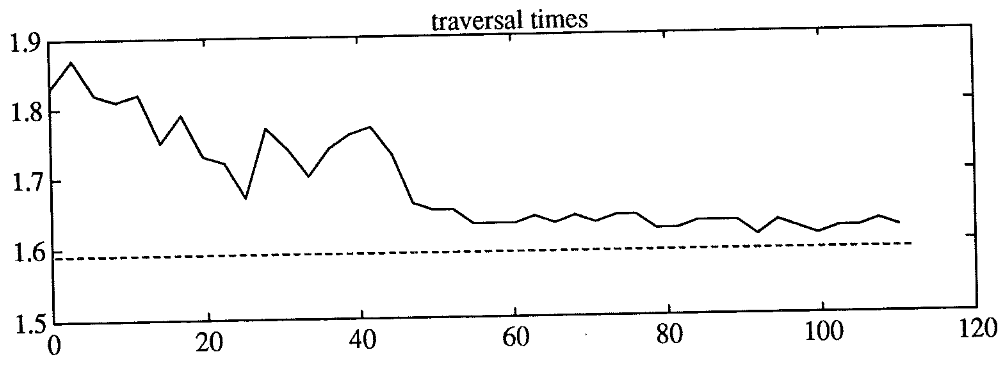


Figure 5.33 Traversal time for individual runs as a function of time.

as velocity variations instead of variations in the path deviation. Compare also the torques for the two cases. In Figure 5.31, the irregularity occurs mostly in τ_1. When the path velocity controller is used, the modification of the reference trajectory affects both torques. This is seen as an irregularity also in τ_2, which can be seen in the lower right plot in Figure 5.32.

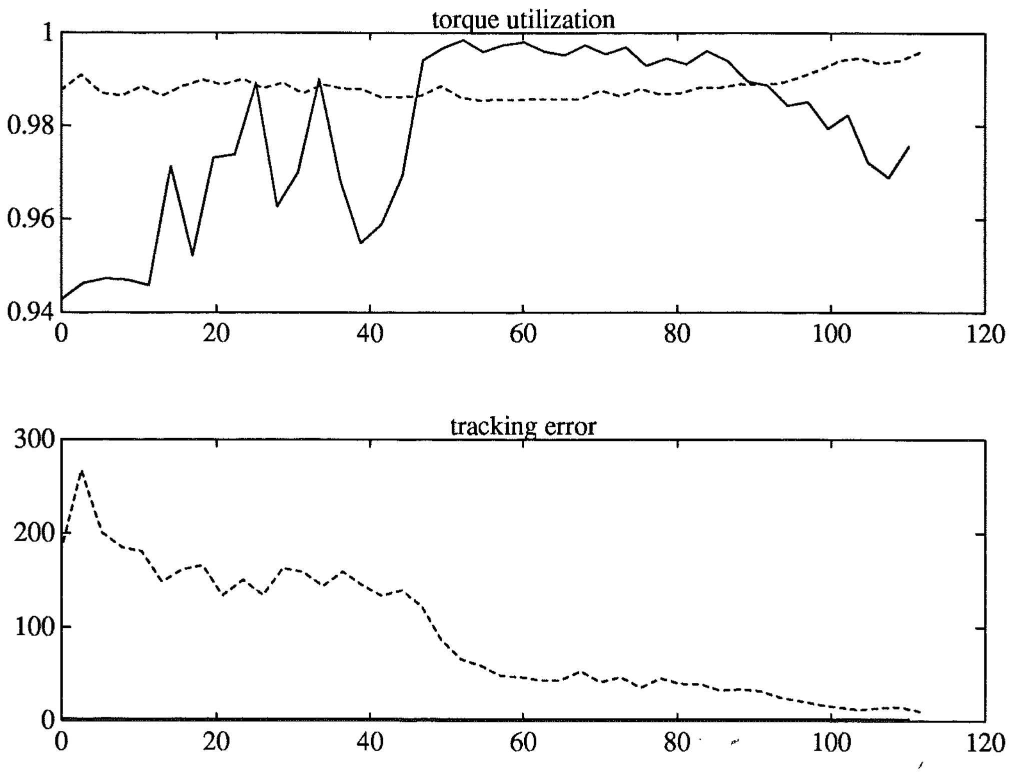


Figure 5.34 Torque utilization and mean square tracking error as functions of time.

Figure 5.33 shows the traversal time for the individual runs as a function of time. As can be seen from the Figure, the result of using the path velocity controller is decreasing path traversal time. Figure 5.34 shows the torque utilization and the *mse* for the torque limited motion, dashed lines, and for the case when the path velocity controller is used, solid lines. The tracking error for the torque limited case is large initially, and decreases slowly. When the path velocity controller is used, the tracking error is kept at a constant, and satisfactory level. The torque utilization is approximately the same for both experiments.

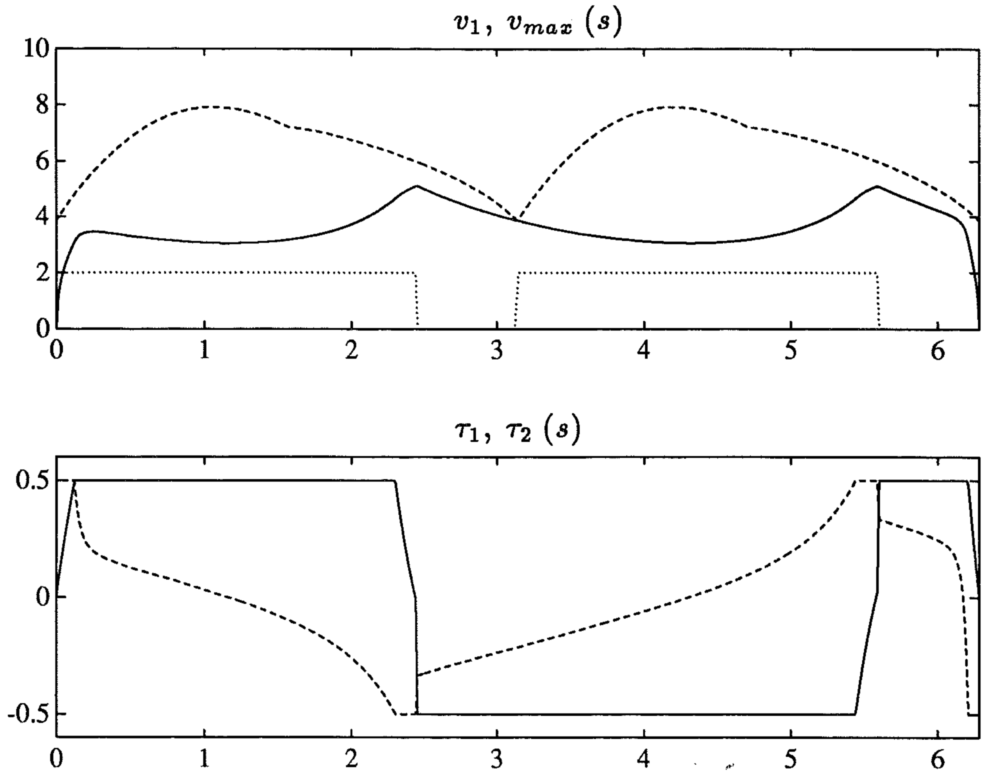


Figure 5.35 The minimum time solution for Experiment 5, and the corresponding torques.

Experiment 5 – Influence of the nominal velocity profile

The preliminary model (5.6) is used in this experiment. The path is shown in the upper plot in Figure 5.3. The nominal minimum time velocity profile, computed from the model (5.6) is shown in Figure 5.35. The result of using this velocity profile as nominal velocity profile for the path velocity controller (4.24) is shown in Figure 5.36. As can be seen from the figure, the path following is good. The torque utilization is however unsatisfactory, as can be seen from the lower right plot. Using the scale factor $\gamma = 1.15$ in the path velocity controller (4.26) gives the result shown in Figure 5.37. The torque utilization has increased from 0.86 to 0.92, and the traversal time has decreased from 1.77 to 1.66. There are however small path deviations, as can be seen in the upper left plot.

An experiment where the nominal velocity profile is instead based on the model (5.7) is shown in Figure 5.38. The traversal time was, as in Figure 5.37, 1.66, but the path following is now better, as can be seen by comparing the upper left plots in Figures 5.37 and 5.38. The comparison shows that if a conservative model is obtained, it is possible to gain performance by using

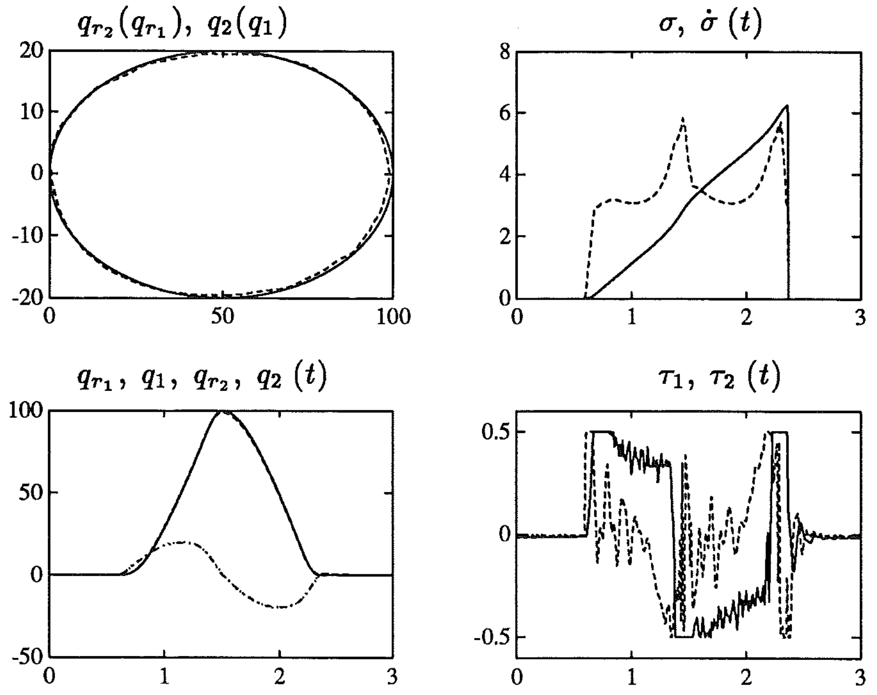


Figure 5.36 The result when the path velocity controller is used. The torque utilization was $\tau_u = 0.86$, the traversal time was $t_f = 1.77$, and the mean square tracking error was $mse = 1.62$.

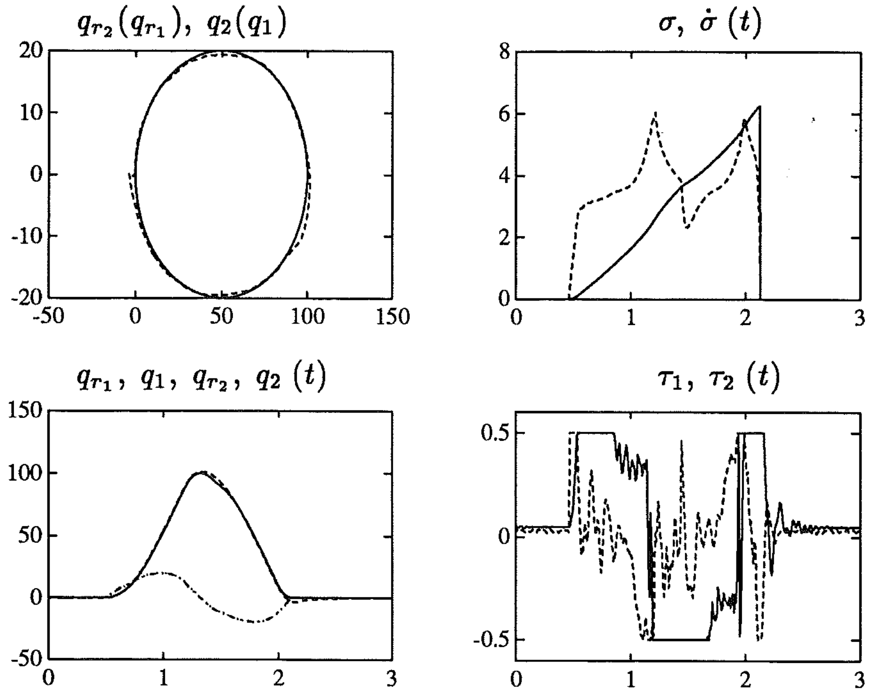


Figure 5.37 The result when the path velocity controller is used with a constant scaling factor $\gamma = 1.15$. The torque utilization was $\tau_u = 0.92$, the traversal time was $t_f = 1.66$, and the mean square tracking error was $mse = 2.90$.

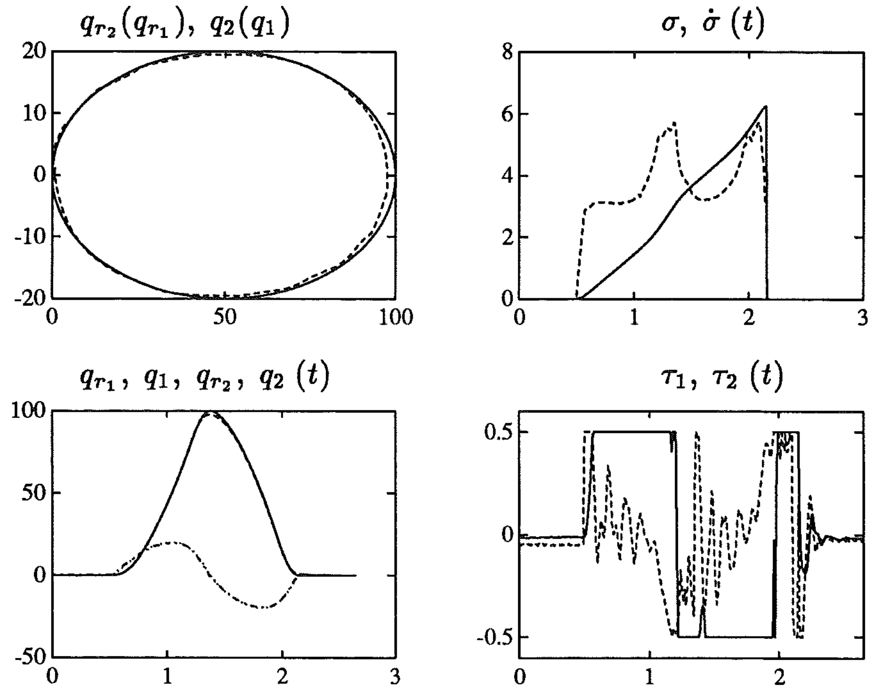


Figure 5.38 The result when the path velocity controller is used with a different nominal velocity profile. The torque utilization was $\tau_u = 0.99$, the traversal time was $t_f = 1.66$, and the mean square tracking error was $mse = 1.90$.

a scaling factor $\gamma > 1$, but also that the same traversal time can be obtained with lower path deviation, by using another model in the minimum time optimization.

5.4 Conclusions

The path velocity controller has been tested experimentally. The experiments were done such that is was possible to separate the performance of the path velocity controller from the performance of the robot controller. The path deviation was evaluated using visual inspection, and a quantitative measure (5.1) for the torque utilization was used.

The experimental results show that the path velocity controller is able to modify a nominal minimum time velocity profile such that the modified velocity profile results in path following, and good utilization of the available torque range. It was shown, Figures 5.7 and 5.18, how the motion obtained when the path velocity controller was used resulted in the same path as the motion obtained without the torque limits. This can to some extent be expected, since the acceleration adjustment done by the path velocity controller has the effect that variations in the torque that would otherwise result in torque saturation, instead, via the adjustment of the path acceleration, become variations in the reference trajectory.

If the limits on acceleration become inadmissible during motion, the path velocity controller is not able to adjust the acceleration such that all torques are inside the limits. It was shown in Figure 5.19 that this occurred during motion. It was observed that although the limits were inadmissible, the motion obtained when the path velocity controller was used, was comparable to the case of no torque limits, Figure 5.18. It was also observed that the inadmissibility of the limits, Figure 5.29, could result in path deviation, in the sense that the path deviation when the path velocity controller was used was slightly larger than for the experiment when the torque limits were removed, Figure 5.28. The inadmissibility of the limits can thus be seen as an explanation of path deviation that is the result of a too high nominal velocity profile. This path deviation is not caused by the robot controller, and can be reduced if the nominal velocity profile is modified.

It was also demonstrated, Figure 5.32, how the path velocity controller could compensate for time varying dynamics. Repeated motion along a path was performed after having the robot kept at zero velocity for 5–10 minutes. The use of path velocity control then resulted in motion along the path, and the adaptation to the time varying dynamics could be seen as decreasing traversal time for repeated runs, Figure 5.33. The path deviation and the torque utilization were kept at satisfactory levels, Figures 5.32 and 5.34.

The path velocity controller has the property that if the nominal velocity profile does not require more torque than what is available, it is not modified. This may result in a conservative motion with low torque utilization, as shown in Figure 5.36. It was shown how a constant scaling of the velocity profile can decrease the traversal time, Figure 5.37, but also how better performance could be obtained using a different nominal velocity profile, Figure 5.38.

6 Conclusions

Fast motion along a predefined path is important in many robot applications, and requires utilization of the maximum allowable torque range. If the torque is at the limit, there is no margin to cope with disturbances or modeling errors, which may result in deviation from the path. This thesis has presented an approach to solve this path following problem by using a path velocity controller for reference trajectory modification. The reference trajectory is modified when the torques saturate. The modification is done such that the modified reference trajectory defines the same path as the nominal. The result is that path following is maintained, at the expense of increased traversal time.

A nominal velocity profile can be obtained using available methods for minimum time optimization. The velocity and acceleration constraints used in the optimization gives insight into the design and interpretation of path velocity control. An optimality result shows that similar bang-bang properties hold for both rigid and flexible joint robots. With the exception of critical points, the p-th order time derivative of the path parameter, $s^{(p)}$, with $p = 2$ for rigid robots, and $p = 4$ for flexible joint robots, should either be maximized or minimized. An approximate minimum time solution for flexible joint robots can be obtained using polynomial approximation. Choosing the function to approximate as $\dot{s}^p/p$ gives the desired boundary conditions on nonzero initial and final $s^{(p)}$. Numerical examples show the properties of the solution, and also how it differs from the rigid solution.

The path velocity controller is used outside the ordinary robot controller. The robot controller is parametrized in the scalar path parameter, but otherwise unchanged, i.e. a well tuned control behavior is kept. A basic algorithm handles the problem of inadmissible path acceleration. The algorithm uses limits on the acceleration, computed from the controller parametrization, together with internal feedback using the nominal velocity profile. The limits give the maximum and minimum acceleration that results in admissible torques. When the acceleration is not saturated, the chosen internal feedback gives a linear system with the path parameter as independent variable, which gives a possibility for tuning using the nominal velocity profile. An extension to higher order systems shows that the feedback can be chosen such that the resulting system becomes linear also for higher order path velocity controllers. The added problem of inadmissible velocity can be handled using velocity profile scaling. An extended algorithm uses feedback for modification of the scaling factor. An approximate analysis of constant scaling gives insight into what type of model errors that can be handled by the path velocity controller.

The path velocity controller has been verified experimentally on an industrial robot. The experiments were done such that is was possible to separate the performance of the path velocity controller from the performance of the robot controller. The path deviation was evaluated using visual inspection, and a quantitative measure for the torque utilization was proposed and used. The experimental results show how the path velocity controller makes it possible to use minimum time optimization in a nonideal situation. It was also demonstrated how the limits on path acceleration obtained during motion can be used for evaluation of the path velocity control performance. The path velocity controller can also be used for compensation for time varying dynamics, where repeated motion along a path resulted in decreased traversal time, while keeping the path deviation and torque utilization at satisfactory levels.

The path velocity controller provides a computationally efficient way to utilize the available torque range by feedback modification of the reference trajectory. The feedback provided by the path velocity controller makes it possible to have a nominal velocity profile which exceeds the robot capability. This is not possible if the reference trajectory is fixed. The use of path velocity control can therefore increase performance in the sense that it is possible to come closer to the minimum possible traversal time, since it is not necessary to introduce a prespecified margin in the path velocity planning to account for modeling errors and disturbances in advance.

7

References

ANDERSSON, L. and A. BLOMDELL (1991): "A real-time programming environment and a real-time kernel." In *National Swedish Symposium on Real-Time Systems.*

ASADA, H. and J.-J. E. SLOTINE (1986): *Robot Analysis and Control.* John Wiley and Sons, New York.

BEN-ASHER, J., J. A. BURNS, and E. M. CLIFF (1987): "Time optimal slewing of flexible spacecraft." In *Proceedings of the 26th Conference on Decision and Control*, pp. 524–528, Los Angeles, CA.

BOBROW, J., S. DUBOWSKY, and J. GIBSON (1985): "Time-optimal control of robotic manipulators along specified paths." *International Journal of Robotics Research*, **4:3**, pp. 3–17.

BRAUN, R., L. NIELSEN, and K. NILSSON (1990): "Reconfiguring an ASEA IRB-6 robot system for control experiments." Technical Report TFRT-7465, Department of Automatic Control, Lund Institute of Technology.

BRYSON, A. E. and Y. HO (1975): *Applied Optimal Control.* Hemisphere Publishing Corporation, New York.

CHAR, B. W., K. O. GEDDES, G. H. GONNET, M. B. MONAGAN, and S. M. WATT (1988): *Maple – Reference Manual.* Symbolic Computation Group, Department of Computer Sciences, University of Waterloo, Waterloo, Ontario, Canada, fifth edition.

CHEN, Y. (1988): *Minimum Time Control of Robotic Manipulators.* PhD thesis, Rensselaer PolyTechnic Institute, Troy, New York.

DAHL, O. (1989): "Torque limited path following by on-line trajectory time scaling." Licentiate Thesis TFRT–3204, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

DAHL, O. (1991): "An interactive environment for real time implementation of control systems." In *Computer Aided Design of Control Systems, Preprints of the IFAC Symposium*, Swansea, UK.

DAHL, O. (1992): "A symbolic/numeric method for time optimal control of resonant systems." Technical report, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden. To appear.

DAHL, O. and L. NIELSEN (1989): "Torque limited path following by on-line trajectory time scaling." In *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, pp. 1122–1127, Scottsdale, Arizona.

DAHL, O. and L. NIELSEN (1990a): "Stability analysis of an on-line algorithm for torque limited path following." In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, pp. 1216–1222, Cincinnati, Ohio.

DAHL, O. and L. NIELSEN (1990b): "Torque limited path following by on-line trajectory time scaling." *IEEE Transactions on Robotics and Automation*, **6:5**.

DE BOOR, C. (1978): *A Practical Guide to Splines.* Springer-Verlag, New York.

DE BOOR, C. (1990): *Spline Toolbox for use with Matlab.* The MathWorks, Inc., Cochituate Place, 24 Prime Parkway, Natick, MA 01760, USA.

GILL, P. E., W. MURRAY, M. A. SAUNDERS, and M. H. WRIGHT (1986): "User's guide for NPSOL." Technical report, Department of Operations Research, Stanford University.

HOLLERBACH, J. (1984): "Dynamic scaling of manipulator trajectories." *ASME J. Dynamic Systems, Measurement, and Control*, **106**, pp. 102–106.

LEITMANN, G. (1981): *The Calculus of Variations and Optimal Control.* Plenum Press, New York.

LIPPMAN, S. B. (1989): *C++ Primer.* Addison-Wesley.

LJUNG, L. (1991): *System Identification Toolbox User's Guide.* The MathWorks, Inc., Cochituate Place, 24 Prime Parkway, Natick, MA 01760, USA.

MARIN, S. P. (1988): "Optimal parametrization of curves for robot trajectory design." *IEEE Transactions On Automatic Control*, **33:2**, pp. 209–214.

MATHWORKS (1990): *PRO-MATLAB – User's Guide.* The Mathworks, Inc., Cochituate Place, 24 Prime Parkway, Natick, MA 01760, USA.

MATTSSON, U. (1991): "Digital reglering med signalprocessorer och C++." Technical Report TFRT-5434, Department of Automatic Control, Lund Institute of Technology. Swedish.

NILSSON, K. (1992a): "Application oriented programming and control of industrial robots." Licentiate thesis, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden. To appear.

NILSSON, K. (1992b): "A matlab interface to real time control systems." Technical report, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden. To appear.

PFEIFFER, F. and R.JOHANNI (1986): "A concept for manipulator trajectory planning." In *IEEE Conf. Robotics and Automation*, San Francisco.

SHILLER, Z. and H. LU (1990): "Computation of path constrained time optimal motions with dynamic singularities." Technical report, Laboratory for Robotics Automation and Manufacturing, Department of Mechanical, Aerospace and Nuclear Engineering, University of California at Los Angeles.

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

SHIN, K. and N.D.MCKAY (1987): "Robust trajectory planning for robotic manipulators under payload uncertainties." *IEEE Transactions On Automatic Control*, **32:12**, pp. 1044–1054.

SLOTINE, J.-J. E. and M. W. SPONG (1985): "Robust robot control with bounded input torques." *Journal of Robotic Systems*, **2:4**, pp. 329–352.

SPONG, M. W. (1987): "Modeling and control of elastic joint robots." *ASME Journal of Dynamic Systems, Measurement, and Control*, **109**, pp. 310–319.

SPONG, M. W. (1990): "Control of flexible joint robots: A survey." Technical report, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign.

A

Implementation

The implementation used in Chapter 5 is described. The velocity and acceleration profiles v_1 and v_2, and the path description f, f', and f'' are stored in vectors. This is however not a necessary requirement, since only the current values of the states σ and $\dot{\sigma}$ are used in the path velocity control algorithm, see e.g. (4.24) and (4.3), and it is therefore not necessary to store v_1, v_2, and the path description in advance. It would e.g. be possible to use the path velocity controller as implemented here in combination with a supervisory system for run-time generation of v_1 and v_2 and/or run-time generation of the path description. In Chapter 5, v_1 and v_2 were obtained from minimum time optimization. The path velocity control algorithm is not dependent on this choice, and the implementation can be combined also with other types of optimization, e.g. minimum time with both torque constraints and velocity constraints.

The robot controller and the path velocity controller are implemented in C++, e.g. [Lippman, 1989], and run in the DSP, see Figure 5.1. The path description is implemented in a separate class, where the current state $(\sigma, \dot{\sigma})$ of the path velocity controller also is stored. The actual control, i.e. the execution of the robot controller and the path velocity controller is implemented as two procedures which are executed periodically. This means that the reference computations and the robot controller are executed at the same sampling rate, and there is no separate reference trajectory generation.

The robot controllers for the individual joints are represented as five instances of a class, where there are procedures e.g. for computation of the jointwise limits on $\ddot{\sigma}$, (4.12) and (4.13). The actual limits (4.14) are computed separately by combining the individual bounds. The use of jointwise limitation of $\ddot{\sigma}$ means that the computation of the limits on $\ddot{\sigma}$ is distributed over the different robot controllers, which has the advantage that the implementation can easily be extended, e.g. to handle more joints, or to obtain synchronization with external axes, as described in [Nilsson, 1992a].

A.1 Implementation of Path Velocity Control

The implementation is described, and selected pieces of code are listed. The code is complete with respect to the control aspects, but the code for e.g. parameter transfer and storage of data that should be plotted have been omitted.

Robot Controller

The robot controller (5.4) is parametrized as (4.1) with

$$\begin{aligned}\beta_{1_i} &= \hat{m}_i f'(\sigma) \\ \beta_{2_i} &= \hat{m}_i f''(\sigma)\dot{\sigma}^2 + \hat{d}_i f'(\sigma)\dot{\sigma} + k_{v_i}(f'(\sigma)\dot{\sigma} - \dot{q}_i) + k_{p_i}(f(\sigma) - q_i)\end{aligned} \tag{A.1}$$

where $1 \leq i \leq 5$. The robot controller is implemented in a C++ class. The five robot controllers for the individual joints are obtained as five instances of this class. The class interface is shown in Listing A.1. The state variable `xf` in Listing A.1 is used for filtering of the robot velocity, which is computed by taking differences of the position measurements. The filter is a first order filter with time constant 0.02 seconds. The procedure `computebeta` computes β_{1_i} and β_{2_i} according to (A.1). The robot velocity $\dot{q}_i$ is given in the variable `dq`, the current values of the path functions $f(\sigma)$, $f'(\sigma)$ and $f''(\sigma)$ are given in `qr`, `qrd`, and `qrdd`, and the path velocity $\dot{\sigma}$ is given in `ds`. The procedure `ddsminmax` computes jointwise limits on $\ddot{\sigma}$, i.e. $\ddot{\sigma}^i_{max}(\beta_{1_i}, \beta_{2_i})$ in (4.12) and $\ddot{\sigma}^i_{min}(\beta_{1_i}, \beta_{2_i})$ in (4.13) are returned in the variables `ddsmax` and `ddsmin`. The procedure `computeu` gives the limited value of τ_i, i.e. the limitation of $\tau_i = \beta_{1_i}\ddot{\sigma} + \beta_{2_i}$ by the torque limits `umin` and `umax`.

Path Velocity Controller

The class interface to the path velocity controller is described in Listing A.2. The number of joints, here five, is given in the variable `njs`, which defines the length of e.g. the vector `q` in Listing A.2. The vectors `q`, `u`, and `qold` contains the current values of the joint positions q, the torque τ, and the previous joint position $q(t-h)$, where h is the sampling interval. Selected variables, e.g. the input vector `q` and the output vector `u` are sent to the Sun, see Figure 5.1, for plotting and/or storage in files. This is done by the private procedure

```
class PD_Controller {
  /* state */ float xf;
  /* par */ float kp, kv, m, d, umin, umax;
  /* auxvar */ float beta1, beta2;
public:
  PD_Controller();
  ~PD_Controller();
  void computebeta(
    float q, float dq,
    float qr, float qrd, float qrdd, float ds);
  void ddsminmax(float &ddsmin, float &ddsmax);
  float computeu(float dds);
  void writepars(
    float kp, float kv, float m, float d,
    float umin, float umax);
  void readpars(
    float &kp, float &kv, float &m, float &d,
    float &umin, float &umax);
};
```

Listing A.1 The class interface for the robot controller

sendtoplot, which is called by newstate. The procedure sendtoplot stores the values of the variables that should be sent to the Sun in a common memory area, where they then are collected by the M68030, see Figure 5.1. The variable P is the path representation. The class interface to this variable is described below in Listing A.4. The variable RobotController in Listing A.2 is a vector of the five robot controllers. The class interface to the robot controller was given in Listing A.1. The variables alfa, scale, smaxeps, dsmin, and delay are parameters that are used in the path representation P. The parameter alfa is α in (4.24), scale is the constant scaling factor γ in (4.26), smaxeps is used for detection of completed motion, i.e. the motion is considered as completed when $s_f - \sigma$ is less than smaxeps, dsmin is a minimum value of $\dot{\sigma}$, and delay is a delay time between each completed motion along the path. The actual control is done by periodic execution of the procedures update and newstate. The implementation of these procedures is shown in Listing A.3. The procedure call P.getref in Listing A.3 gives the current values of $f(\sigma)$, $f'(\sigma)$, $f''(\sigma)$, σ, and $\dot{\sigma}$, and also the current value of the variable u_r in (4.24). The calls to computebeta and ddsminmax for the individual joints then give the jointwise limits (4.12) and (4.13). The actual limits (4.14) are then computed by the procedure calls min5 and max5. It

```
class PVC {
  /* input  */ float q[njs];
  /* output */ float u[njs];
  /* state  */ float qold[njs];
  /*  par */ float kp[njs], kv[njs], m[njs], d[njs],
             umin[njs], umax[njs];
  /* par */ float alfa, scale, smaxeps, dsmin, delay;
  /* par */ float h;
  /* auxvar */ float qr[njs], qrd[njs], qrdd[njs];
  /* auxvar */ float s, ds, dds, ddsmin, ddsmax, limused;
  Path P;
  PD_Controller RobotController[njs];
public:
  PVC();
  ~PVC();
  void update(float* q, float* u);
  void newstate();
private:
  void sendtoplot();
};
```

Listing A.2 The class interface for the path velocity controller

is then checked if the limits should be used or not, i.e. it is checked if $\ddot{\sigma}_{min} < \ddot{\sigma}_{max}$ and if u_r is outside the limits. If the limits are admissible, i.e. if $\ddot{\sigma}_{min} < \ddot{\sigma}_{max}$, the procedure call `sat(ur,ddsmin,ddsmax)` gives the limited $\ddot{\sigma}$. If the limits are not admissible, the assignment `dds = ur` gives $\ddot{\sigma} = u_r$, as was discussed in Chapter 4, see the discussion following (4.24). The controller output τ is then computed by `computeu` as $\tau = \beta_1\ddot{\sigma} + \beta_2$.

The procedure call `P.updateref` in Listing A.3 updates the states σ and $\dot{\sigma}$ in the path velocity controller, using a forward Euler discretization. The procedure `updateref` also contains logic for handling the case $\sigma = s_f$, which means that the reference trajectory has traversed the path. If this is the case, a delay is introduced by setting a counter. This means that between each path traversal there is a delay where the reference values given by the procedure call `P.getref(qr,qrd,qrdd,s,ds,ur)` in listing A.3 gives zero velocity and acceleration, i.e. `ds = ur = 0`, and the robot is kept at a constant position. The implementation of the procedures `getref` and `updateref` are shown in Listings A.5 and A.6. The interface to the path class is shown in Listing A.4. The variable `svec` in Listing A.4 is a vector containing the discretized values of σ, i.e. a discretization of the interval $[s_0, s_f]$. The values in this vector are

```
void PVC::update(float* nq, float* nu) {
  float dq[njs], ddsminvec[njs], ddsmaxvec[njs];
  float ur;
  for (int i = 0; i < njs; ++i) {
    q[i] = nq[i];
    dq[i] = (q[i] - qold[i])/h;
  }
  P.getref(qr,qrd,qrdd,s,ds,ur);
  for (i = 0; i < njs; ++i) {
    RobotController[i].computebeta(
      q[i],dq[i],qr[i],qrd[i],qrdd[i],ds);
    RobotController[i].ddsminmax(
      ddsminvec[i],ddsmaxvec[i]);
  }
  ddsmax = min5(ddsmaxvec);
  ddsmin = max5(ddsminvec);
  if ((ddsmin < ddsmax) && (ur > ddsmax || ur < ddsmin))
    limused = 1;
  else
    limused = 0;
  if (ddsmin < ddsmax)
    dds = sat(ur,ddsmin,ddsmax);
  else
    dds = ur;
  for (i = 0; i < njs; ++i) {
    u[i] = RobotController[i].computeu(dds);
    nu[i] = u[i];
  }
}

void PVC::newstate() {
  sendtoplot();
  P.updateref(h,dds);
  for (int i = 0; i < njs; ++i) {
    yold[i] = y[i];
  }
}
```

Listing A.3 The implementation of the path velocity controller

```
class Path {
  floatvec svec, v1, v2;
  fmatrix f, fd, fdd;
  int waitcount, waiting;
  /* par */ float alfa, scale, smaxeps, dsmin, delay;
  /* state */ float s, ds;
public:
  Path();
  ~Path();
  void writepars(
    float alfa, float scale, float smaxeps,
    float dsmin, float delay);
  void getref(float* qr, float* qrd, float* qrdd,
    float &s, float &ds, float &ur);
  void updateref(float h, float dds);
private:
  int getindex(float s);
};
```

Listing A.4 The interface to the path representation

accessed by the private procedure `getindex`, which gives the position in the vector `svec` corresponding to the value `s`. The variables `v1` and `v2` are the nominal velocity and acceleration profiles $v_1(\sigma)$ and $v_2(\sigma)$. The variables `f`, `fd`, and `fdd` contain the path description, i.e. the functions $f(\sigma)$, $f'(\sigma)$, and $f''(\sigma)$. The variables `waitcount` and `waiting` are used to keep the timing between different runs. The variables `s` and `ds` are the current values of σ and $\dot{\sigma}$.

Main Program

Both the path velocity controller and the robot controllers are run as one process by a periodically executed routine, shown in Listing A.7, where the procedure `docontrol` is called every sample.

```
void Path::getref(float* qr, float* qrd, float* qrdd,
  float &sout, float &dsout, float &ur) {

  int index = getindex(s);
  for (int i = 0; i < njoints; ++i) {
    qr[i] = f[i][index];
    qrd[i] = fd[i][index];
    qrdd[i] = fdd[i][index];
  }
  sout = s;
  if (waiting) {
    dsout = 0;
    ur = 0;
  }
  else {
    dsout = ds;
    ur = scale*scale*v2[index] +
         alfa/2*(scale*scale*v1[index]*v1[index] - ds*ds);
  }
}
```

Listing A.5 The implementation of the procedure **getref**. The variable ur is computed according to (4.26).

```
void Path::updateref(float h, float dds) {
  if (waiting) {
    waitcount = waitcount + 1;
    if (waitcount > delay) {
      waiting = 0;
      waitcount = 0;
    }
  }
  else {
    float news = s + h*ds;
    float newds = ds + h*dds;
    s = news;
    ds = newds;
    if (ds < dsmin) ds = dsmin;
    if (s > svec[maxindex] - smaxeps) {
      waiting = 1;
      s = svec[0];
      ds = v1[0];
    }
  }
}
```

Listing A.6 The implementation of the procedure updatref. The states σ and $\dot{\sigma}$ are stored in the variables s and ds, and are updated using forward Euler approximation.

```
void docontrol() {
  y = ServoIO.abs5Pos(); //AD-conversion
  PVCinstance.update(y,u);
  ServoIO.Ref5Out(u); //DA-conversion
  PVCinstance.newstate();
}
```

Listing A.7 The periodically executed routine for path velocity control.