Torque Limited Path Following by On-Line Trajectory Time Scaling

Dahl, Ola

1989

Document Version:
Publisher's PDF, also known as Version of record

Torque Limited Path Following by On-line Trajectory Time Scaling

Ola Dahl

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
Licentiate Thesis

Date of issue
May 1989

Document Number
CODEN: LUTFD2/(TFRT-3204)/1–86/(1989)

Author(s)
Ola Dahl

Supervisor
Lars Nielsen, Karl Johan Åström

Sponsoring organisation
Swedish National Board for Technical Development, contract 87-01384P

Title and subtitle
Torque Limited Path Following by On-line Trajectory Time Scaling

Abstract

Fast motion along a geometric path is an important problem in robotics. The motion is limited by torque constraints, and the goal is to have fast motion along the path, without violating the torque constraints. The path is given by the application, and a controller for reference trajectory tracking is assumed available. Further, a nominal reference trajectory is available from an off-line procedure, e.g. a minimum time optimization. The problem studied here is the design of a secondary control loop for on-line modification of the nominal trajectory. The nominal, high performance trajectory typically leads to torques that are at the limits, hence leaving no control authority to compensate for modeling errors and disturbances. By modifying the time scaling of the nominal trajectory when the torques saturate, the reference trajectory used by the primary controller is adjusted to meet the requirements given by the torque constraints, thereby leaving room for closed loop action. A key idea is that a scalar quantity, the path acceleration, is modified, resulting in coordinated adjustment of the individual joint motions. Two algorithms for on-line trajectory scaling are presented. One is based on on-line bounds on path acceleration, and one is designed to handle nominal minimum time trajectories. The primary controller is parametrized in the scalar path coordinate, but otherwise not changed. The functionality of the secondary controller is verified by simulations, experiments, and analysis.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
86

Recipient's notes

Security classification

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S–221 00 LUND
Sweden

Published 1989
Printed in Sweden

Acknowledgements

This work was supported by the Swedish National Board for Technical Development under contract 87-01384P, and carried out in a friendly and stimulating environment at the Department of Automatic Control. I would like to express my gratitude to my supervisor Lars Nielsen. He has provided continuous guidance and support during the work, and it has been a great pleasure to work with him. I would also like to thank Per Hagander for valuable criticism on the manuscript. I am grateful to Bo Bernhardsson for his comments on the manuscript, and to Klas Nilsson for valuable discussions on different aspects of industrial robotics. Prof. Karl Johan Åström is gratefully acknowledged for creating a stimulating environment at the department, and for valuable discussions of my work. Finally, I thank my wife Pia, for her understanding, support and patience.

Contents

1. Introduction

Fast motion along a geometric path is a central problem in high performance robotics. Typical applications are gluing, arc welding of small pieces, and laser or high pressure water cutting. A glue string, for example, has to be applied very precisely to obtain the desired adhesion effect but also to prevent the creation of corrosion pockets. Further, overflow should be prevented when pressing the glued pieces together. In these applications the robot performance is limiting the production speed, and thus it is of general interest to be able to perform a path as fast as possible. The path is of course given from the application and a first step is to obtain robot trajectories. The trajectories are precalculated by interactive programming, teach-in, minimum-time or other types of optimization. The so obtained trajectories are typically, at least for one joint, at the torque limit. Therefore, there is no control authority left for these joints to take care of disturbances or modeling discrepancies. The objective of this thesis is to execute such fast trajectories, along a given path, by taking errors into account in a feedback scheme for on-line trajectory modification.

A secondary feedback loop is used for modification of a nominal trajectory, and the new idea is to have a feedback scheme that modifies only the time scaling of the nominal trajectory, resulting in a trajectory still moving on the given path, but perhaps at a slower speed than the nominal. The scheme leaves the original robot and primary controller unchanged, only the reference trajectory is modified. This means that the basic dynamic properties, e.g. robustness, is preserved. The time scaling is done by modification of a scalar quantity. This simplifies the design of the secondary controller, and results in coordinated adjustment of the individual joint motions.

The thesis is organized as follows. Trajectory planning and execution are treated in Chapter 2. A short review of minimum time trajectory planning is given, and the problem of executing fast trajectories, when the motion is limited by torque constraints, is discussed. References to related work are also given. The main results are given in Chapter 3 and Chapter 4. Chapter 3 describes the main ideas in our concept for trajectory time scaling. The primary controller is written in a form

well suited for devising the secondary loop, and algorithms for feedback trajectory scaling to obtain path following, are designed and analyzed. The proposed algorithms are verified by simulations and experiments. Chapter 4 contains further analysis and discussion, and the conclusions are given in Chapter 5.

Chapters 1-3 are based on a paper at the 1989 IEEE Conference on Robotics and Automation (Dahl and Nielsen, 1989). The experiments presented in Chapter 3 were performed using an extended version of a compiler, described in (Dahl, 1989), to simplify the real time implementation of the algorithms.

2. Trajectory Planning and Execution

Our scheme for trajectory modification uses a nominal trajectory as input. This chapter describes model based trajectory planning as a method for obtaining the nominal trajectory. Trajectory planning is here meant as a calculation of a trajectory from a given path. Model based trajectory planning is often based on optimization, for example minimum time or other criteria (Bobrow, Dubowsky, and Gibson, 1983, 1985; Shin and McKay, 1985; Pfeiffer and Johanni, 1986). The algorithms use a dynamic model of the robot together with a description of the path and the input constraints. In the case of minimum time optimization, the so obtained nominal trajectories are of a bang-bang character, meaning here that at every time instant, at least one of the joint torques are at the limit. Minimum time optimization can thus be seen as a "worst-case" test for execution of fast trajectories along a geometric path, when the motion is limited by torque constraints, and a dynamic model is used for the trajectory planning. We have therefore restricted the presentation in this chapter to minimum time even though our work on trajectory execution is not limited to that case.

Section 2.1 describes the robot dynamics along a given path. The torque constraints are converted to constraints on speed and acceleration, and a geometric interpretation of the constraints is given. A short review of minimum time trajectory planning (Bobrow, Dubowsky, and Gibson, 1983, 1985; Shin and McKay, 1985; Pfeiffer and Johanni, 1986) is given in Section 2.2. The problem of executing fast trajectories, when the motion is limited by torque constraints is discussed in Section 2.3.

2.1 Robot Dynamics and Torque Constraints

The rigid body dynamics of a robot can be written as

$$H(q)\ddot{q} + v(q,\dot{q}) + d(q)\dot{q} + g(q) = \tau \tag{2.1}$$

where $q \in R^n$ is the vector of joint variables, $\tau \in R^n$ 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). Assume that the path is given in parametrized form as a vector function $f(s) \in R^n$ of the scalar path parameter $s \in R$, $s_0 \leq s \leq s_1$, where $f(s_0)$ is the starting point and $f(s_1)$ is the end point of the path. Moving the robot along the path gives

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

where $' = \frac{d}{ds}$, and it is assumed that the appropriate derivatives exist. Using the fact that the elements of the coriolis and centrifugal vector $v(q,\dot{q})$ are of the form $v_i = \sum_{j,k} v_{ijk}(q)\dot{q}_j\dot{q}_k$, the dynamic equation (2.1) can now be written as

$$a_1(s)\ddot{s} + a_2(s)\dot{s}^2 + a_3(s)\dot{s} + a_4(s) = \tau \tag{2.2}$$

where

$$\begin{aligned} a_1(s) &= H(f(s))f'(s) \\ a_2(s) &= H(f(s))f''(s) + v(f(s), f'(s)) \\ a_3(s) &= d(f(s))f'(s) \\ a_4(s) &= g(f(s)) \end{aligned} \tag{2.3}$$

The torque constraints are in general given by a region $E(q,\dot{q})$ in the input space where the admissible torques satisfy $\tau \in E$, but typically each component of the torque vector is upper and lower limited by constants leading to a hyper rectangle as torque constraint region E. The torque constraints can be converted to constraints on the path speed $\dot{s}$, and on the path acceleration $\ddot{s}$, by the following reasoning. Given s, $\dot{s}$, and bounds on τ, the admissible values of $\ddot{s}$ are obtained from (2.2). Now given only s and bounds on τ, the admissible values of $\dot{s}$ are those where there exist at least one admissible $\ddot{s}$ from equation (2.2). The resulting constraints on $\dot{s}$ and $\ddot{s}$ can be illustrated by plotting the boundary cases for $\dot{s}$ admissible as a function of s. This gives a region in the s-$\dot{s}$-space containing the admissible values of $\dot{s}$, from now on referred to as the admissible region. It is a necessary condition on a feasible trajectory that the $\dot{s}(s)$-curve is inside the admissible region.

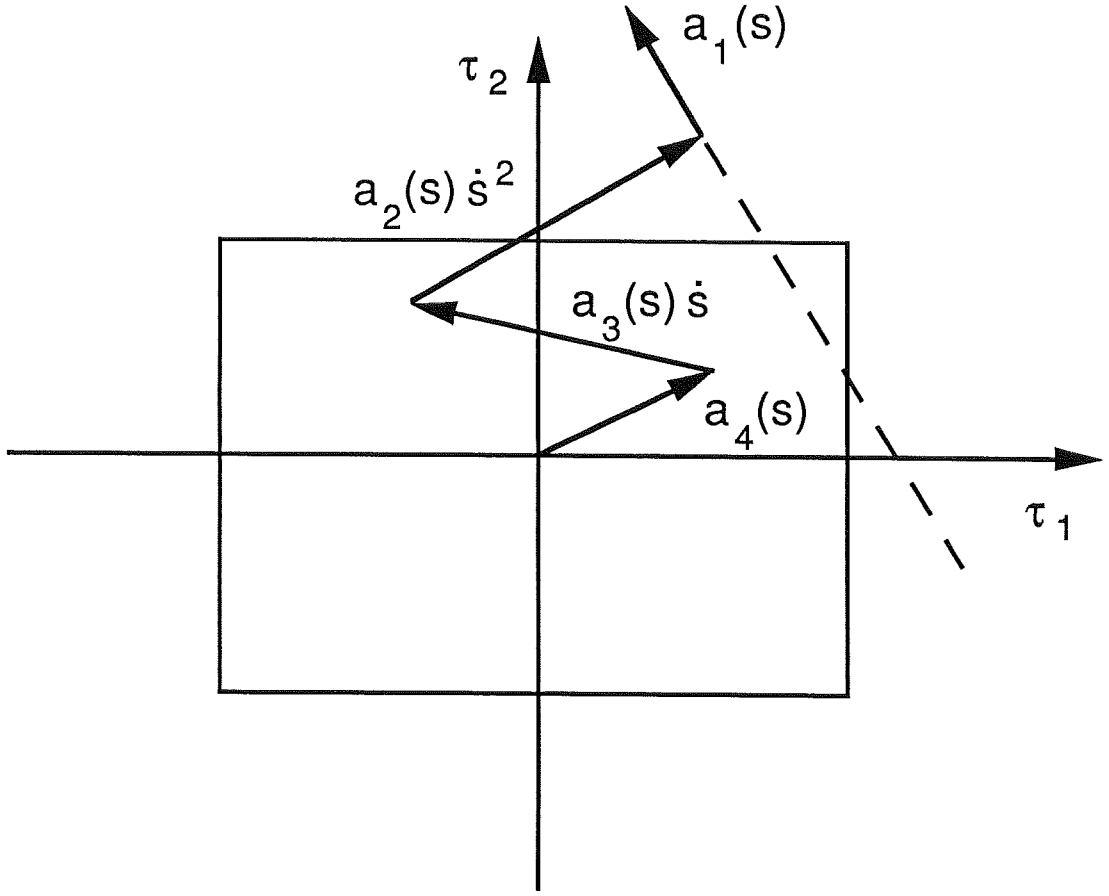

Figure 2.1 A geometric interpretation of the constraints on $\dot{s}$ and $\ddot{s}$. The region E, where τ is admissible, is illustrated by a rectangle. The continuation of the vector a_1 intersects the region E, which shows that the path speed $\dot{s}$ is admissible. The bounds for $\ddot{s}$ admissible are given by the intersection points.

A Geometric Interpretation of the Constraints

The constraints on $\dot{s}$ and $\ddot{s}$ can also be given a geometric interpretation. The torque vector τ, given by equation (2.2), is written as

$$\tau = a_1(s)\ddot{s} + a_{12}(s, \dot{s})$$

The admissible torques are satisfying $\tau \in E$. Given s and $\dot{s}$, consider the line with a direction given by $a_1(s)$, and passing through the end point of the vector $a_{12}(s, \dot{s})$. The boundary cases for the admissible values of $\ddot{s}$ are then found as the intersections between the 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 speed $\dot{s}$ is outside the admissible region. Figure 2.1 illustrates the constraints on $\dot{s}$ and $\ddot{s}$ for the two-joint case.

2.2 Minimum Time Trajectory Planning

As shown in the previous section, the torque constraints can be converted to constraints on the path speed $\dot{s}$, and the path acceleration $\ddot{s}$. The trajectory planning problem can then be formulated as an optimal control problem with input constraints and state constraints. The controlled system is a double integrator with input $\ddot{s}$, and the states are s and $\dot{s}$, which is more tractable than the original $2n$-states $(q, \dot{q})$. The states s and $\dot{s}$ are constrained by the admissible region, and the input $\ddot{s}$ is constrained by eq. (2.2). In the case of minimum time, the objective function is the traversal time

$$t_f = \int_0^{t_f} dt = \int_{s_0}^{s_1} \frac{dt}{ds} ds = \int_{s_0}^{s_1} \frac{1}{\dot{s}} ds$$

Algorithms for minimum time trajectory planning are found in (Bobrow, Dubowsky, and Gibson, 1983, 1985; Shin and McKay, 1985; Pfeiffer and Johanni, 1986), where the minimum time optimization is solved by constructing a number of curves inside the admissible region. Each curve is the result of integrating either $\ddot{s} = \ddot{s}_{max}(s, \dot{s})$ or $\ddot{s} = \ddot{s}_{min}(s, \dot{s})$ backward or forward, i.e. with decreasing or increasing s, where $\ddot{s}_{max}$ and $\ddot{s}_{min}$ are computed from the constraints on $\ddot{s}$. The initial points for the integration are the starting point $(s_0, \dot{s}_0)$, the stopping point $(s_1, \dot{s}_1)$, and points at the boundaries of the admissible region. The result of the optimization is a graph $\dot{s}(s)$ consisting of segments with either maximum or minimum acceleration. One example of the optimization can be seen in Figure 3.3.

Nominal Minimum Time Trajectory

The minimum time trajectory computed by the trajectory planning algorithm can, as any other trajectory, be represented either as a trajectory or as torques $\tau(t)$, see equation (2.1). The trajectory is described in path coordinates by $s(t)$, $\dot{s}(t)$, and $\ddot{s}(t)$, or described in joint space by $q(t)$, $\dot{q}(t)$, and $\ddot{q}(t)$. The nominal minimum time trajectory has at every point either maximum or minimum path acceleration $\ddot{s}$, which means that at least one joint at the time is at the torque limit. Furthermore, the trajectory may touch the boundary of the admissible region. For example, if the trajectory approaches a boundary point with maximum deceleration (Figure 3.3, $s = \pi/2$), the speed at the starting point of the deceleration is the highest possible speed that results in a trajectory inside the

admissible region. Since the optimality of the trajectory is based on the assumption of a perfect dynamic model of the robot, this may not be the case when the trajectory is executed. If, during execution, a situation where the path speed is too high occurs, the rest of the motion cannot be continued without moving away from the path. This indicates that problems may arise during the execution of the nominal minimum time trajectory.

2.3 Trajectory Execution

Trajectory planning is an off-line procedure resulting in a nominal trajectory to be used as a reference trajectory for the robot's control system. High performance trajectories are typically at the torque limit, and more specifically if it is a minimum time trajectory then one or more joint torques is always at the limit. It is therefore custom to be conservative on the requirements and reduce the assumed torque bounds to leave room for closed-loop control action (Asada and Slotine, 1986, Section 6.6). Another method in the same spirit is described in (Shin and McKay, 1987), where bounds on parametric model errors are used for off-line modification of the trajectories. In (Slotine and Spong, 1985), an on-line adjustment scheme is proposed based on pointwise optimal control, where the trajectory is modified on-line by changing the reference trajectory. The modified trajectory is executed in the same time as the nominal trajectory. The path traced by the modified reference trajectory is however different from the nominal path, and path tracking is violated. The idea is thus to do the best possible in nominal time.

Path Tracking by Trajectory Scaling

It is obvious from the optimization problem but also from physical reasons that if the robot is behind the nominal trajectory then it is impossible to catch up if the controller already is at the torque limit. A constructive way to avoid the problem is to slow down the reference trajectory, at least in the applications mentioned in Chapter 1 where path tracking is at priority. Further, it is untractable to be conservative already in the trajectory planning stage because of productivity reasons. We will in the rest of this thesis propose a scheme for on-line modification of the time scaling of the reference trajectory, where the goal is to keep following the path at the expense of an increase in the path traversal

time. Ideally, the traversal time should only increase if needed and then as little as possible. The proposed scheme leaves the primary controller unchanged, only the reference trajectory is changed. This keeps the dynamic properties of the closed loop system, e.g. robustness is preserved. A secondary control loop is used for modification of only a scalar variable, which simplifies the design and analysis of the secondary controller.

3. Path Following by Trajectory Time Scaling

The proposed scheme for feedback trajectory scaling is described in this chapter. In Section 3.1, the primary controller is parametrized in the scalar path coordinate s. Two algorithms for trajectory scaling are designed and analyzed in Section 3.2. The first algorithm uses on-line bounds on the path acceleration $\ddot{s}$ together with feedback from the nominal path velocity. The second algorithm is an extension of the first algorithm, designed to handle nominal minimum time trajectories. A method for obtaining the nominal trajectory from a differential equation is proposed, and used as the basis for the design of the secondary controller. The analysis presented is based on the interpretation of the path parameter s as a transformed time scale. Simulations and experimental results are presented in Section 3.3 and Section 3.4, and a summary of the results is given in Section 3.5.

3.1 Path Parametrization of the Controller

The primary controller, designed for good performance, disturbance rejection etc., is kept unchanged in our scheme for trajectory scaling. The primary controller is however parametrized in the path parameter s, a parametrization of the same type as the parametrization of the robot model used in trajectory planning, equation (2.2). Such a parametrization of the controller seems not to have been reported before, and it is used in our concept for trajectory scaling as the basis for connecting measurements to the path parameter s. We will here use the controller parametrization

$$\tau = b_1(s, \dot{s}, q, \dot{q})\ddot{s} + b_2(s, \dot{s}, q, \dot{q}) \tag{3.1}$$

Note that b_1 and b_2 depend on measured quantities. Compare with equation (2.2), that is used to compute off-line bounds on $\ddot{s}$. We will use

eq. (3.1) to compute on-line bounds on the path acceleration $\ddot{s}$, which in turn will be used for trajectory scaling.

To exemplify the controller parametrization, two common controllers, feedforward and computed torque (Asada and Slotine, 1986), are written in the form (3.1). A feedforward controller with position and velocity feedback is given by

$$\tau = \hat{H}(q_r)\ddot{q}_r + \hat{v}(q_r, \dot{q}_r) + \hat{d}(q_r)\dot{q}_r + \hat{g}(q_r) + K_p e + K_v \dot{e} \tag{3.2}$$

where K_p and K_v are feedback matrices. The reference trajectory is denoted q_r, and the the tracking error e is defined as $e = q_r - q$. The variables $\hat{H}$, $\hat{v}$, $\hat{d}$, and $\hat{g}$ represent an available model of the robot. Executing the trajectory along the path $f(s)$ means that $q_r(t) = f(s(t))$. Taking derivatives and inserting into equation (3.2) gives

$$\begin{aligned} b_1(s) =& \hat{H}(f(s))f'(s) \\ b_2(s, \dot{s}, q, \dot{q}) =& (\hat{H}(f(s))f''(s) + \hat{v}(f(s), f'(s)))\dot{s}^2 \\ &+ \hat{d}(f(s))f'(s)\dot{s} + \hat{g}(f(s)) + K_p e + K_v \dot{e} \end{aligned}$$

A computed torque controller is given by

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

resulting in

$$\begin{aligned} b_1(s, q) =& \hat{H}(q)f'(s) \\ b_2(s, \dot{s}, q, \dot{q}) =& \hat{H}(q)(f''(s)\dot{s}^2 + K_p e + K_v \dot{e}) + \hat{v}(q, \dot{q}) \\ &+ \hat{d}(q)\dot{q} + \hat{g}(q) \end{aligned}$$

On-line Constraints

We can compute on-line constraints on $\dot{s}$ and $\ddot{s}$, in analogy with the off-line constraints in Chapter 2. The on-line admissible values of $\ddot{s}$ are the values that result in admissible τ, i.e. $\tau \in E$, the set of admissible torques, when τ is now computed from equation (3.1). The on-line admissible values of $\dot{s}$ are then given as the values of $\dot{s}$ that results in on-line admissible $\ddot{s}$. The on-line maximum and minimum values of $\ddot{s}$ are computed by the same computational procedure as in the case of off-line

trajectory planning, but now by using equation (3.1) where b_1 and b_2 depend on measured quantities. Constant bounds on the input torques are written as

$$\tau_i^{min} \leq \quad \tau_i = b_{1i}\ddot{s} + b_{2i} \quad \leq \tau_i^{max}, \quad 1 \leq i \leq n \tag{3.3}$$

Each joint i now gives upper and lower bounds on $\ddot{s}$, computed by

$$c_i = \begin{cases} (\tau_i^{max} - b_{2i})/b_{1i}, & b_{1i} > 0 \\ (\tau_i^{min} - b_{2i})/b_{1i}, & b_{1i} < 0 \\ \infty, \quad b_{1i} = 0 \end{cases} \tag{3.4}$$

and

$$d_i = \begin{cases} (\tau_i^{min} - b_{2i})/b_{1i}, & b_{1i} > 0 \\ (\tau_i^{max} - b_{2i})/b_{1i}, & b_{1i} < 0 \\ -\infty, \quad b_{1i} = 0 \end{cases} \tag{3.5}$$

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

$$\ddot{s}_{max} = \min_i c_i, \quad \ddot{s}_{min} = \max_i d_i \tag{3.6}$$

Observe that these bounds depend on the measured quantities q and $\dot{q}$. Note also the notation used, where the quantities $\ddot{s}_{min}$ and $\ddot{s}_{max}$ represent the result of the computation (3.3)-(3.6), i.e. it is not guaranteed that $\ddot{s}_{min} \leq \ddot{s}_{max}$.

3.2 On-line Trajectory Scaling

Algorithms for time scaling of a nominal reference trajectory will now be derived. The nominal trajectory is specified in advance, e.g. precalculated by model based trajectory planning as described in Chapter 2, or specified by a robot operator/programmer. Let the nominal trajectory be given by the nominal path parameter $s_n(t)$ so that the nominal reference trajectory is $q_r(t) = f(s_n(t))$. The scaling is done by on-line modification of the scalar function $s_n(t)$ to a new function $s(t)$. The modified reference trajectory $q_r(t) = f(s(t))$ is then used as input to the primary controller. The first algorithm for trajectory time scaling

uses bounds on the path acceleration $\ddot{s}(t)$ together with feedback from the nominal path speed, and the second algorithm is a modified version where the s-$\dot{s}$-chart of the nominal trajectory is utilized. The primary controller is parametrized in the path parameter s as in the previous section, $\tau = b_1\ddot{s} + b_2$. In order to avoid computing derivatives of measured quantities, the modification of $s(t)$ is actually done by modifying the path acceleration $\ddot{s}(t)$. The scaling algorithm is a secondary control loop outside the ordinary controller, where the output of the secondary controller is the scaled trajectory, represented by $s(t)$, $\dot{s}(t)$, and $\ddot{s}(t)$.

Nominal Trajectory Execution

The nominal path parameter is given by

$$\begin{aligned} &s_n(t), \quad \dot{s}_n(t), \quad \ddot{s}_n(t) \\ &s_n(0) = s_{n_0}, \quad \dot{s}_n(0) = \dot{s}_{n_0}, \quad \ddot{s}_n(0) = \ddot{s}_{n_0} \end{aligned}$$

If $\dot{s}_n(t) > 0$ for all t, the nominal path velocity $\dot{s}_n(t)$ can be regarded as a function of $s_n(t)$. The nominal trajectory can then be represented as a velocity profile

$$\dot{s}_n(t) = v_n(s_n(t)) \tag{3.7}$$

The nominal path acceleration $\ddot{s}_n(t)$ is then given by

$$\ddot{s}_n(t) = \frac{d}{dt}v_n(s_n(t)) = \frac{dv_n(s_n(t))}{ds_n}\dot{s}_n(t) = \frac{dv_n(s_n(t))}{ds_n}v_n(s_n(t)) \tag{3.8}$$

Introduce the function a_n by

$$a_n(s_n) = \frac{dv_n(s_n)}{ds_n}v_n(s_n) = \frac{d}{ds_n}(\frac{1}{2}v_n(s_n)^2) \tag{3.9}$$

The functions v_n and a_n are computed from the nominal trajectory $s_n(t)$, $\dot{s}_n(t)$, and $\ddot{s}_n(t)$, by equations (3.7) and (3.9). They can thus be precomputed and stored in advance. Further, they are not explicit functions of time, and therefore suitable for use in trajectory time scaling. A function of the nominal trajectory that is an explicit function of time would lead to difficulties, since a time scaling of the nominal trajectory may result in a path traversal time that is larger than the nominal traversal time,

and the nominal trajectory is typically defined only for times t smaller than the nominal traversal time.

Using equations (3.8) and (3.9), we see that the nominal trajectory is now obtained from the differential equation

$$\begin{aligned} \ddot{s}(t) &= a_n(s(t)) \\ \dot{s}(0) &= \dot{s}_{n_0} \\ s(0) &= s_{n_0} \end{aligned} \tag{3.10}$$

We have introduced the nominal velocity profile v_n and the nominal acceleration profile a_n. These are given functions, computed from the nominal trajectory $s_n(t)$. We introduce in the same way for the actual trajectory $s(t)$, the actual velocity and acceleration profiles, v and a, as

$$\dot{s}(t) = v(s(t)), \quad \ddot{s}(t) = a(s(t))$$

Time Scale Transformation

The interpretation of the variable s as a transformed time variable will be used to give insight and ideas for devising a secondary control loop for on-line trajectory time scaling. A differential equation for s is derived and used in the next subsection as the basis for introducing measured quantities to form a secondary feedback loop.

Using (3.8) with s_n replaced by s, we get

$$\ddot{s}(t) = \frac{d}{dt}\dot{s}(t) = \frac{d}{dt}v(s(t)) = \frac{dv(s(t))}{ds}v(s(t)) = \frac{d}{ds}(\frac{1}{2}v(s(t))^2) \tag{3.11}$$

Introduce y and y_r by

$$y(s) = \frac{1}{2}v(s)^2, \quad y_r(s) = \frac{1}{2}v_n(s)^2 \tag{3.12}$$

From (3.9), we get

$$a_n(s) = \frac{d}{ds}(\frac{1}{2}v_n(s)^2) = \frac{dy_r(s)}{ds} \tag{3.13}$$

and from (3.11)

$$\ddot{s} = \frac{dy(s)}{ds} \tag{3.14}$$

Equation (3.10) can now be written as a linear differential equation

$$\begin{aligned} \frac{dy(s)}{ds} &= \frac{dy_r(s)}{ds} \\ y(s_0) &= \frac{1}{2}\dot{s}_{n_0}^2 \end{aligned} \tag{3.15}$$

where $s_0 = s_{n_0}$. The equation (3.15) can be combined with feedback from the nominal and actual path velocities, for example to handle numerical errors in the integration or, as is shown in the first simulation example in Section 3.3, to cover the case where the nominal path acceleration is not specified. A linear feedback in the transformed time scale gives

$$\frac{dy(s)}{ds} = \frac{dy_r(s)}{ds} + \alpha(y_r(s) - y(s)) \quad \Rightarrow$$

$$(\frac{d}{ds} + \alpha)(y_r(s) - y(s)) = 0 \tag{3.16}$$

and we see that $y(s) \rightarrow y_r(s)$, i.e. $v(s)^2 \rightarrow v_n(s)^2$ with a time constant $\frac{1}{\alpha}$ in the transformed time scale. When the nominal path acceleration a_n is not specified, we get

$$\frac{dy(s)}{ds} = \alpha(y_r(s) - y(s)) \quad \Rightarrow$$

$$(\frac{d}{ds} + \alpha)y(s) = \alpha y_r(s) \tag{3.17}$$

which gives $y(s)$ as a low-pass filtered version of $y_r(s)$. The time constant of the filter, again in the transformed time scale is $\frac{1}{\alpha}$. The two alternatives, (3.16) and (3.17), are summarized in

$$\begin{aligned} \frac{dy(s)}{ds} &= \beta\frac{dy_r(s)}{ds} + \alpha(y_r(s) - y(s)) \\ y(s_0) &= \frac{1}{2}\dot{s}_{n_0}^2 \end{aligned} \tag{3.18}$$

where $\beta = 1$ or $\beta = 0$. The equation (3.18) can be translated back to the untransformed time scale as

$$\begin{aligned} \ddot{s}(t) &= \beta a_n(s(t)) + \frac{1}{2}\alpha(v_n(s(t))^2 - \dot{s}^2(t)) \\ \dot{s}(0) &= \dot{s}_{n_0} \\ s(0) &= s_{n_0} \end{aligned} \tag{3.19}$$

Note that equation (3.19) is a state equation to generate either the nominal trajectory ($\beta = 1$) or a low-pass filtered version of the nominal trajectory ($\beta = 0$). This equation will now be extended to a secondary feedback controller by including bounds on the path acceleration $\ddot{s}$, where the bounds depend on measured quantities.

Path tracking with bounded path acceleration: Algorithm 1

The torque limits will now be accounted for. A first algorithm is easily obtained by extending the method for nominal trajectory execution, equation (3.19), with on-line bounds on $\ddot{s}$

$$\begin{aligned} a_r &= \beta a_n(s) + \frac{1}{2}\alpha(v_n^2(s) - \dot{s}^2) \\ \ddot{s} &= sat(a_r, \ddot{s}_{min}, \ddot{s}_{max}) \end{aligned} \tag{3.20}$$

where the saturation function sat is defined as the limitation of $\ddot{s}$ by the on-line bounds $\ddot{s}_{min}$ and $\ddot{s}_{max}$ given by (3.6). Note that the bounds depend on s, $\dot{s}$, and the measured quantities q and $\dot{q}$. If the bounds are in conflict, i.e. if the computational procedure used to compute the bounds, equations (3.3)-(3.6), gives the result $\ddot{s}_{min} > \ddot{s}_{max}$, then $\ddot{s} = a_r$ is used. The feedback signal $v_n^2(s) - \dot{s}^2$ is here used to achieve the nominal path speed $v_n(s)$. As long as the $\ddot{s}$-limits are not active, the gain α has the same interpretation as in the case of nominal trajectory execution.

This algorithm works well as long as $\dot{s}$ is on-line admissible, i.e. as long as the on-line bounds on $\ddot{s}$ are computable. The effect is that the saturation function limits the slope of the velocity profile $v_n(s)$. The slope may be too large due to several reasons. A robot programmer may have specified a too demanding velocity profile, or, if a dynamic model is used for the trajectory planning, modeling errors may result in unrealizable path accelerations, e.g. the inertia may be underestimated in some robot configuration. Further analysis of Algorithm 1 is given in Section 4.1, and in Appendix B, where a modified version of the algorithm is shown stable, meaning here that if the nominal path velocity $\dot{s}_n$ is bounded, the actual path velocity $\dot{s}$ is also bounded. Algorithm 1 is illustrated in Figure 3.1, where a block diagram of the system with the secondary control loop is shown. A simulation example using Algorithm 1 is given in Section 3.3 and shown in Figure 3.2.

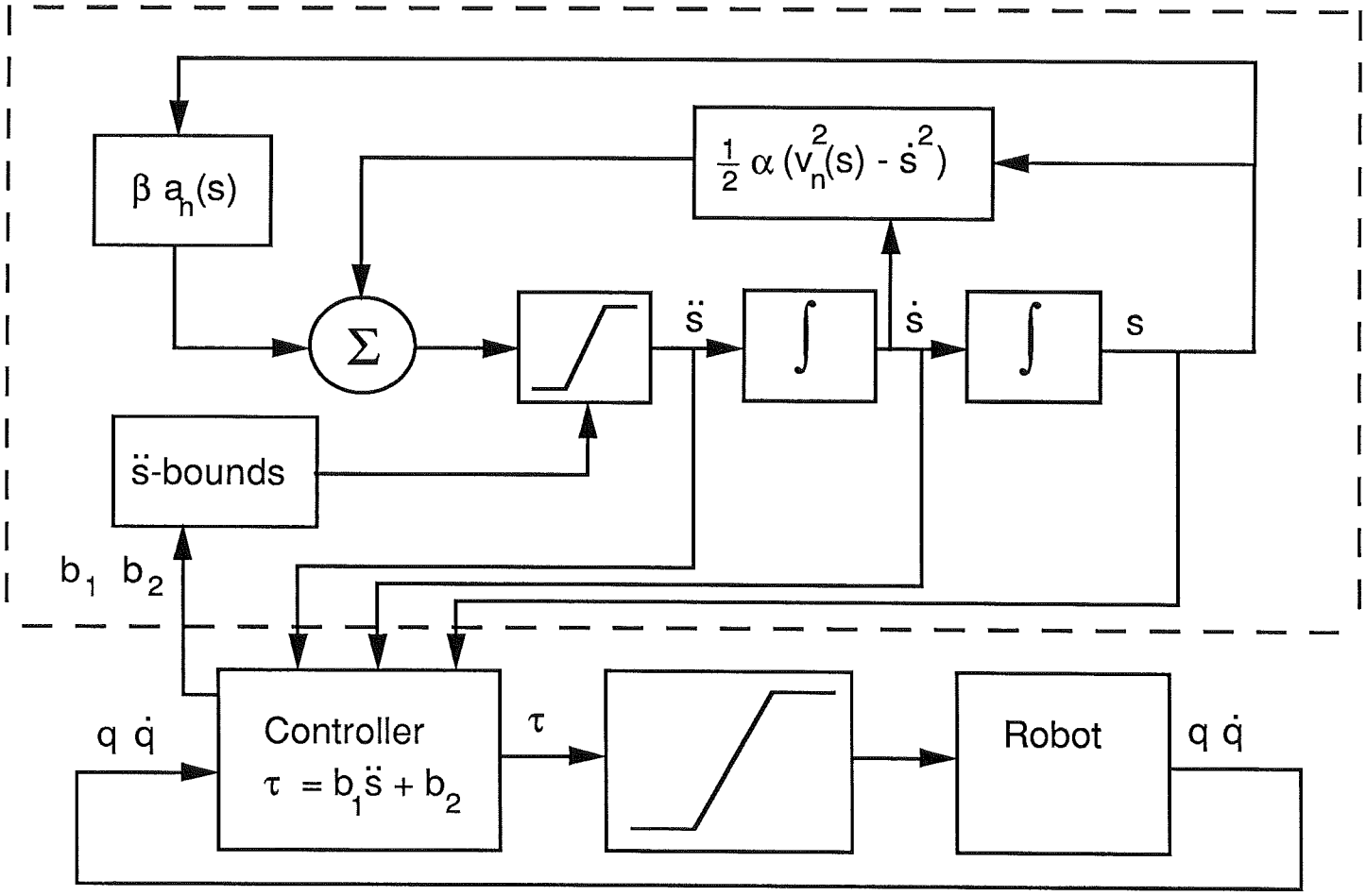

Figure 3.1 A block diagram of the robot with primary and secondary control loop. The primary controller is parametrized in s. Observe that b_1 and b_2 depend on measured quantities.

Including velocity profile constraints: Algorithm 2

It is not sufficient to consider only the path acceleration constraints when executing minimum time trajectories. Also the constraints on the velocity profile has to be taken into account. Already the nominal path velocity may, at some points along the path, be at the boundary of the nominal admissible region, see Figure 3.3. Model uncertainties may then result in $\dot{s}$ not being on-line admissible during the complete motion. This means that at some point along the path, there exists no $\ddot{s}$ leading to admissible torques, and the motion cannot be continued without moving away from the path.

Algorithm 1 is therefore extended with a modification of the nominal velocity profile in order to handle this problem. The nominal path speed may be too large, e.g. due to modeling errors, and the idea of the velocity

profile modification is to have a reduction of the path speed, resulting in a modified velocity profile, lower than the nominal. We choose to implement the modification as a scaling of the nominal velocity profile, i.e. to use $\gamma v_n(s)$ instead of $v_n(s)$ as the nominal path velocity, where γ is a scaling factor. The scaling factor γ is then updated as the robot moves. A scaling of the nominal velocity profile also influences the nominal path acceleration $a_n(s)$. The relation between the nominal path acceleration and the velocity profile is given by (3.9). Assuming small modifications of γ, so that γ can be regarded as constant, the path acceleration $a_n(s)$ is replaced by $\gamma^2 a_n(s)$. This results in the algorithm

$$\begin{aligned} a_r &= \beta\gamma^2 a_n(s) + \frac{1}{2}\alpha(\gamma^2 v_n^2(s) - \dot{s}^2) \\ \ddot{s} &= sat(a_r, \ddot{s}_{min}, \ddot{s}_{max}) \end{aligned} \tag{3.21}$$

where the saturation function, the parameters α and β have the same meaning as in Algorithm 1, equation (3.20).

The design of an update law for the modification of γ in (3.21) is here based on the assumption that the nominal trajectory is a minimum time trajectory. Recall that a minimum time velocity profile consists of intervals with nominally either maximum or minimum path acceleration, where an acceleration interval is always followed by a deceleration interval. The nominal acceleration may be too high during an acceleration interval, e.g. due to modeling errors. If we regard this as an indication that the nominal path velocity may be too high further along the path, a reduction of the scaling factor during acceleration is a way to ensure that the path speed will be admissible also during the rest of the motion.

Algorithm 2 is now obtained by combining (3.21) with an update law for γ. The update law for γ proposed here uses the quotient between the nominal velocity profile $v_n(s)$ and the actual path speed $\dot{s}$ as input. Suppose that the nominal acceleration is too high during an acceleration interval, and that this results in torques at the limit during the acceleration. The on-line bounds on path acceleration, (3.21), are then activated, resulting in path following with limited torques during the acceleration. Since the limited torques are a result of a too high nominal acceleration, the actual velocity profile will be below the nominal velocity profile. The quotient between the nominal velocity profile $v_n(s)$, and the actual path velocity $\dot{s}$, is then used to adjust the scaling factor γ. In the simulations and experiments, in Sections 3.3 and 3.4, this is done by the following

update law

$$\begin{aligned}\gamma &= 1 + kx_f \\ \dot{x}_f &= \begin{cases} \dot{s}(-ax_f + 1 - \gamma v_n(s)/\dot{s}), & \gamma v_n(s)/\dot{s} \geq 1 \\ \dot{s}(-ax_f), & \gamma v_n(s)/\dot{s} < 1 \end{cases}\end{aligned} \tag{3.22}$$

where x_f is a scalar filter variable, and k is a scalar constant. Other update laws could be considered, and one alternative is given in Section 4.2, equation (4.14).

Tuning rules for the parameters k and a can be derived from the following approximate analysis, where the differential equation (3.22) for $\dot{x}_f$ is interpreted as a filter in the transformed time scale s. The equation for $\dot{x}_f$ is written in the transformed time scale as

$$\frac{d}{ds}x_f = \begin{cases} -(a + kv_n(s)/\dot{s})x_f + 1 - v_n(s)/\dot{s}, & \gamma v_n(s)/\dot{s} \geq 1 \\ -ax_f, & \gamma v_n(s)/\dot{s} < 1 \end{cases} \tag{3.23}$$

Assuming an actual path velocity $\dot{s}$ approximately equal to the nominal velocity profile $v_n(s)$, i.e. $\dot{s}/v_n(s) \approx \tilde{\gamma} \approx 1$, and choosing the parameters k and a such that $k \gg a$, we see that the upper equation in (3.23) represents a time constant of $1/k$ in the transformed time scale. In the case of a too high nominal path acceleration during an acceleration interval, the upper alternative in (3.23) is selected, and the parameter k can be used to determine the time constant for the update of γ during the acceleration. If the nominal trajectory does not result in the activation of the $\ddot{s}$-bounds, we get $\dot{s} \approx \gamma v_n(s)$, resulting in $x_f \rightarrow 0$, i.e. $\gamma \rightarrow 1$, with the time constant $1/a$. Note that the transformed time scale is useful for tuning of the parameters k and a, and that the s-$\dot{s}$ diagram for the nominal trajectory can be used to determine suitable parameter values.

3.3 Simulations

The proposed methods have been simulated for different robots, but the presentation will here be restricted to a decoupled two-link robot. The model used is describing the robot used in the experiments, presented in Section 3.4. The simulations have been performed using Simnon (Elmqvist, Åström , and Schönthal, 1986). The robot model has

been determined by physical modeling and experiments. The dynamic equations are

$$m_i\ddot{x}_i + d_i\dot{x}_i = \tau_i, \quad i = 1, 2 \tag{3.24}$$

Assuming x_i and τ_i are measured in Volts, the parameters of the model are $m_1 = m_2 = 0.050$, $d_1 = d_2 = 0.0048$. The torques are constrained by constant bounds $\mid \tau_1 \mid \leq 0.2$, $\mid \tau_2 \mid \leq 0.2$, and the path is $f_1(s) = 0.4(1 - \cos(s))$, $f_2(s) = 0.8\sin(s)$, $0 \leq s \leq 2\pi$.

First, a simple trajectory is executed by Algorithm 1. The nominal trajectory used is an example of an unfollowable reference trajectory, e.g. the robot operator/programmer may have specified a too demanding velocity profile. It is from an application point of view important that a control scheme is robust to such nominally unfollowable reference trajectories.

The nominal trajectory is represented as a velocity profile

$$v_n(s) = \begin{cases} v_{n_1}, & s_0 \leq s \leq s_1 \\ v_{n_2}, & s_1 \leq s \leq s_2 \\ v_{n_2} - \frac{v_{n_2}}{s_3 - s_2}(s - s_2), & s_2 \leq s \leq s_3 \end{cases}$$

where $v_{n_1} = 2.1$, $v_{n_2} = 0.7$, $s_1 = 3$, $s_2 = 5$, and $s_3 = 2\pi$. The velocity profile is piecewise constant between s_0 and s_2, i.e. the path acceleration is nominally infinite at the transition points, s_0 and s_1.

The controller is a computed torque controller

$$\tau_i = m_i(\ddot{x}_{r_i} + k_{v_i}\dot{e}_i + k_{p_i}e_i) + d_i\dot{x}_i, \quad i = 1, 2$$

where x_{r_i} denotes the reference trajectory, and $e_i = x_{r_i} - x_i$ is the tracking error. The use of a secondary controller for trajectory time scaling should not influence the tuning of the primary controller. Here, the feedback gains were chosen to give poles with critical damping, and a natural frequency of $9rad/s$, i.e. $k_{p_1} = k_{p_2} = 81$, $k_{v_1} = k_{v_2} = 18$.

The result of using Algorithm 1, equation (3.20), with $\beta = 0$ and $\alpha = 30$, is shown in Figure 3.2. The time constant $1/\alpha$ in the transformed time scale is here $1/30$, and should be compared with the length of the path. The upper left plot shows the path tracking, which is excellent. The mid left plot shows the nominal and the actual velocity profiles. The slope of the velocity profile has been limited, resulting in a followable reference trajectory. The difference in nominal and actual path traversal

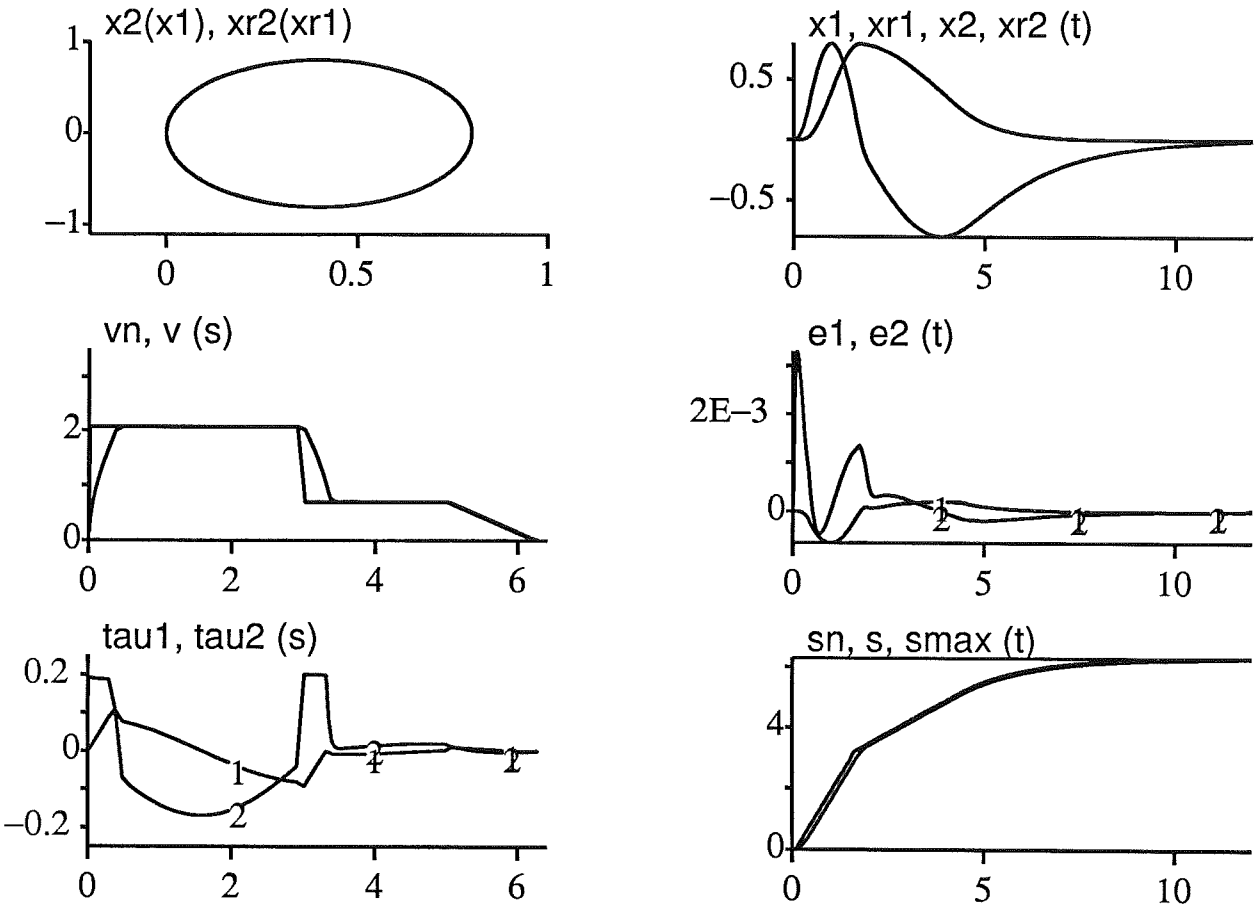

Figure 3.2 The result of using Algorithm 1 on a nominally unfollowable reference trajectory. The path acceleration is limited by the algorithm, resulting in path following with limited torques.

times is small, as can be seen in the lower right plot where s and s_n are shown. The end point of the path, here denoted s_{max} is also shown in the lower right plot. Note that one of the torques, shown in the lower left plot, is at the limit during the velocity transitions.

Algorithm 2 will now be used to handle a minimum time trajectory. The trajectory planning is based on a robot model with parametric errors. The nominal trajectory leads to large path deviations, and it will be shown how path following can be obtained by a small modification of the nominal trajectory.

The nominal trajectory was chosen as a minimum time trajectory for the nominal model, equation (3.24). The minimum time trajectory planning is based on (Shin and McKay, 1985) and implemented in MATLAB (Moler, Little, and Bangert, 1987). The implementation is further described in Appendix A. The nominal trajectory and the corresponding torques, computed from equation (3.24), are shown in Figure 3.3. The

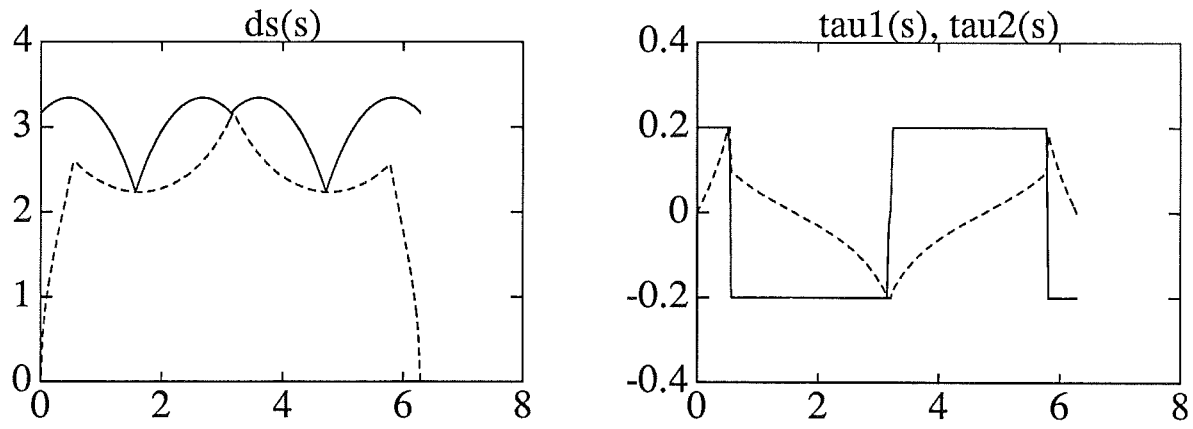

Figure 3.3 An s-$\dot{s}$-diagram is shown in the left plot. The nominal minimum time trajectory (dashed) is represented as a velocity profile $\dot{s}(s)$. The solid curve is the boundary of the admissible region. The right plot shows the nominal minimum time torques as functions of the path parameter.

nominal trajectory was executed by (3.19) with $\beta = 1$ and $\alpha = 20$, using a robot model with the parameters m_1 and m_2 5 % larger than the nominal values. The primary controller was the same as in the previous example. The result is shown in Figure 3.4. The nominal trajectory leads to large path deviations, as can be seen in the upper left plot. Since the actual robot model is heavier than the robot model used in the trajectory planning, this is expected. The mid left plot shows that the actual velocity profile is the same as the nominal velocity profile. The nominal velocity profile is also found in the left plot in Figure 3.3. The lower left plot shows that one of the torques is at the limits during the complete motion. The tracking performance can be seen in the upper right and the mid right plots. The reference trajectory of joint 2 cannot be followed, a result of the limited torque 2.

Using Algorithm 2 with $\beta = 1$, $\alpha = 20$, $k = 4$, and $a = 0.05$ results in accurate path following. The parameters were chosen as follows. The nominal path acceleration is used in the trajectory execution, which gives $\beta = 1$. The parameter α determines the time constant for $\dot{s}^2 = v(s)^2$ when the bounds on $\ddot{s}$ are not active. The time constant is here $1/\alpha = 0.05$. The parameter k gives the time constant for the modification of γ, and is here chosen as $k = 4$, which gives a time constant of $1/k = 0.25$,

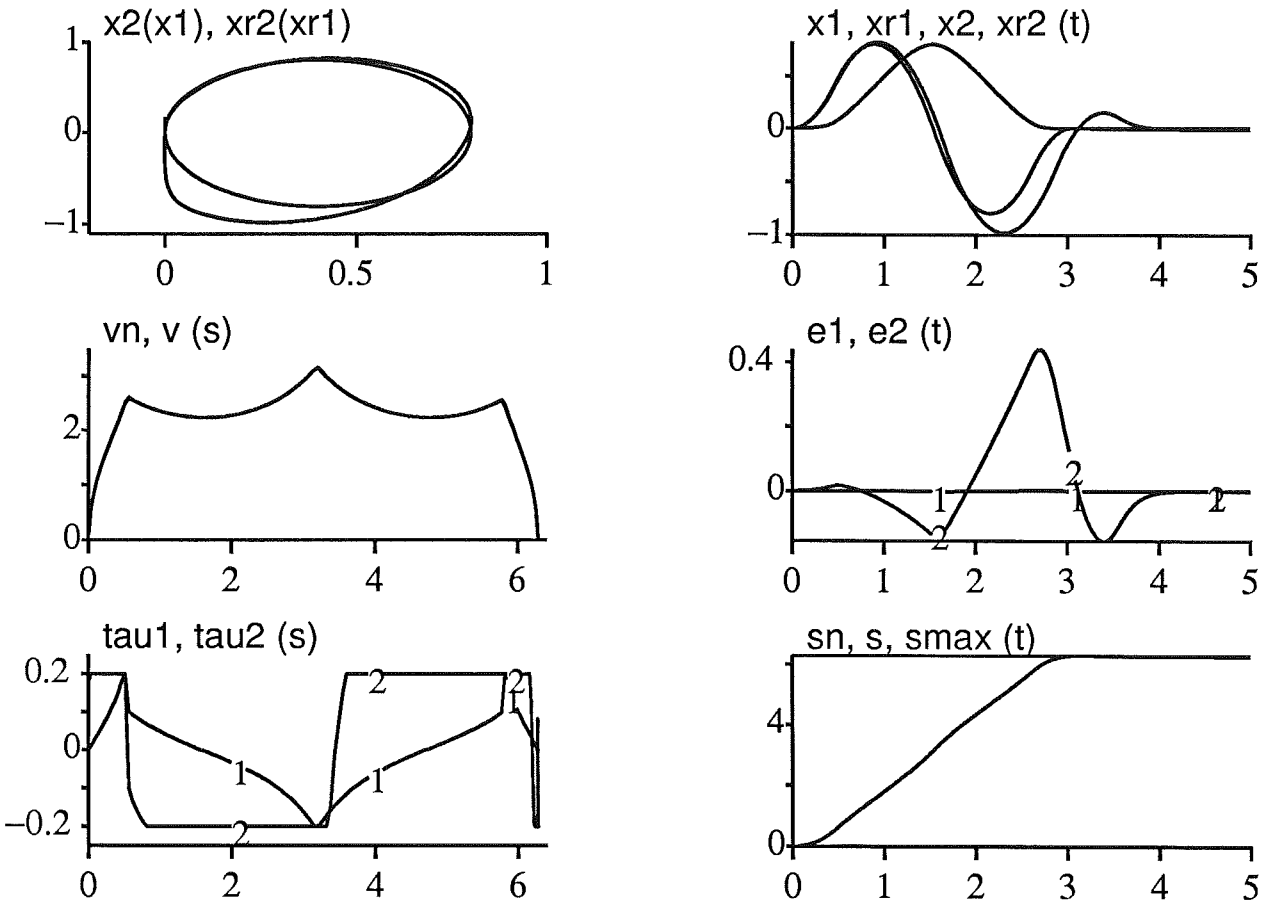

Figure 3.4 An example where nominal trajectory execution is unsuccessful due to an erroneous model. The actual parameters m_1 and m_2 are 5 % larger than the nominal parameters.

which is approximately equal to half the first acceleration interval, see Figure 3.3. The purpose is to make the current nominal velocity profile $\gamma v_n(s)$ converge to the actual velocity profile $\dot{s} = v(s)$ during the first acceleration interval. The parameter a is here chosen as $a = 0.05$, which gives the time constant for resetting γ to 1, $1/a = 20$.

The result of using Algorithm 2 on the nominal minimum time trajectory is shown in Figure 3.5. The evaluation of the result is based on path following and torque utilization. The path following is good, as can be seen in the upper left plot. The torques are also close to their limits, see the lower left plot. The nominal and the modified velocity profile are shown in the mid left plot, and the tracking errors are shown in the mid right plot. Note the sensitivity of the path following, upper left plot in Figure 3.4 and 3.5, with respect to the traversal time. Large path following errors have been eliminated by a small increase in traversal time.

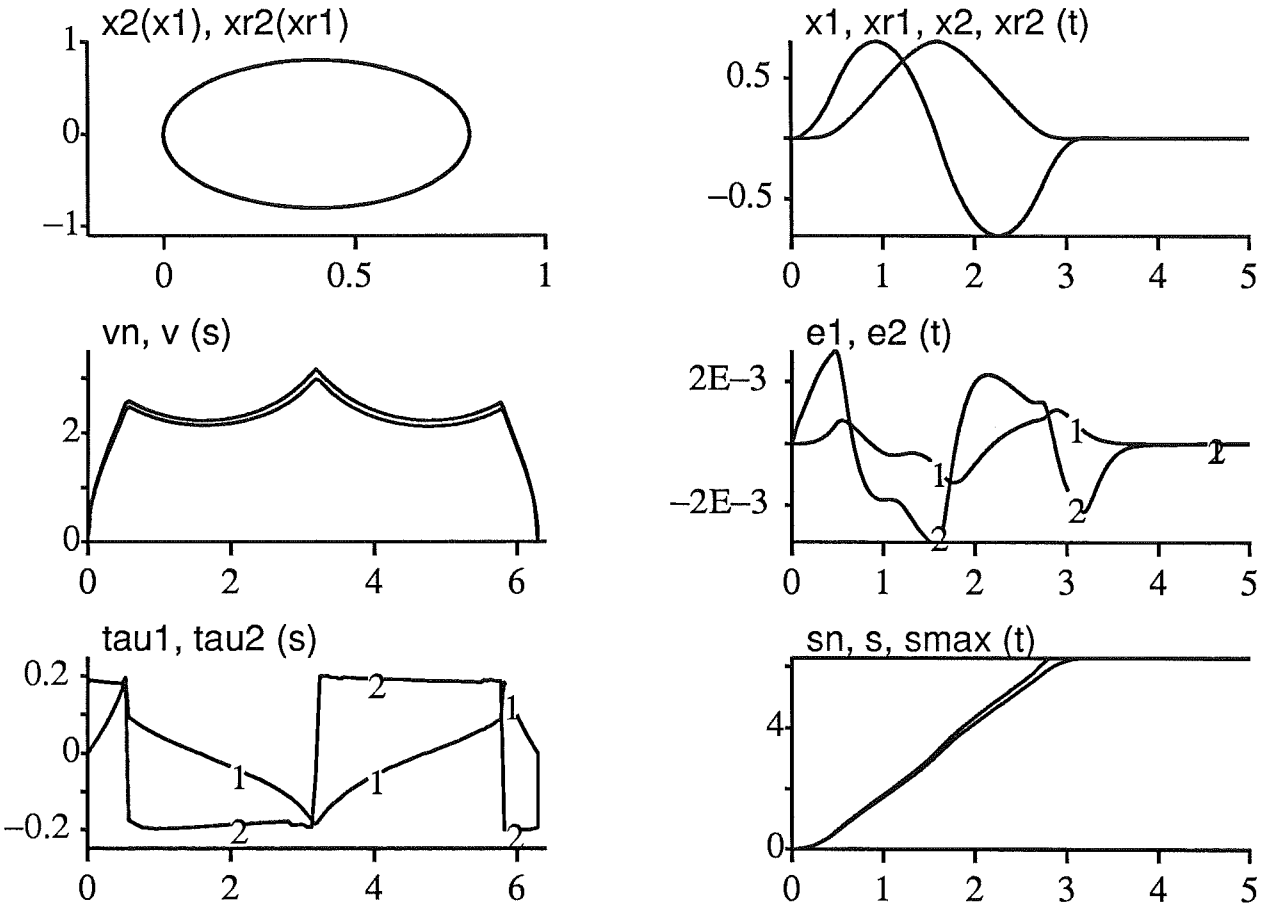

Figure 3.5 Using Algorithm 2 on the nominal minimum time trajectory, shown in Figure 3.3, leads to tracking. The increase in path traversal time is shown in the lower right diagram.

3.4 Experiments

The experiments were performed using a laboratory system controlled by an IBM-AT computer. The robot consists of two decoupled DC servos and is described by the model (3.24). The DC servos are further described in e.g. (Jönsson, 1988). The software used in the experiments is written in Modula-2, using library modules from Logitech (Logitech, 1987), and from the Department of Automatic Control.

Implementation Environment

A compiler for translation of Simnon systems into Modula-2 code was used to simplify the implementation. The compiler is an extended version of the compiler described in (Dahl, 1989). A discrete time Simnon system is translated into a Modula-2 module, and procedures for accessing parameters and other variables are generated. The primary controller

and the scaling algorithm was implemented in one IBM-AT computer, using a foreground/background scheduler (Andersson, 1989). Plotting facilities and data logging was implemented in a second computer, using a real time kernel. The data transfer between the computers was done using the AD/DA converters. The Simnon/Modula-2 compiler also allows a mixture of Simnon code and Modula-2 code as input. A first implementation of the algorithm was done in Simnon, and the functionality was verified by simulation. The nominal trajectory was chosen as a simple velocity profile. Modula-2 code was then added to cover the minimum time case, where the nominal trajectory, computed in MATLAB, was stored in a file.

Experimental Results

Experimental results obtained by using Algorithm 2 on the same minimum time trajectory as in the previous section will be presented. The path and the primary controller were the same as in the simulation, and the sampling time was 0.017 seconds. The feedback gains in the primary controller were empirically chosen as $k_{p_1} = 225$, $k_{p_2} = 100$, $k_{v_1} = 21$, $k_{v_2} = 14$. The nominal trajectory is shown in Figure 3.3. Algorithm 2 was used, however discretized by a forward Euler approximation. Simulations of the discrete algorithm led to the parameter choice $\alpha = 5$ and $\beta = 1$. The experimental results on the real system are shown in Figures 3.6-3.9. The x-axis is scaled in seconds. The sampling time for the data logging was 0.07 seconds.

Figure 3.6 shows an artificial experiment where the torque bounds have been removed. The purpose is to show that the primary controller is properly tuned, i.e. the motion along the path is not constrained by the performance of the primary controller. We see in the upper left plot that this is the case. The path following is good, and the tracking errors are small, see the lower left plot. The torques required to track the reference trajectory are, due to imperfect modeling, larger than the nominal, compare the limiting joint 2 in Figure 3.3 and in Figure 3.6, the lower right side plot.

In Figure 3.7, the torques are constrained by the bounds used in the trajectory planning, $| \tau_1 | \leq 0.2$, $| \tau_2 | \leq 0.2$. The modeling errors lead to very poor tracking, caused by the limited torques. Figure 3.8 shows the result of using Algorithm 2 with $k = 1.5$, and $a = 0.1$. The evaluation of the result is based on path following and torque utilization. The nominal trajectory is slowed down, and the modified trajectory can be followed,

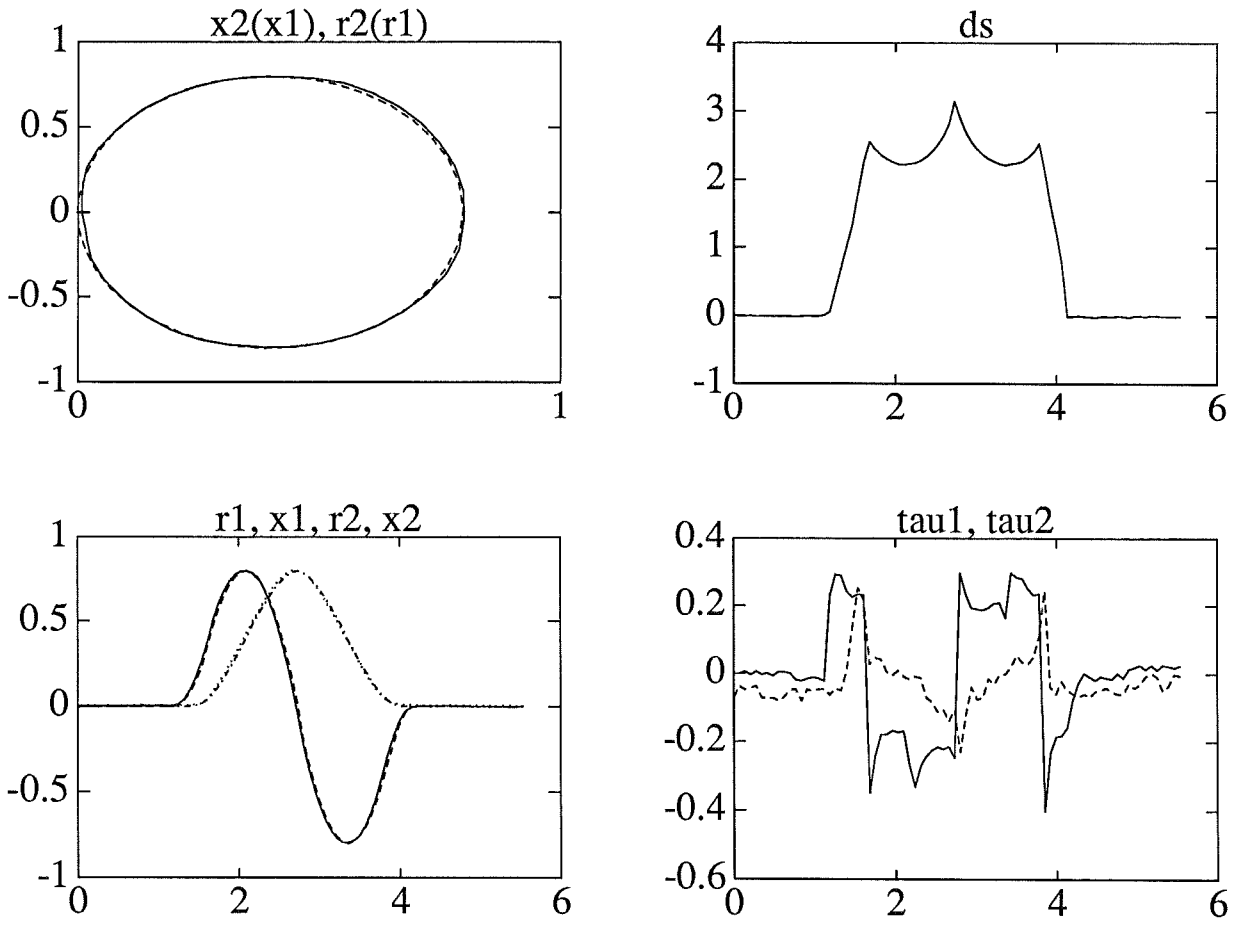

Figure 3.6 An artificial experiment where the torque limits have been removed. The purpose is to show that the primary controller is properly tuned.

resulting in good path following, see the upper left plot. The capability of the robot is utilized as can be seen from the torques of joint 2 which are close to the bound 0.2. Note that in the simulation in Section 3.3, the model used in the trajectory planning was deliberately modified by 5 % in order to demonstrate the properties of the algorithm. This is not the case here. The model used is derived from physical modeling and identification experiments. The model accuracy is good enough for controller design, see Figure 3.6, but not for minimum time trajectory planning, see Figure 3.7. A crude measure of the model uncertainty could be the increase in traversal time, see Figure 3.9, where the nominal velocity $\dot{s}_n$ and the actual velocity $\dot{s}$ are shown as functions of time.

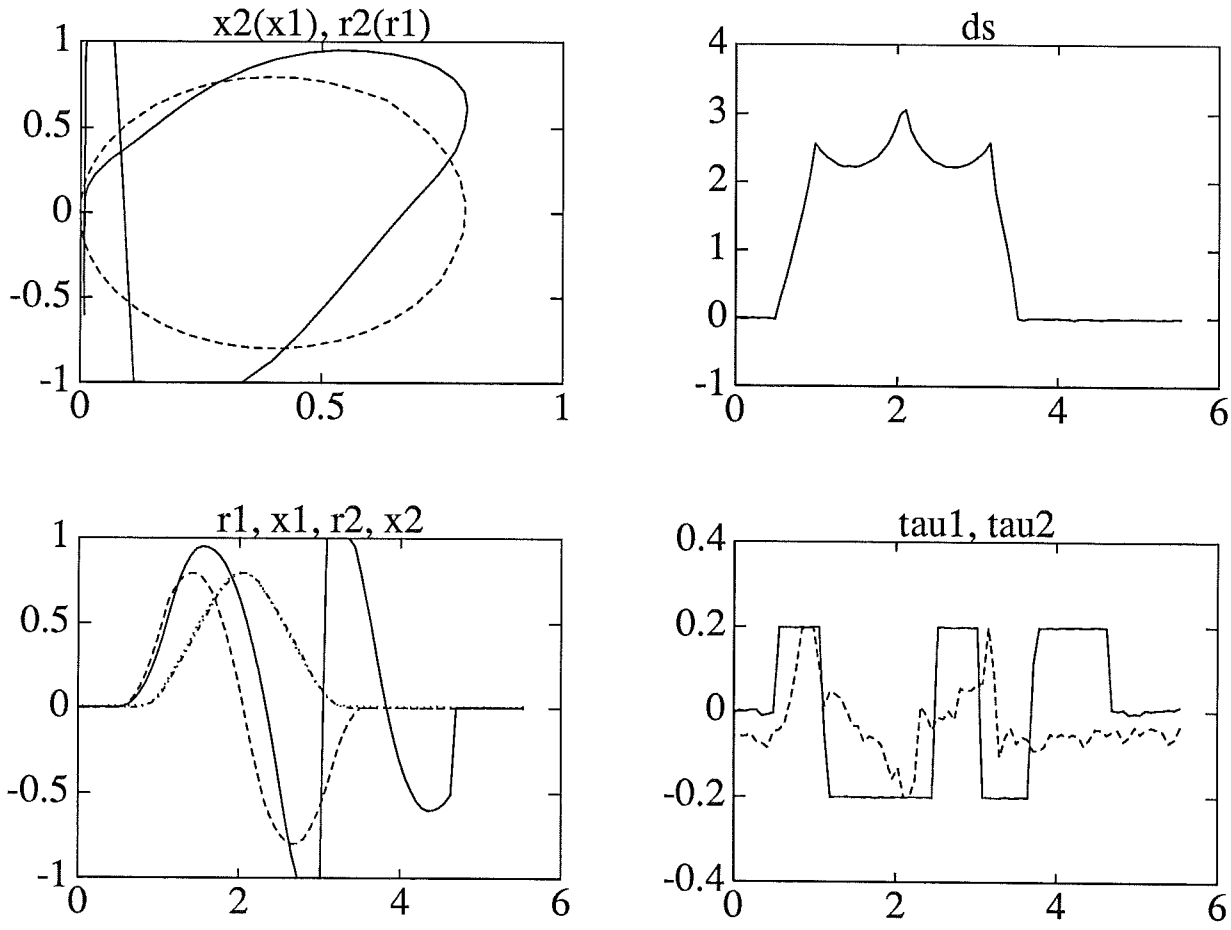

Figure 3.7 Execution of the nominal trajectory when the torques are bounded leads to large path deviations.

3.5 Summary

A feedback scheme for trajectory time scaling has been proposed. A secondary controller is used for the modification of a nominal trajectory during motion. The primary controller is parametrized in the path coordinate s, but otherwise not changed. A scalar quantity, the path acceleration $\ddot{s}$ is modified, resulting in coordinated adjustment of the individual joint motions. Two algorithms have been designed, and verified by simulations and experiments. One algorithm is based on bounds on the path acceleration, together with feedback from the nominal path velocity. The algorithm limits the slope of the velocity profile, and can be used for simplification of trajectory programming, in the sense that a too demanding velocity profile, for example specified by an operator/programmer, is ad-

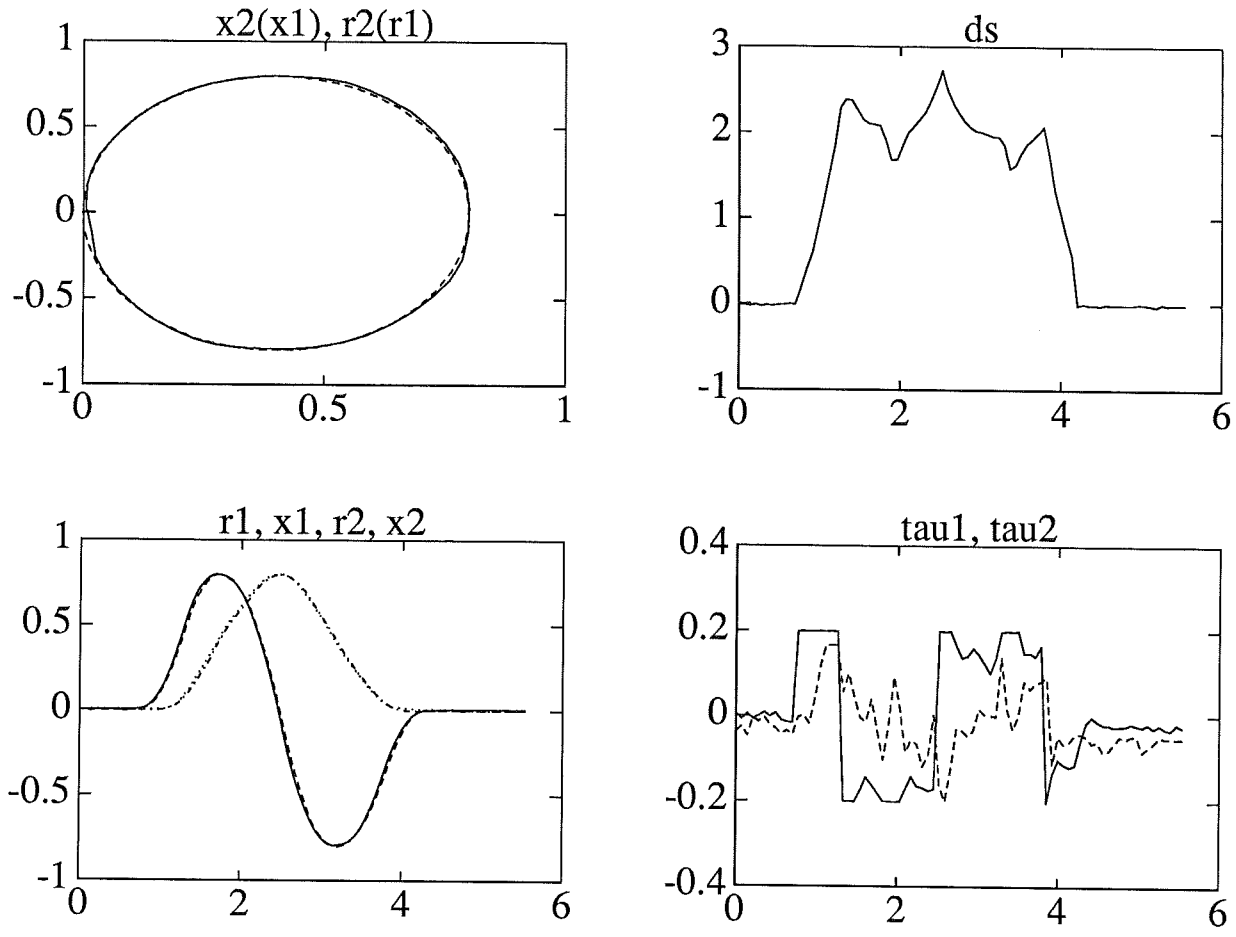

Figure 3.8 The result of using a secondary feedback loop, based on Algorithm 2, on the nominal minimum time trajectory. The modified trajectory leads to path following with limited torques.

justed to meet the requirements given by the torque constraints. The second algorithm is an extension of the first algorithm, and designed to handle nominal minimum time trajectories. A scaling of the nominal velocity profile is introduced, and the scaling factor is modified on-line. The nominal minimum time trajectory is typically too fast, and the idea is to have a reduction of the path speed during acceleration, resulting in admissible torques also during the following deceleration. The speed reduction should only occur if needed, and then be as small as possible.

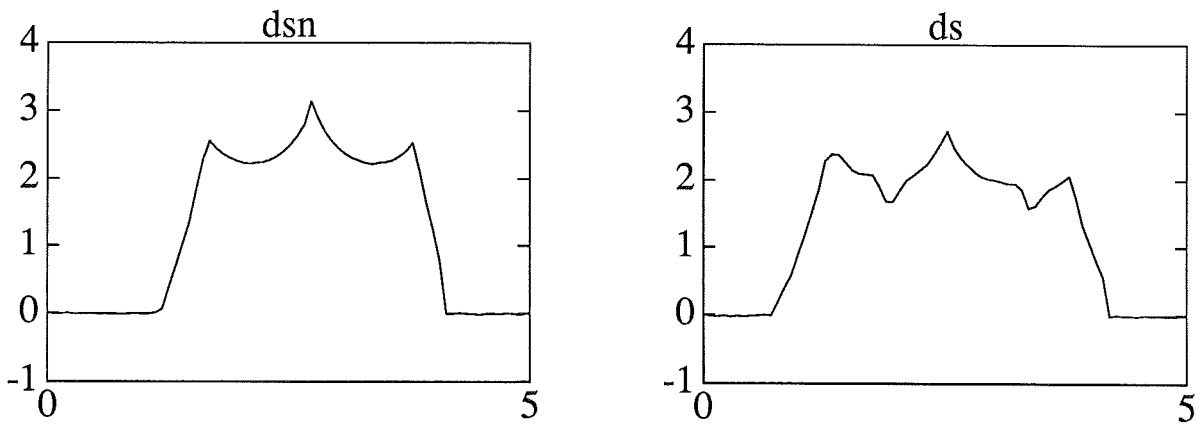

Figure 3.9 The nominal (left) and the modified velocity

4. Further Analysis

This chapter presents, together with Appendix B, further analysis of the proposed algorithms for trajectory time scaling. The design of the secondary controller is further motivated by a stability analysis for a restricted case of Algorithm 1, where the on-line bounds on path acceleration are assumed to have different signs, i.e. the lower bound is never positive, and the upper bound is always nonnegative. The condition that the bounds have different signs is also interpreted as a restriction on the path speed $\dot{s}$. The purpose of the stability analysis is to justify the design of the secondary loop, and not to give detailed information about e.g. tuning of parameters. The main ideas in the analysis are given in Section 4.1, and detailed calculations are given in Appendix B.

Analysis relevant for Algorithm 2 is given in Section 4.2. It is shown how a constant scaling of the velocity profile influences the torques, and requirements on a scaling factor resulting in admissible torques are derived. A discussion of limitations and modifications of Algorithm 2 is also given.

4.1 Restricted Stability Analysis

The design of the secondary loop is illustrated by Figure 3.1, where Algorithm 1 is shown. Algorithm 2 is obtained by including a scaling factor γ such that $v_n(s)$ is replaced by $\gamma v_n(s)$, and $a_n(s)$ is replaced by $\gamma^2 a_n(s)$. If the on-line modification of γ is small, so that γ can be regarded as constant, the block diagram is also relevant for Algorithm 2. Figure 3.1 shows that the closed loop system is nonlinear. The robot dynamics are nonlinear, and the output of the primary controller is limited. The secondary loop is also a nonlinear system, with internal feedback from the squared path velocity $\dot{s}^2$, and limitations on $\ddot{s}$, where the bounds depend on the robot position and speed, q and $\dot{q}$.

The restriction in the stability analysis is a modification of Algorithm 1. The modification is described in this section, and a stability result for the modified algorithm is given in Appendix B. The result is a bounded-input bounded-output type result, and shows that if the nominal velocity

profile $v_n(s)$ is bounded, the actual velocity profile $v(s)$ is also bounded, and the bounds are independent of the tracking errors.

The stability of the closed loop system, i.e. the system given by the robot, the primary controller, and the secondary controller, is also discussed in this section. The discussion is formalized in Appendix B, where it is shown how a stable closed loop system can be obtained by using the modified Algorithm 1, in connection with a primary controller with guaranteed tracking performance. The possibility of achieving a controller with this property is discussed in Appendix B, using a result from (Craig, 1988).

Algorithm 1

Algorithm 1 is obtained from the method for nominal trajectory execution, equations (3.18) and (3.19), by including on-line bounds on the path acceleration $\ddot{s}$. The bounds, denoted $\ddot{s}_{min}$ and $\ddot{s}_{max}$, are computed from the torque constraints and the controller parametrization, see equations (3.3)-(3.6). Note that the bounds exist only if there are numbers, $\ddot{s}_{min}$ and $\ddot{s}_{max}$, such that $\tau = b_1\ddot{s} + b_2$ is inside the torque limits for all $\ddot{s}$, satisfying $\ddot{s}_{min} \leq \ddot{s} \leq \ddot{s}_{max}$. If the bounds do not exist, the computational procedure used, equations (3.3)-(3.6), gives the result $\ddot{s}_{min} > \ddot{s}_{max}$. The controller is parametrized as $\tau = b_1\ddot{s} + b_2$, where b_1 and b_2 are allowed to be functions of s, $\dot{s}$, q, and $\dot{q}$. Since $e = q_r - q$, we can equivalently regard b_1 and b_2 as functions of s, $\dot{s}$, e, and $\dot{e}$. This means that the on-line bounds on $\ddot{s}$ are also functions of the same quantities. Algorithm 1 is now written as

$$\begin{aligned} a_r &= \beta a_n(s) + \frac{1}{2}\alpha(v_n(s)^2 - \dot{s}^2) \\ \ddot{s} &= sat(a_r, s, \dot{s}, e, \dot{e}) \\ \dot{s}(0) &= \dot{s}_{n_0} \\ s(0) &= s_{n_0} \end{aligned} \tag{4.1}$$

where the dependence on the tracking errors is explicitly shown by the function $sat(a_r, s, \dot{s}, e, \dot{e})$, defined as the limitation of a_r by the on-line bounds $\ddot{s}_{min}(s, \dot{s}, e, \dot{e})$ and $\ddot{s}_{max}(s, \dot{s}, e, \dot{e})$ if $\ddot{s}_{min} \leq \ddot{s}_{max}$, and a_r otherwise.

Modification of Algorithm 1

A modified version of Algorithm 1 will be shown stable, meaning here that a bounded nominal velocity profile will result in a bounded actual velocity profile, where the bounds are independent of the tracking errors e and $\dot{e}$. The modification of the algorithm is a modification of the saturation function *sat*, that limits the path acceleration $\ddot{s}$ by the on-line bounds $\ddot{s}_{min}$ and $\ddot{s}_{max}$. In Algorithm 1, the path acceleration is limited as long as the bounds satisfy $\ddot{s}_{max} \geq \ddot{s}_{min}$. The modified version limits the path acceleration only if the bounds $\ddot{s}_{min}$ and $\ddot{s}_{max}$ have different signs, i.e. $\ddot{s}_{max} \geq 0$ and $\ddot{s}_{min} \leq 0$. The condition that the bounds have different signs can be interpreted as a restriction on the magnitude of the path speed $\dot{s}$, see Lemma 4.1 and the following discussion. We therefore define $\dot{s}$ admissible as follows.

DEFINITION 4.1
Given b_1 and b_2 in the controller parametrization $\tau = b_1\ddot{s} + b_2$, the path speed $\dot{s}$ is admissible if the $\dot{s}$-dependent $\ddot{s}$-bounds $\ddot{s}_{min}(s, \dot{s}, e, \dot{e})$ and $\ddot{s}_{max}(s, \dot{s}, e, \dot{e})$ exist, are finite, and satisfy $\ddot{s}_{max} \geq \ddot{s}_{min}$, $\ddot{s}_{max} \geq 0$, and $\ddot{s}_{min} \leq 0$.

Remark 1. The condition that the $\ddot{s}$-bounds exist can also be formulated as there exist numbers, $\ddot{s}_{min}$ and $\ddot{s}_{max}$, such that $\tau = b_1\ddot{s} + b_2$ is inside the torque limits if $\ddot{s}$ satisfies $\ddot{s}_{min} \leq \ddot{s} \leq \ddot{s}_{max}$.

Remark 2. This definition of $\dot{s}$ admissible is only used in this section, and in Appendix B. In the off-line trajectory planning, described in Chapter 2, $\dot{s}$ admissible has another meaning. □

The following Lemma is given to demonstrate that the fact that $\dot{s}$ is admissible can be interpreted as a restriction on $|\ \dot{s}\ |$.

LEMMA 4.1
Suppose that the torque constraints can be expressed as $|\ \tau\ | \leq \tau_{max}$, where $|\ \tau\ |$ denotes the vector norm of τ. A necessary and sufficient condition for $\dot{s}$ admissible is then given by $|\ b_1\ | \neq 0$, and $|\ b_2\ | \leq \tau_{max}$.

Proof: Suppose that $\dot{s}$ is admissible. We then have bounds on $\ddot{s}$ such that τ is admissible for all $\ddot{s}$, satisfying $\ddot{s}_{min} \leq \ddot{s} \leq \ddot{s}_{max}$, where $\ddot{s}_{min} \leq 0$, and $\ddot{s}_{max} \geq 0$. This means that τ is admissible if $\ddot{s} = 0$, i.e. $|\ b_1 0 + b_2\ | \leq \tau_{max}$, which implies $|\ b_2\ | \leq \tau_{max}$. Furthermore, the bounds on $\ddot{s}$ are finite, which implies $|\ b_1\ | \neq 0$, since if $|\ b_1\ | = 0$, then $b_1 = 0$, which results in $|\ \tau\ | = |\ b_2\ | \leq \tau_{max}$ for all $\ddot{s}$, which contradicts the fact that the bounds on $\ddot{s}$ are finite.

Suppose now that $\mid b_2 \mid \leq \tau_{max}$, and $\mid b_1 \mid \neq 0$. We then have

$$\mid b_1 \ddot{s} + b_2 \mid \leq \mid b_1 \mid \mid \ddot{s} \mid + \mid b_2 \mid$$

which gives bounds on $\ddot{s}$ as

$$\ddot{s}_{min} = -\frac{\tau_{max} - \mid b_2 \mid}{\mid b_1 \mid} \leq \ddot{s} \leq \frac{\tau_{max} - \mid b_2 \mid}{\mid b_1 \mid} = \ddot{s}_{max}$$

and we see that the bounds exist, are finite, and satisfy $\ddot{s}_{max} \geq \ddot{s}_{min}$, $\ddot{s}_{max} \geq 0$, and $\ddot{s}_{min} \leq 0$, i.e. $\dot{s}$ is admissible. □

The interpretation of $\dot{s}$ admissible as a restriction on $\mid \dot{s} \mid$ is motivated as follows. Suppose that the controller is a computed torque controller. The controller parametrization then gives

$$\begin{aligned} b_1 &= \hat{H}(q) f'(s) \\ b_2 &= \hat{H}(q)(f''(s)\dot{s}^2 + K_v \dot{e} + K_p e) + \hat{v}(q, \dot{q}) + \hat{d}(q)\dot{q} + \hat{g}(q) \end{aligned} \tag{4.2}$$

Suppose further the tracking errors are small, so that $q \approx q_r$ and $\dot{q} \approx \dot{q}_r$. Since $q_r = f(s)$, the vector $\hat{v}$ can then be approximated as $\hat{v}(q, \dot{q}) \approx \hat{v}(q_r, \dot{q}_r) = v(f, f')\dot{s}^2$, see equations (2.2) and (2.3). This gives

$$\begin{aligned} b_2 &\approx \hat{H}(q_r) f''(s)\dot{s}^2 + \hat{v}(f, f')\dot{s}^2 + \hat{d}(q) f'(s)\dot{s} + \hat{g}(q) \\ &= b_{21}\dot{s}^2 + b_{22}\dot{s} + b_{23} \end{aligned}$$

and we see that the equivalent formulation of $\dot{s}$ admissible as given by Lemma 4.1, $\mid b_2 \mid \leq \tau_{max}$, now gives restrictions on the magnitude of the path speed, $\mid \dot{s} \mid$.

Algorithm 1 is now modified by replacing the saturation function sat, equation (4.1), by the function sat_1, with the same arguments, and defined as the limitation of x by $\ddot{s}_{min}(s, \dot{s}, e, \dot{e})$ and $\ddot{s}_{max}(s, \dot{s}, e, \dot{e})$ if $\dot{s}$ is admissible and 0 otherwise. This means, for example, that $sat_1(x, s, \dot{s}, e, \dot{e})$ is finite and that $\mid sat_1(x, s, \dot{s}, e, \dot{e}) \mid \leq \mid x \mid$ for all x, s, $\dot{s}$, e, and $\dot{e}$. Note that $\dot{s}$ being admissible means that the resulting τ, $\tau = b_1\ddot{s} + b_2$, will also be admissible, i.e. inside the torque limits. The modified algorithm is then given by

$$\begin{aligned} a_r &= \beta a_n(s) + \frac{1}{2}\alpha(v_n(s)^2 - \dot{s}^2) \\ \ddot{s} &= sat_1(a_r, s, \dot{s}, e, \dot{e}) \\ \dot{s}(0) &= \dot{s}_{n_0} \\ s(0) &= s_{n_0} \end{aligned} \tag{4.3}$$

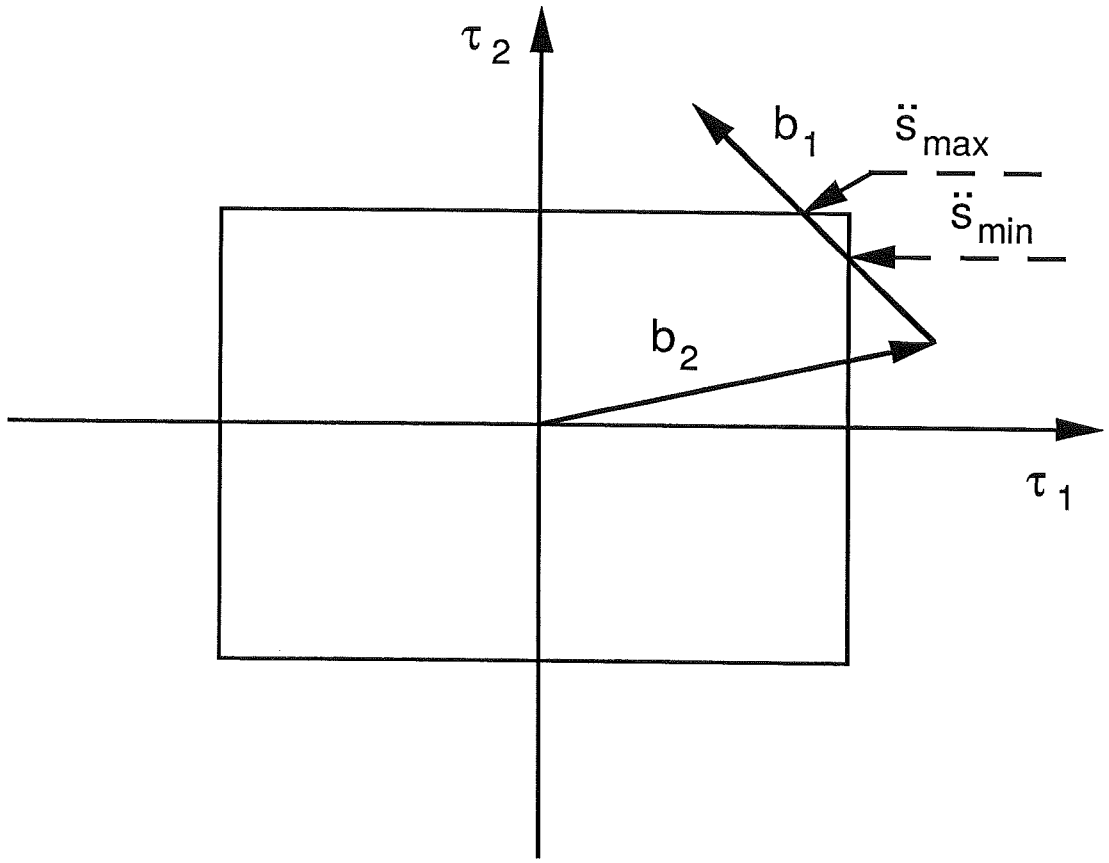

Figure 4.1 A situation that is handled by Algorithm 1, but not by the modified Algorithm 1. The path speed $\dot{s}$ is not admissible, as can be seen from the bounds on $\ddot{s}$, which have the same sign.

The modified Algorithm 1 can only handle path speeds $\dot{s}$ that are admissible. If the torque constraints can be expressed as $\mid \tau \mid \leq \tau_{max}$, this means path speeds $\dot{s}$ such that $\mid b_2 \mid \leq \tau_{max}$. The unmodified algorithm can handle the case $\mid b_2 \mid > \tau_{max}$, which typically would mean higher path speeds. The difference is illustrated for the 2-joint case in Figures 4.1 and 4.2.

Stability of the Modified Algorithm 1

The interpretation of s as a transformed time variable will now be used to rewrite the modified algorithm. We will use the same notation as in Section 3.2. The nominal and the actual velocity profiles are $v_n(s)$ and $v(s)$, where $v(s)$ is the function obtained when the path speed $\dot{s}$ is viewed as a function of s, i.e. $\dot{s}(t) = v(s(t))$. The variables y_r and y are given

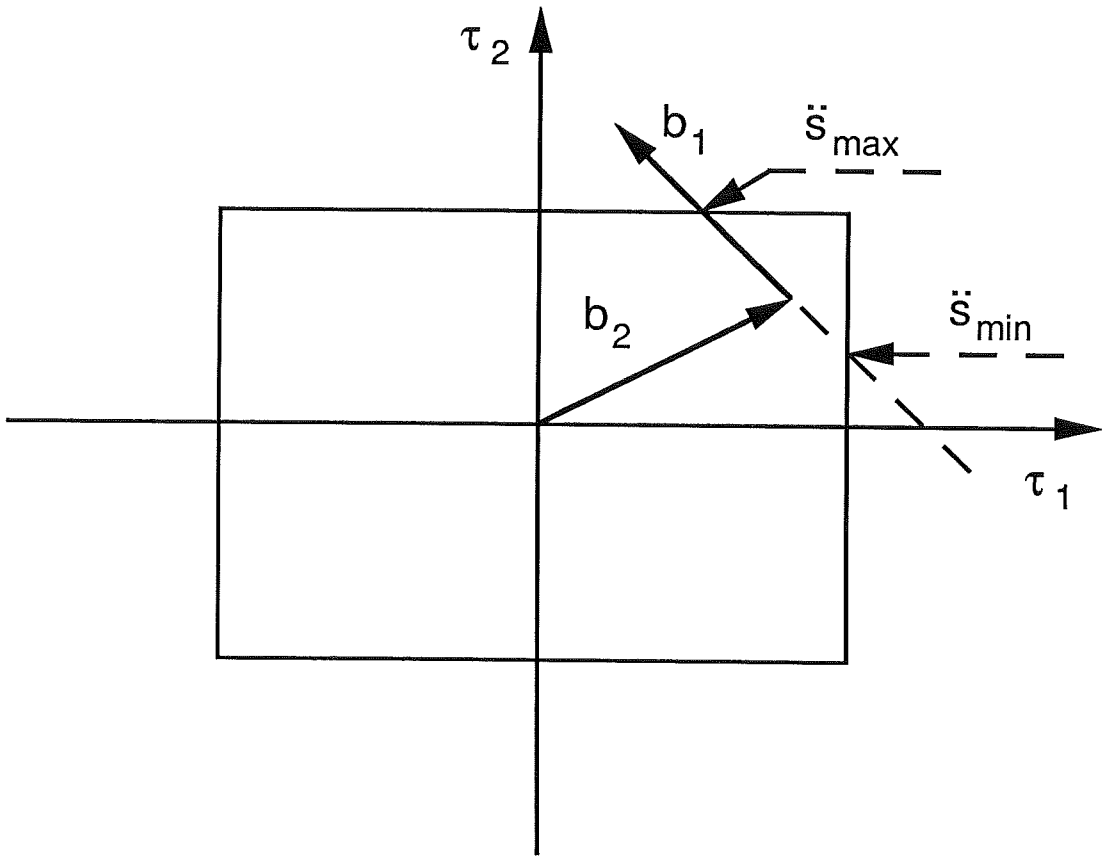

Figure 4.2 A situation where the path speed $\dot{s}$ is admissible, i.e. also the modified Algorithm 1 would lead to a limitation of $\ddot{s}$, resulting in admissible torques.

by

$$y(s) = \frac{1}{2}v(s)^2, \quad y_r(s) = \frac{1}{2}v_n(s)^2$$

The path acceleration $\ddot{s}$ is then given by $\ddot{s} = \frac{dy}{ds}$, see equation (3.14), and the nominal path acceleration is given by the nominal acceleration profile $a_n(s) = \frac{dy_r}{ds}$, see equation (3.13).

The modified Algorithm 1, equation (4.3), can now be written in the transformed time scale as

$$\begin{aligned} a_r &= \beta\frac{dy_r}{ds} + \alpha(y_r - y) \\ \frac{dy}{ds} &= sat_1(a_r, s, v, e, \dot{e}) \\ y(s_0) &= \frac{1}{2}\dot{s}_{n_0}^2 \end{aligned} \tag{4.4}$$

where $s_0 = s_{n_0}$, and $v(s(t)) = \dot{s}(t)$. A stability result for the modified Algorithm 1 is given in Theorem B.1 in Appendix B. The result is based on the observation that the modified algorithm can be interpreted as a first order system with input y_r and output y, and where the derivative of the output, $\frac{dy}{ds}$, is limited. The limitation is done by function sat_1, and occurs only if the bounds used in the limitation have different signs, i.e. if $\dot{s}$ is admissible, see Definition 4.1. If this is not the case, the value of sat_1 is 0, resulting in $\frac{dy}{ds} = 0$.

Theorem B.1 shows that if the gain α is chosen sufficiently large, and $y_r(s)$ and $\beta\frac{dy_r}{ds}$ are bounded, then $y(s)$ and $\frac{dy}{ds}$ are also bounded, and the bounds are independent of the tracking errors e and $\dot{e}$.

The Closed Loop System

The closed loop system includes the robot with torque limitations, the primary controller and the scaling algorithm, in this case the modified Algorithm 1. A stability result for the closed loop system is given in Theorem B.2 in Appendix B. The following reasoning is given to demonstrate the main ideas in the stability analysis of the closed loop system.

The stability result for the modified Algorithm 1, Theorem B.1, gives bounds on the actual velocity profile that are functions only of the nominal velocity profile. This means that bounds on the actual velocity profile can be computed independently of the tracking errors e and $\dot{e}$. This can be used to obtain a stable closed loop system by the following procedure.

1. Given a nominal velocity profile, bounds on the actual velocity profile are given by Theorem B.1. Furthermore, the nominal trajectory should have the property that $\dot{s}$ is admissible when $\dot{s}$ is below the bound given by Theorem B.1, *and* the tracking errors are smaller than a given tolerance, e.g. given as $\mid e(t) \mid \leq e_{max}$ and $\mid \dot{e}(t) \mid \leq \dot{e}_{max}$ for all t. This means that the resulting torques will also be admissible as long as the tracking errors are smaller than the given tolerance.
2. The primary controller should give tracking errors below the specified tolerance, when the reference trajectory is satisfying the bounds computed by Theorem B.1.
3. Since the bounds computed in 1, gives tracking errors below the tolerances, $\dot{s}$ will be admissible during the complete motion, and therefore the torques will also be admissible.

Note that in 2, the torque limits do not have to be taken into account.

The primary controller gives bounds on the tracking errors, given the bounds on the reference trajectory. The bounds on the reference trajectory are independent of the tracking errors, and given by Theorem B.1. This means that the tracking error bounds will be satisfied during the motion. Since the path speed $\dot{s}$ is admissible for all tracking errors below the tracking error bounds, the resulting torques will also be admissible. The possibility of achieving a primary controller with guaranteed tracking performance, given bounds on the reference trajectory, is demonstrated in Appendix B, for a computed torque controller with model uncertainties.

4.2 Velocity Profile Scaling

An approximate analysis of Algorithm 2 will be presented. It is shown how a constant scaling of the velocity profile influences the torques, and requirements on a scaling factor that results in admissible torques, when the trajectory is a minimum time trajectory, are given. Limitations and modifications of Algorithm 2 are also discussed.

Scaling of the Velocity Profile

The rigid body dynamics when the robot is moving along the path are written as

$$a_1(s)\ddot{s} + a_2(s)\dot{s}^2 + a_3(s)\dot{s} + a_4(s) = \tau \tag{4.5}$$

Assume the path velocity $\dot{s}$ can be expressed as a velocity profile $\dot{s}(t) = v(s(t))$. The path acceleration $\ddot{s}$ is then represented by the acceleration profile $a(s)$ as

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

A constant scaling of the velocity profile $v(s)$ is now introduced by replacing $\dot{s} = v(s)$ with $\gamma v(s)$, and replacing $\ddot{s} = a(s)$ with $\gamma^2 a(s)$, where γ is the scaling factor. Equation (4.5) then becomes

$$a_1(s)\gamma^2 a(s) + a_2(s)\gamma^2 v(s)^2 + a_3(s)\gamma v(s) + a_4(s) = \tau(s) \tag{4.6}$$

which shows how the scaling factor influences the torques. If there is no viscous friction the term $a_3(s)\gamma v(s)$ is zero. Introducing $\tau' = \tau - a_4(s)$,

we see that γ^2 can then be viewed as a scaling of τ'. Constant time scaling of a trajectory is also treated in (Hollerbach, 1984).

Scaling of a Minimum Time Trajectory

Requirements on a scaling factor that results in admissible torques when the trajectory is minimum time will be derived. We will assume decoupled linear dynamics, in the form

$$m_i(x)\ddot{x}_i + d_i(x) = \tau_i \tag{4.7}$$

where $1 \leq i \leq n$, the number of joints. The parameters m_i typically represents the mass of the robot. The parameter d_i could represent gravity if the linear model is a simplification of the nonlinear model, but could also represent certain types of friction. If the friction is of the form $d_0 sign(\dot{x}_i)$, and the robot is moving on the path $f(s)$, then $\dot{x}_i = f'(s)\dot{s}$. If $\dot{s}$ is positive during the motion, $sign(\dot{x}_i) = sign(f'(s)\dot{s}) = sign(f'(s))$, a function only of the position of the robot, represented by the path parameter s, which motivates that the term $d_i(x)$, a function only of the position x, could represent friction. The torque constraints are given by

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

Moving the robot along the path gives

$$m_i(s)(f_i'(s)a(s) + f_i''(s)v(s)^2) + d_i(s) = \tau_i(s)$$

If the trajectory is a scaling of the nominal velocity profile, i.e. $v(s) = \gamma v_n(s)$, and $a(s) = \gamma^2 a_n(s)$, we get

$$m_i(s)(f_i'(s)\gamma^2 a_n(s) + f_i''(s)\gamma^2 v_n(s)^2) + d_i(s) = \tau_i(s)$$

The scaling factor γ should then be chosen so that

$$\tau_i^{min} \leq \tau_i(s) \leq \tau_i^{max}$$

for all s and i, $1 \leq i \leq n$. Since the trajectory is minimum time, one or more joints are always at the limit. Suppose that, for a given s, joint j is the limiting joint, i.e. joint j is one of the joints having torques at the limit, and that the limitation is max, i.e. the nominal velocity profile

$v_n(s)$ results in $\tau_j(s) = \tau_j^{max}$. Furthermore, suppose that if γ is chosen so that the limiting joint is inside the torque limits, all other joints will also be inside the torque limits. The scaling factor γ should then satisfy

$$m_j(s)\gamma^2(f_j'(s)a_n(s) + f_j''(s)v_n(s)^2) + d_j(s) \leq \tau_j^{max} \tag{4.8}$$

We will also assume that the limiting joint during motion is the same as the limiting joint in the trajectory planning. The trajectory planning is based on a model with parameter errors, given by

$$\hat{m}_i(x)\ddot{x}_i + \hat{d}_i(x) = \tau_i \tag{4.9}$$

The torque limits are assumed to be known. Using the fact that the nominal trajectory is a minimum time trajectory for the available model (4.9), we get

$$\hat{m}_j(s)(f_j'(s)a_n(s) + f_j''(s)v_n(s)^2) + \hat{d}_j(s) = \tau_j^{max}$$

Inserting this into (4.8) gives

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

where we have assumed $\tau_j^{max} - d_j(s) > 0$, and $\tau_j^{max} - \hat{d}_j(s) > 0$. If the limitation is min, the inequality

$$m_j(s)\gamma^2(f_j'(s)a_n(s) + f_j''(s)v_n(s)^2) + d_j(s) \geq \tau_j^{min} \tag{4.10}$$

is used instead of (4.8). The trajectory planning gives

$$\hat{m}_j(s)(f_j'(s)a_n(s) + f_j''(s)v_n(s)^2) + \hat{d}_j(s) = \tau_j^{min}$$

Inserting this into (4.10), under the assumption that $\tau_j^{min} - d_j(s) < 0$, and $\tau_j^{min} - \hat{d}_j(s) < 0$, gives

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

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

$$\gamma^2 \leq \frac{\hat{m}_j(s)}{m_j(s)} \frac{\tau_j^m - d_j(s)}{\tau_j^m - \hat{d}_j(s)}$$

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

A minimum time trajectory has the property that the nominal path acceleration $a_n(s)$ is at every point s along the path either maximum or minimum, leading to one or more torques at the limit. The trajectory can thus be divided into s-intervals with maximum acceleration or maximum deceleration, where an acceleration interval is always followed by a deceleration interval. Let $s_1 \leq s \leq s_2$ be an acceleration interval, and $s_2 \leq s \leq s_3$ be the following deceleration interval. Suppose for simplicity that only one joint is limiting during each interval, i.e. joint j is limiting during acceleration, and joint k is limiting during deceleration. The quantity g_a, defined by

$$g_a^2 = \min_{s_1 \leq s \leq s_2} \frac{\hat{m}_j(s)}{m_j(s)} \frac{\tau_j^m - d_j(s)}{\tau_j^m - \hat{d}_j(s)} \tag{4.11}$$

is then the maximum constant scaling that gives admissible torques during the acceleration. The corresponding quantity for the deceleration is g_b, given by

$$g_b^2 = \min_{s_2 \leq s \leq s_3} \frac{\hat{m}_k(s)}{m_k(s)} \frac{\tau_k^m - d_k(s)}{\tau_k^m - \hat{d}_k(s)} \tag{4.12}$$

When more than one joint is limiting during each interval, the above reasoning has to be modified. The limiting joints during the acceleration are $j_1, j_2, \ldots j_a$, and the corresponding subintervals along the path are denoted $J_1, J_2, \ldots J_a$. The limiting joints during the following deceleration are $k_1, k_2, \ldots k_d$, and the subintervals are $K_1, K_2, \ldots K_d$. This gives

$$g_a^2 = \min_{1 \leq l \leq a} \left(\min_{s \in J_l} \frac{\hat{m}_{j_l}(s)}{m_{j_l}(s)} \frac{\tau_{j_l}^m - d_{j_l}(s)}{\tau_{j_l}^m - \hat{d}_{j_l}(s)} \right)$$

$$g_b^2 = \min_{1 \leq l \leq d} \left(\min_{s \in K_l} \frac{\hat{m}_{k_l}(s)}{m_{k_l}(s)} \frac{\tau_{k_l}^m - d_{k_l}(s)}{\tau_{k_l}^m - \hat{d}_{k_l}(s)} \right)$$

If γ is chosen as $\gamma = \min(g_a, g_b)$, the torques will be admissible during both intervals. If g_a could be estimated and $g_a \leq g_b$, we could use $\gamma = g_a$.

Estimation of g_a

The update law for γ in Algorithm 2, given by

$$\begin{aligned} \gamma &= 1 + kx_f \\ \dot{x}_f &= \begin{cases} \dot{s}(-ax_f + 1 - \gamma v_n(s)/\dot{s}), & \gamma v_n(s)/\dot{s} \geq 1 \\ \dot{s}(-ax_f), & \gamma v_n(s)/\dot{s} < 1 \end{cases} \end{aligned} \tag{4.13}$$

is interpreted as an estimation of g_a by the following reasoning. Suppose that the actual velocity profile $\dot{s} = v(s)$ is below the current velocity profile $\gamma v_n(s)$, and the reason for this is limited path acceleration, caused by the torque limitations. If $1 - \gamma v_n(s)/\dot{s} \leq ax_f$, see equation (4.13), then γ will be reduced. When the on-line bounds on $\ddot{s}$ are not active, we get $\gamma v_n(s)/\dot{s} \approx 1$, resulting in γ approaching 1 with the time constant $1/a$ in the transformed time scale. If $1/a$ is large compared to the length of the path, γ can be regarded as constant when the bounds on $\ddot{s}$ are not active, hence γ is only adjusted as long as the path acceleration is limited, a result of the torque limitations, which motivates the interpretation of the γ-adjustment as an estimation of g_a, the maximum constant scaling that gives admissible torques during the acceleration.

Algorithm 2 : Limitations and Modifications

We can now state some conditions for Algorithm 2 to give admissible torques during the complete motion.

* If the adjustment of γ during an acceleration interval results in $\gamma \leq g_a$, the maximum constant scaling that gives admissible torques during the acceleration interval, then the torques will be admissible during the acceleration.

* If $g_a \leq g_b$, and $\gamma \leq g_b$ during the following deceleration, the torques will be admissible also during the deceleration part. The condition $\gamma \leq g_b$ during the deceleration can be achieved by choosing $a = 0$ in (4.13). If the apriori model knowledge is such that for each acceleration/deceleration part of the trajectory, $g_a \leq g_b$, and $\gamma \leq g_a$, then the torques will be admissible during the complete motion. Note that the condition $g_a \leq g_b$ can in certain cases be guaranteed by the available model knowledge. Suppose that the parameters of the models, (4.7) and (4.9), are constant, and that only one joint is limiting during each acceleration/deceleration interval. If there are errors only in the mass parameters m_i, equations (4.11) and

(4.12) give $g_a \leq g_b$ if the relative error in the limiting joint during acceleration is smaller or equal to the relative error in the limiting joint during deceleration. If the model knowledge is such that there are errors in the mass parameters, but the relation between the mass parameters is known, for instance the relation between the sizes of the individual links are known but not their exact masses, we would get $g_a = g_b$.

Algorithm 2 is a method for execution of fast trajectories, where it is not sufficient to modify only the slope of the velocity profile as in Algorithm 1. The path velocity has to be modified, and in Algorithm 2, this in done by a scaling of the nominal velocity profile. This is a simple modification, where the influence on the torques can be investigated, (4.6). Other methods of velocity profile modification could be considered. Another limitation is in the analysis, where the primary controller is not taken into account, i.e. it is assumed that the robot is on the path, equation (4.5). It is also assumed that the dynamics are linear and decoupled, equation (4.7).

The inequality $g_a \leq g_b$ can be used to investigate the influence of model errors on the performance during motion. The estimated model quantities, $\hat{m}_i$ and $\hat{d}_i$, could also be modified during trajectory planning in order to guarantee $g_a \leq g_b$.

The update law for γ, equation (4.13), can be modified. The approximate analysis in Section 3.2 assumed a constant quotient $\dot{s}/v_n(s)$, $k \gg a$, and small modifications of the velocity profile, i.e. $\dot{s}/v_n(s) \approx 1$. The following update law could also be used.

$$\dot{\gamma} = \begin{cases} \dot{s}(-a_1\gamma + k(\dot{s}/v_n(s) - \gamma)), \\ \dot{s}(-a_2(\gamma - 1)), \end{cases} \tag{4.14}$$

Different conditions for choosing alternative in (4.14) can be used. The conditions in (4.13), decreases γ only when the current nominal velocity profile $\gamma v_n(s)$ is above the actual velocity profile $\dot{s} = v(s)$. Furthermore, if the time constant in the transformed time scale, $1/a$, is large compared to the length of the path, the increase of γ will be small, see equation (4.13), the lower alternative. This means that a decrease of γ during an acceleration interval may lead to a trajectory that is too conservative further along the path. Another method could be to reset γ to 1 at the start of each acceleration interval, and modify γ only during acceleration, for example by using $a_1 = a_2 = 0$ in (4.14), and switching alternative

in (4.14) at the switching points between acceleration and deceleration. The upper equation in (4.14) can be rewritten as

$$\frac{d\gamma}{ds} = -(a_1 + k)\gamma + k\frac{\dot{s}}{v_n(s)} \tag{4.15}$$

and we see that if we use the transformed time scale s, γ is a low-pass filtered version of $\dot{s}/v_n(s)$, with a time constant given by $1/(a_1 + k)$. Suppose that during an acceleration interval, the on-line bounds on path acceleration are activated, and that this results in $\dot{s} = \tilde{\gamma}v_n(s)$. If $a_1 = 0$, equation (4.15) implies that γ approaches $\tilde{\gamma}$ with a time constant given by $1/k$.

4.3 Summary

The properties of the proposed algorithms for trajectory time scaling have been further investigated. A short summary of the results is given in this section.

Algorithm 1

Algorithm 1 is a method for torque limited path following, where the slope of the velocity profile is modified to meet the requirements given by the torque constraints. A modified version of the algorithm has been shown stable. The modification is interpreted as a restriction on the path speed $\dot{s}$, i.e. the unmodified algorithm can handle higher path speeds. The stability result for the modified Algorithm 1 shows that if the nominal velocity profile is bounded, the actual velocity profile is also bounded, and the bounds are independent of the tracking errors. A stable closed loop system is then obtained by requiring stability of the controlled robot, i.e. the actual reference trajectory should result in tracking errors below a given tolerance, and by having a path speed that results in admissible torques for all tracking errors below the given tolerance.

Algorithm 2

Algorithm 2 is designed to handle nominal minimum time trajectories, where it is not sufficient to only modify the slope of the velocity profile. A scaling of the nominal velocity profile is used for the velocity modification, and the scaling factor is updated on-line. The analysis shows how a constant scaling influences the torques, and requirements on a scaling factor that results in admissible torques, are derived. These requirements include real and estimated robot parameters, and can be used to investigate how model errors influence the performance of the algorithm. Limitations and possible modifications of the algorithm are also discussed.

5. Conclusions

Fast motion along a geometric path is an important problem in robotics. A major problem is that minimum time trajectories or other high speed trajectories lead to torques that are at the limits, hence leaving no control authority to cope with uncertainties. It is reasonable to modify the speed of the reference trajectory if problems occur, and we propose here to use a secondary controller for on-line time scaling of a nominal reference trajectory. The primary controller is parametrized in the scalar path coordinate, but otherwise unchanged, i.e. a well tuned control behavior is kept. The secondary controller modifies a scalar quantity, the path acceleration. This simplifies the design of the secondary controller, and results in coordinated adjustment of the individual joint motions. The nominal trajectory is represented as a function of the path parameter s by the nominal acceleration and velocity profiles. This representation is used as the basis for the design of algorithms for on-line trajectory time scaling, where the path parameter s is generated from a dynamical system, and it is also used to simplify the tuning of the the secondary loop by inspection of the s-$\dot{s}$-diagram for the nominal trajectory.

Two algorithms for trajectory time scaling are designed. The first algorithm uses bounds on the path acceleration, computed from measurements and the torque constraints, together with feedback from the nominal path velocity. The algorithm limits the slope of the velocity profile. The result is that the path acceleration is limited to meet the requirements given by the torque constraints. The second algorithm is an extension of the first algorithm, and is designed to handle nominal minimum time trajectories. A scaling of the nominal velocity profile is introduced, and the scaling factor is modified on-line. The interpretation of the path coordinate as a transformed time scale is used in the analysis of both algorithms, and for tuning of parameters.

The proposed algorithms are verified by analysis, simulations, and experiments. The analysis of the first algorithm includes a stability result for a modified version of the algorithm, where the modification is interpreted as a restriction on the path speed. The analysis of the second algorithm gives relations between the scaling factor required for admissible torques, and model errors. The simulations and the experiments

demonstrate how the algorithms can be used to obtain torque limited path following. The parameters were successfully tuned using the transformed time scale and the s-$\dot{s}$-diagrams. The simulations also include a case showing how the trajectory programming can be simplified by allowing a nominally unfollowable trajectory to be specified. The experimental results show an example of how minimum time trajectory planning can be used in a nonideal situation, where model errors and disturbances are present. A nominally too fast trajectory is modified on-line, and the resulting trajectory gives path following with good torque utilization. The software development for the experiments was simplified by using a compiler generating Modula-2 code from a Simnon system. Different versions of the algorithms could therefore first be tested in simulation, and then efficiently implemented in the real time system used for the experiments.

References

ANDERSSON, L. (1989): "A Modula-2 Real-Time Scheduler – Use and Implementation," CODEN: LUTFD2/TFRT-7414, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

ASADA, H. and J.-J.E. SLOTINE (1986): *Robot Analysis and Control*, John Wiley and Sons, New York.

BOBROW, J.E., S. DUBOWSKY and J.S. GIBSON (1983): "On the Optimal Control of Robotic Manipulators with Actuator Constraints," *ACC-83, San Fransisco.*

BOBROW, J.E., S. DUBOWSKY and J.S. GIBSON (1985): "Time Optimal Control of Robotic Manipulators Along Specified Paths," *Int.J.Robotics Research*, 4, 3.

CRAIG, J.J. (1988): *Adaptive Control of Mechanical Manipulators*, Addison-Wesley.

DAHL, O. (1989): "Generation of Structured Modula-2 Code from a Simnon System Description," CODEN: LUTFD2/TFRT-7416, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

DAHL, O. and L.NIELSEN (1989): "Torque Limited Path Following by On-line Trajectory Time Scaling," *1989 IEEE Conf. Robotics and Automation, Scottsdale, Arizona, USA.*

ELMQVIST, H., K.J. ÅSTRÖM and T. SCHÖNTHAL (1986): *SIMNON User's Guide for MS-DOS Computers*, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

HOLLERBACH, J.M. (1984): "Dynamic Scaling of Manipulator Trajectories," *ASME J. Dynamic Systems, Measurement, and Control*, **106**, 102–106.

JÖNSSON, L. (1988): "An Experimental Robot Simulator (in Swedish)," CODEN: LUTFD2/TFRT-5386, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

LOGITECH, INC. (1987): *Logitech Modula-2 Version 3.0, User's Manual.*

MOLER, C., J. LITTLE and S. BANGERT (1987): *PRO-MATLAB User's Guide*, The MathWorks, Inc., Sherborn, MA, USA.

PFEIFFER, F. and R.JOHANNI (1986): "A Concept for Manipulator Trajectory Planning," *IEEE Conf. Robotics and Automation, San Fransisco.*

SHIN, K.G. and N.D.MCKAY (1985): "Minimum-Time Control of Robotic Manipulators with Geometric Path Constraints," *IEEE Transactions On Automatic Control*, **AC-30**, No. 6, 531–541.

SHIN, K.G. and N.D.MCKAY (1987): "Robust Trajectory Planning for Robotic Manipulators Under Payload Uncertainties," *IEEE Transactions On Automatic Control*, **AC-32**, No. 12, 1044–1054.

SLOTINE, J.-J.E. and H.S. YANG (1989): "Improving the Efficiency of Time-Optimal Path-Following Algorithms," *IEEE Transactions on Robotics and Automation*, **5**, 1, 118–124.

SLOTINE, J.-J.E. and M.W. SPONG (1985): "Robust Robot Control with Bounded Inputs," *J. Robotics Systems*, **2**, 4.

A. Program Descriptions

The software used in the simulations and experiments, presented in Chapter 3, is described. Selected pieces of code are listed, and comments are given.

A.1 Minimum Time Trajectory Planning

An algorithm for minimum time trajectory planning as described in (Shin and McKay, 1985) was implemented in MATLAB (Moler, Little, and Bangert, 1987). The algorithm consists of the steps:

1. Compute the boundaries of the region where $\dot{s}$ is admissible. This is done by using the expressions for the bounds on $\ddot{s}$. Given s, the boundary case for $\dot{s}$ admissible is given by $\ddot{s}_{min}(s, \dot{s}) = \ddot{s}_{max}(s, \dot{s})$.
2. Compute critical points at the boundary of the admissible region. A critical point is defined as a point where a trajectory can touch the admissible region. The criterion for determining if a given boundary point is a critical point is based on a comparison between the bounds on $\ddot{s}$, and the slope of the boundary curve. The critical points are here found by searching the boundary curves. In (Slotine and Yang, 1989), it is described how the critical points can be obtained without computing the boundary curves.
3. Construct a number of curves inside the admissible region. Each curve is the result of integrating one of the equations

$$\begin{aligned} \ddot{s} &= \ddot{s}_{max}(s, \dot{s}) \\ \ddot{s} &= \ddot{s}_{min}(s, \dot{s}) \end{aligned} \tag{A.1}$$

forward or backward, i.e. with increasing or decreasing s. Two of the curves are given by integrating forward with maximum acceleration from the starting point of the path, and integrating backward with maximum deceleration from the end point of the path. The rest of the curves are constructed by integration backward and forward, with maximum and minimum acceleration, from the critical points.

4. The curves are now viewed as a directed graph, and the minimum time trajectory is found by searching the graph for the highest curve between the starting point, and the end point.

The integration of (A.1) is implemented as a zero order hold sampling of a double integrator, and the bounds $\ddot{s}_{min}(s,\dot{s})$ and $\ddot{s}_{max}(s,\dot{s})$, are computed from equation (2.2). The minimum time trajectory used in the simulations and in the experiments was computed with a discretization of 200 points in s, and the sampling interval $h = 0.003s$ was used in the integration.

A.2 Simulations

The simulations were performed using Simnon (Elmqvist, Åström , and Schönthal, 1986). The scaling algorithm was implemented as a Pascal system. The important declarations in the Pascal system are listed below.

```
Module Two_Dim_C(input,output,simnon_data_file);

const
   maxindex = 10;        { e.g...}
   max_n_sn = 500; (* length of nominal trajectory vector *)
   dsmin = 1e-3;
   infinity = 1E9;
   zero = 1E-5;

%include 'use:[ola.poly.simnon]simdefs'

var

(* state *) snom, dsnom : real; (* nominal position and *)
                                (* velocity along the path *)
(* der *) dersnom, derdsnom : real;

(* state *) s, ds : real; (* position and velocity along *)
                          (* the path *)
(* der *) ders, derds : real;

(* input *) x1_in, x1d_in, x2_in, x2d_in : real;
            (* robot position and velocity *)

(* output *) u1, u2 : real; (* control signals *)
```

```
(* output *) r1out, r2out : real; (* reference values *)

(* parameters *)

m1, l1, lc1, d1, m2, l2, lc2, d2,
umin1, umin2, umax1, umax2, (* robot parameters *)
w1, w2, z1, z2 : real;  (* regulator poles *)
rnum : real; (* trajectory planning strategy *)
regnum : real; (* type of controller *)
ulimon, ddslimon : real; (* saturation switches *)
smax : real; (* end of path *)
sstart : real; (* beta is 1 between s0 and sstart *)
compute_unom : real; (* switch for computation of *)
                     (* nominal torques *)
alfanom : real;
rad1, rad2 : real;
beta1 : real;
xfsat : real;
beps : real;

alfa : real;

(* parameter *) k, a : real; (* filter parameters *)
(* state *) xf : real; (* filter state *)
(* der *) derxf : real;


(* parameter *) dseps : real;

(* auxvars *)

dds : real; (* current acceleration *)

u11, u12, u21, u22 : real; (* coefficients in *)
                           (* control signal *)
                           (* u1 = u11*dds + u12 *)
```

The nominal trajectory is stored in a file, and read into a vector when the system is initialized. A parameter is used to select between different algorithms for trajectory scaling. Each algorithm is coded as a separate procedure. At each time step in the integration, one of the procedures is called. The following procedure is used for execution of the nominal

trajectory.

```
procedure new_states_7;
var beta : real;
begin
  update_spos;
  compute_nominal(dsn,ddsn);
  if s < sstart then
    beta := beta1
  else
    beta := 0;
  dds := beta*ddsn + alfa/2*(dsn*dsn - ds*ds);
  derds := dds;
  ders := ds;
end;
```

The procedures update_spos and compute_nominal are common for all scaling algorithms. The current s-position in the nominal trajectory vector is updated by update_spos. The procedure compute_nominal is then called to get the nominal trajectory, represented by the nominal acceleration and velocity profiles. Algorithm 2 is implemented in the following procedure.

```
procedure new_states_8;
var ddsmax1, ddsmax2,
    ddsmin1, ddsmin2, beta, vr_over_ds : real;
begin
  update_spos;
  compute_nominal(dsn,ddsn);
  gamma := 1 + k*xf;
  vr := gamma*dsn;
  if ds > dseps then
    vr_over_ds := vr/ds
  else
    vr_over_ds := 1;
  if xfsat > 0.5 then
    if vr_over_ds < 1 then
      derxf := ds*(-a*xf)
    else
      derxf := ds*(-a*xf + 1 - vr_over_ds)
  else
    derxf := ds*(-a*xf + 1 - vr_over_ds);
  if s < sstart then
    beta := beta1
```

```
    else
      beta := 0;
    ar := beta*gamma*gamma*ddsn + alfa/2*(vr*vr - ds*ds);
    ddsbounds(u11,u12,umax1,umin1,ddsmax1,ddsmin1);
    ddsbounds(u21,u22,umax2,umin2,ddsmax2,ddsmin2);
    ddsmax := min(ddsmax1,ddsmax2);
    ddsmin := max(ddsmin1,ddsmin2);
    if ddsmax < ddsmin + beps then
      dds := ar
    else
      dds := sat(ar,ddsmin,ddsmax);
    derds := dds;
    ders := ds;
  end;
```

The procedure ddsbounds is implemented as follows.

```
  procedure ddsbounds(u1, u2, umax, umin : real;
                      var maxdds, mindds : real);
  const bounds_eps = 1e-3;
  var temp1, temp2 : real;
  begin
    if abs(u1) < bounds_eps then
    begin
      maxdds := infinity;
      mindds := -infinity;
    end
    else
    begin
      temp1 := (umax - u2)/u1;
      temp2 := (umin - u2)/u1;
      maxdds := max(temp1,temp2);
      mindds := min(temp1,temp2);
    end;
  end;
```

A.3 Experiments

The experimental environment is described in Section 3.4. This section contains Simnon and Modula-2 listings, including the primary and the secondary controller. The Simnon part of the combined Simnon/Modula-2 code used in the first implementation of the secondary controller, is

listed below.

```
Discrete system mtlime

"File mtlime.t

state s ds xf xf1 xf2
new news newds newxf newxf1 newxf2

state ttime count oldm
new newttime newcount newoldm

input x11 x12 x21 x22 stop
output dsout sout r1 r2 u1 u2

time t
tsamp ts

ts = t + h
h : 0.03

"The modula code is stored in a special file

% modula-file mtlimemod.t

"time handling

newttime = if oldm and not moving then count*h else ttime
newcount = if moving and not oldm then 1 else count + 1
newoldm = moving

"Output scaling

dsout = if moving then ds/dsscale else 0
sout = if moving then s/sscale else 0

dsscale : 10
sscale : 10

"Process parameters

m1 : 0.0404
m2 : 0.0404
```

```
d1 : 0.0048
d2 : 0.0048

"Control signal limits

ul : 0.2

"Closed loop specifications

w1 : 8
w2 : 8

z1 : 0.7
z2 : 0.7

"Switches

uon : 0
ddslimon : 0
efbon : 0

"Trajectory execution parameters

vs : 1    "Scaling of nominal velocity profile
alfa : 10

"The nominal trajectory is here set to zero. The correct
"values are computed in the Modula-2 code.

ddsn = 0
dsn = 0

"Trajectory update

moving = s < 6.28 - seps and ds > dseps
dseps : -0.0025
seps : 0.01

"The assignment of the path acceleration includes error
"feedback, implemented in efbterm. This can be
"used as an alternative to the upper on-line bound on dds.

"s is reset to zero when the robot is not moving.
```

```
dds = if moving then g*g*ddsn+alfa*(g*g*dsn*dsn-ds*ds)+
      efbterm else -4*ds-4*s

efbterm = if efbon then -ke*(en + tde*den) else 0
ke : 7
tde : 2

en = sqrt((r1-x11)*(r1-x11) + (r2-x21)*(r2-x21))

den = if en > 1e-5 then den1/en else 0
den1 = (r1-x11)*(r1d*ds - 10.10*x12) +
       (r2-x21)*(r2d*ds - 10.10*x22)

"Velocity profile modification

k : 0
a : 0.05

g = 1 + k*xf
div = if ds > 0.01 then g*dsn/ds else 1

dxf = if moving then dxf1 else -8*xf
dxf1 = if div > 1 then ds*(-a*xf + 1 - div) else ds*(-a*xf)

"Reference trajectory

rad1 : 0.4
rad2 : 0.8

sins = sin(s)
coss = cos(s)

r1 = if moving then rad1*(1 - coss) else 0
r2 = if moving then rad2*sins else 0

r1d = if moving then rad1*sins else 0
r1dd = if moving then rad1*coss else 0

r2d = if moving then rad2*coss else 0
r2dd = if moving then -r2 else 0

"Acceleration limitation
```

```
width : 0.1

ddslim = if ddsmax > ddsmin + width and ddslimon then ddslim1
         else dds
ddslim1 = if dds<ddsmin then ddsmin else if dds<ddsmax then
          dds else ddsmax

ddsmin = if zero1 and zero2 then -1e5 else if min1>min2 then
         min1 else min2

ddsmax = if zero1 and zero2 then 1e5 else if max1<max2 then
         max1 else max2

zero1 = b11 > -1e-5 and b11 < 1e-5
zero2 = b21 > -1e-5 and b21 < 1e-5

max1 = if zero1 then 1e5 else if tmax11>tmin11 then tmax11
       else tmin11

tmax11 = if zero1 then 1e5 else (ul - b12)/b11
tmin11 = if zero1 then -1e5 else (-ul - b12)/b11

min1 = if zero1 then -1e5 else if tmax11<tmin11 then tmax11
       else tmin11

max2 = if zero2 then 1e5 else if tmax21>tmin21 then tmax21
       else tmin21

tmax21 = if zero2 then 1e5 else (ul - b22)/b21
tmin21 = if zero2 then -1e5 else (-ul - b22)/b21

min2 = if zero2 then -1e5 else if tmax21<tmin21 then tmax21
       else tmin21

"Controller

b11 = m1*r1d
b12 = if filton then ff1temp + xf1 else ff1temp + fb1
b21 = m2*r2d
b22 = if filton then ff2temp + xf2 else ff2temp + fb2

newxf1 = xf1 + h/tf*(-xf1 + fb1)
```

```
newxf2 = xf2 + h/tf*(-xf2 + fb2)

tf : 0.2
filton : 0

ff1temp = m1*r1dd*ds*ds + d1*10.10*x12

v1 = b11*ddslim + ff1temp + fb1
fb1 = m1*(w1*w1*(r1 - x11) + 2*z1*w1*(r1d*ds - 10.10*x12))

ff2temp = m2*r2dd*ds*ds + d2*10.10*x22

v2 = b21*ddslim + ff2temp + fb2
fb2 = m2*(w2*w2*(r2 - x21) + 2*z2*w2*(r2d*ds - 10.10*x22))

u1 = if not uon then v1 else if v1<-ul then -ul else if v1<ul
     then v1 else ul
u2 = if not uon then v2 else if v2<-ul then -ul else if v2<ul
     then v2 else ul

"State update

dds1 = if moving and ds > limds then ddslim else dds

limds : 0

news = if stop > -0.05 then s else s + h*ds
newds = if stop > -0.05 then ds else ds + h*dds1
newxf = if stop > -0.05 then xf else xf + h*dxf

end
```

The following listing shows the complete algorithm, including the primary controller. Most of the code is generated automatically. The modifications done are mainly for efficiency, e.g. eliminating variables and assignments.

```
IMPLEMENTATION MODULE mtlime;

TYPE

StateType = RECORD
  s, ds, xf, xf1, xf2, ttime, count, oldm : REAL;
```

```
END;

InputType = RECORD
  x11, x12, x21, x22, stop : REAL;
END;

OutputType = RECORD
  dsout, sout, r1, r2, u1, u2 : REAL;
END;

ParType = RECORD
  h, dsscale, sscale, m1, m2, d1, d2, ul, w1, w2, z1, z2, uon,
  ddslimon, efbon, vs, alfa, dseps, seps, ke, tde, k, a, rad1,
  rad2, width, tf, filton, limds : REAL;
END;

SystemRecordRef = POINTER TO RECORD
  State : StateType;
  New : NewType;
  Input : InputType;
  Output : OutputType;
  Par : ParType;
  AuxVar : AuxVarType;
  TimeVar : TimeVarType;
END;

VAR

snvector : ARRAY [1..maxnsn] OF
  RECORD
    sn, dsn, ddsn : REAL;
  END;

datafile : File;

snpos : CARDINAL;

PROCEDURE InitPars(
  S : SystemRecordRef);
VAR System : SystemRecordRef;
BEGIN
  System := S;
  WITH System^.Par DO
```

```
    limds := 0.0;
    filton := 0.0;
    tf := 0.2;
    width := 0.1;
    rad2 := 0.8;
    rad1 := 0.4;
    a := 0.05;
    k := 0.0;
    tde := 2.0;
    ke := 7.0;
    seps := 0.01;
    dseps :=  - 0.0025;
    alfa := 10.0;
    vs := 1.0;
    efbon := 0.0;
    ddslimon := 0.0;
    uon := 0.0;
    z2 := 0.7;
    z1 := 0.7;
    w2 := 8.0;
    w1 := 8.0;
    ul := 0.2;
    d2 := 0.0048;
    d1 := 0.0048;
    m2 := 0.0404;
    m1 := 0.0404;
    sscale := 10.0;
    dsscale := 10.0;
    h := 0.03;
  END;
END InitPars;

PROCEDURE UpDate1(
  S : SystemRecordRef);
VAR System : SystemRecordRef;

VAR continue : BOOLEAN;
    snpos1, snpos2 : CARDINAL;
    sn1, sn2, dsn1, dsn2, ddsn1, ddsn2,
    slopedsn, slopeddsn : REAL;

BEGIN
  System := S;
```

```
  WITH System^.State DO WITH System^.New DO
  WITH System^.Input DO WITH System^.Output DO
  WITH System^.Par DO WITH System^.AuxVar DO
  WITH System^.TimeVar DO

    x11 := ADIn(0);
    x12 := ADIn(1);
    x21 := ADIn(2);
    x22 := ADIn(3);

    movingb := (s < 6.28 - seps) AND (ds > dseps);


IF movingb THEN
  IF NOT (stop >  - 0.05) THEN
    IF snpos < maxnsn THEN
      continue := s > snvector[snpos+1].sn;
    ELSE
      continue := FALSE;
    END;
    WHILE continue DO
      snpos := snpos + 1;
      IF snpos < maxnsn THEN
        continue := s > snvector[snpos+1].sn;
      ELSE
        continue := FALSE;
      END;
    END;
    snpos1 := snpos;
    snpos2 := snpos + 1;
    sn1 := snvector[snpos1].sn;
    sn2 := snvector[snpos2].sn;
    dsn1 := snvector[snpos1].dsn;
    dsn2 := snvector[snpos2].dsn;
    ddsn1 := snvector[snpos1].ddsn;
    ddsn2 := snvector[snpos2].ddsn;
    slopedsn := (dsn2 - dsn1)/(sn2 - sn1);
    slopeddsn := (ddsn2 - ddsn1)/(sn2 - sn1);
    dsn := vs*(dsn1 + slopedsn*(s - sn1));
    ddsn := vs*vs*(ddsn1 + slopeddsn*(s - sn1));
  END;
ELSE
  snpos := 1;
```

```
  tmin11 := ( - ul - b12)/b11;
END;
IF zero1b THEN
  max1 := 1.0e5;
ELSIF tmax11 > tmin11 THEN
  max1 := tmax11;
ELSE
  max1 := tmin11;
END;
ff2temp := m2*r2dd*ds*ds + d2*10.10*x22;
fb2 := m2*(w2*w2*(r2 - x21) +
       2.0*z2*w2*(r2d*ds - 10.10*x22));
IF filton > 0.5 THEN
  b22 := ff2temp + xf2;
ELSE
  b22 := ff2temp + fb2;
END;
IF zero2b THEN
  tmax21 := 1.0e5;
ELSE
  tmax21 := (ul - b22)/b21;
END;
IF zero2b THEN
  tmin21 :=  - 1.0e5;
ELSE
  tmin21 := ( - ul - b22)/b21;
END;
IF zero2b THEN
  max2 := 1.0e5;
ELSIF tmax21 > tmin21 THEN
  max2 := tmax21;
ELSE
  max2 := tmin21;
END;
IF (zero1b) AND (zero2b) THEN
  ddsmax := 1.0e5;
ELSIF max1 < max2 THEN
  ddsmax := max1;
ELSE
  ddsmax := max2;
END;
IF zero1b THEN
  min1 :=  - 1.0e5;
```

```
    ELSIF tmax11 < tmin11 THEN
      min1 := tmax11;
    ELSE
      min1 := tmin11;
    END;
    IF zero2b THEN
      min2 :=  - 1.0e5;
    ELSIF tmax21 < tmin21 THEN
      min2 := tmax21;
    ELSE
      min2 := tmin21;
    END;
  END; END; END; END; END; END; END;
END UpDate1;

PROCEDURE UpDate(
  S : SystemRecordRef);
VAR System : SystemRecordRef;

VAR continue : BOOLEAN;
    snpos1, snpos2 : CARDINAL;
    sn1, sn2, dsn1, dsn2, ddsn1, ddsn2,
    slopedsn, slopeddsn : REAL;

CONST MaxTime = 1.0E10;
BEGIN
  UpDate1(S);
  System := S;
  WITH System^.State DO WITH System^.New DO
  WITH System^.Input DO WITH System^.Output DO
  WITH System^.Par DO WITH System^.AuxVar DO
  WITH System^.TimeVar DO
    IF (zero1b) AND (zero2b) THEN
      ddsmin :=  - 1.0e5;
    ELSIF min1 > min2 THEN
      ddsmin := min1;
    ELSE
      ddsmin := min2;
    END;
    g := 1.0 + k*xf;
    en := sqrt((r1 - x11)*(r1 - x11) + (r2 - x21)*(r2 - x21));
    den1 := (r1 - x11)*(r1d*ds - 10.10*x12) +
            (r2 - x21)*(r2d*ds - 10.10*x22);
```

```
IF en > 1.0e-5 THEN
  den := den1/en;
ELSE
  den := 0.0;
END;
IF efbon > 0.5 THEN
  efbterm :=  - ke*(en + tde*den);
ELSE
  efbterm := 0.0;
END;
IF movingb THEN
  dds := g*g*ddsn + alfa*(g*g*dsn*dsn - ds*ds) + efbterm;
ELSE
  dds :=  - 4.0*ds - 4.0*s;
END;
IF dds < ddsmin THEN
  ddslim1 := ddsmin;
ELSIF dds < ddsmax THEN
  ddslim1 := dds;
ELSE
  ddslim1 := ddsmax;
END;
IF (ddsmax > ddsmin + width) AND (ddslimon > 0.5) THEN
  ddslim := ddslim1;
ELSE
  ddslim := dds;
END;
v1 := b11*ddslim + ff1temp + fb1;
IF  NOT (uon > 0.5) THEN
  u1 := v1;
ELSIF v1 <  - ul THEN
  u1 :=  - ul;
ELSIF v1 < ul THEN
  u1 := v1;
ELSE
  u1 := ul;
END;
v2 := b21*ddslim + ff2temp + fb2;
IF  NOT (uon > 0.5) THEN
  u2 := v2;
ELSIF v2 <  - ul THEN
  u2 :=  - ul;
ELSIF v2 < ul THEN
```

```
  u2 := v2;
ELSE
  u2 := ul;
END;

 DAOut(0,dsout);
 DAOut(1,sout);
 DAOut(2,r1);
 DAOut(3,r2);
 DAOut(4,u1);
 DAOut(5,u2);

IF ds > 0.01 THEN
  div := g*dsn/ds;
ELSE
  div := 1.0;
END;
IF div > 1.0 THEN
  dxf1 := ds*( - a*xf + 1.0 - div);
ELSE
  dxf1 := ds*( - a*xf);
END;
IF movingb THEN
  dxf := dxf1;
ELSE
  dxf :=  - 8.0*xf;
END;
IF (movingb) AND (ds > limds) THEN
  dds1 := ddslim;
ELSE
  dds1 := dds;
END;
IF stop >  - 0.05 THEN
  news := s;
  newds := ds;
  newxf := xf;
ELSE
  news := s + h*ds;
  newds := ds + h*dds1;
  newxf := xf + h*dxf;
END;
newxf1 := xf1 + h/tf*( - xf1 + fb1);
newxf2 := xf2 + h/tf*( - xf2 + fb2);
```

```
    IF (oldm > 0.5) AND  NOT (movingb) THEN
      newttime := count*h;
    ELSE
      newttime := ttime;
    END;
    IF (movingb) AND  NOT (oldm > 0.5) THEN
      newcount := 1.0;
    ELSE
      newcount := count + 1.0;
    END;
    IF movingb THEN
      newoldm := 1.0;
    ELSE
      newoldm := 0.0;
    END;

    s := news;
    ds := newds;
    xf := newxf;
    xf1 := newxf1;
    xf2 := newxf2;
    ttime := newttime;
    count := newcount;
    oldm := newoldm;

    IF t > MaxTime THEN
      t := 0.0;
    ELSE
      t := t + h;
    END;
  END; END; END; END; END; END; END;
END UpDate;

END mtlime.
```

B. Miscellaneous Calculations

B.1 Symmetric Torque Limits

In Lemma 4.1, it is assumed that the torque constraints can be written as $\mid \tau \mid \leq \tau_{max}$. This is for instance the case if $\mid x \mid$, where x is a vector, means $\max_i \mid x_i \mid$ where x_i are the elements of the vector, and if the torque limits are symmetric and equal. If the torque constraints are of the form

$$\tau_i^{min} \leq \tau_i \leq \tau_i^{max} \tag{B.1}$$

this can be achieved by a linear transformation of the torques. Introduce the n-vector $\bar{\tau}$, where n is the number of joint variables, by $\bar{\tau} = A\tau + b$ where A is an n by n, diagonal matrix with diagonal elements

$$A_{ii} = \frac{2\tau_{max}}{\tau_i^{max} - \tau_i^{min}}$$

and b is an n-vector with elements

$$b_i = -\frac{\tau_i^{max} + \tau_i^{min}}{\tau_i^{max} - \tau_i^{min}}\tau_{max}$$

The model

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

with the torque constraints (B.1), is then transformed to

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

with the symmetric and equal torque constraints

$$-\tau_{max} \leq \tau_i \leq \tau_{max}$$

and

$$\bar{H}(q) = AH(q), \quad \bar{v}(q,\dot{q}) = Av(q,\dot{q}),$$
$$\bar{d}(q) = Ad(q), \quad \bar{g}(q) = Ag(q) + b$$

B.2 Stability of the Modified Algorithm 1

The following result shows stability of the modified Algorithm 1, in the sense that a bounded nominal velocity profile $v_n(s)$ results in a bounded actual velocity profile $\dot{s} = v(s)$, where the bounds are independent of the tracking errors.

THEOREM B.1—Stability of the Modified Algorithm 1
If the modified Algorithm 1, equation (4.4), is used for trajectory execution, and

1. $y_r(s) = \frac{1}{2}v_n(s)^2$ satisfies $\Delta + \epsilon < y_r(s) < y_{max} - \Delta$, $\Delta, \epsilon > 0$ for all $s \geq s_0$, and $\mid \beta \frac{dy_r}{ds} \mid$ is bounded for all $s \geq s_0$.
2. The feedback gain α satisfies $\alpha \geq \mid \frac{\beta}{\Delta} \frac{dy_r}{ds}(s) \mid$ for all $s \geq s_0$.
3. $y(s) = \frac{1}{2}v(s)^2$ satisfies the initial conditions $\epsilon \leq y(s_0) \leq y_{max}$, $v(s_0) > 0$.

then

* $\epsilon \leq y(s) \leq y_{max}$ for all $s \geq s_0$, and $v(s) > 0$ for all $s \geq s_0$. The resulting path acceleration $\ddot{s} = \frac{dy}{ds}$ is bounded by $\mid \frac{dy}{ds} \mid \leq \mid \beta \frac{dy_r}{ds} \mid + \mid \alpha y_{max} \mid$.

Proof: The upper bound $y(s) \leq y_{max}$ is first shown. Suppose that for some s, $s = s_2 > s_0$, $y(s_2) > y_{max}$. Then, since $y(s_0) \leq y_{max}$, there are one or more values of $s, s_0 \leq s < s_2$ where $y(s) = y_{max}$. Let s_1 be the largest of these values. This gives $y(s_1) = y_{max}$, $y(s) > y_{max}$ for $s_1 < s < s_2$. Then, by the Mean Value Theorem, there is a point $\sigma_1, s_1 < \sigma_1 < s_2$, such that

$$\frac{dy}{ds}(\sigma_1) = \frac{y(s_2) - y(s_1)}{s_2 - s_1} > 0$$

Using equation (4.4), this implies

$$\frac{dy}{ds}(\sigma_1) = sat_1(a_r(\sigma_1), \sigma_1, v(\sigma_1), e(\sigma_1), \dot{e}(\sigma_1)) > 0 \tag{B.2}$$

Since the lower bound in the limitation by sat_1 is never positive, (B.2) implies $a_r(\sigma_1) > 0$. This gives

$$a_r(\sigma_1) = \beta \frac{dy_r}{ds}(\sigma_1) + \alpha(y_r(\sigma_1) - y(\sigma_1)) > 0$$

But $y(\sigma_1) > y_{max}$ and $y_r(\sigma_1) < y_{max} - \Delta$ implies $y_r(\sigma_1) - y(\sigma_1) < -\Delta < 0$. This results in $a_r(\sigma_1) \leq 0$ if $\alpha \geq \mid \frac{\beta}{\Delta} \frac{dy_r}{ds}(\sigma_1) \mid$ which gives a contradiction. Hence, $y(s) \leq y_{max}$ for all $s \geq s_0$.

The lower bound $y(s) \geq \epsilon$ is now shown by same method. Suppose that for some s, $s = s_4 > s_0$, $y(s_4) < \epsilon$. Then, since $y(s_0) \geq \epsilon$, there are one or more values of $s, s_0 \leq s < s_4$ where $y(s) = \epsilon$. Let s_3 be the largest of these values. This gives $y(s_3) = \epsilon$, $y(s) < \epsilon$ for $s_3 < s < s_4$. Then there is a point $\sigma_2, s_3 < \sigma_2 < s_4$, such that

$$\frac{dy}{ds}(\sigma_2) = \frac{y(s_4) - y(s_3)}{s_4 - s_3} < 0$$

Using equation (4.4), this implies

$$\frac{dy}{ds}(\sigma_2) = sat_1(a_r(\sigma_2), \sigma_2, v(\sigma_2), e(\sigma_2), \dot{e}(\sigma_2)) < 0 \tag{B.3}$$

Since the upper bound in the limitation by sat_1 is always nonnegative, (B.3) implies $a_r(\sigma_2) < 0$. This gives

$$a_r(\sigma_2) = \beta \frac{dy_r}{ds}(\sigma_2) + \alpha(y_r(\sigma_2) - y(\sigma_2)) < 0$$

But $y(\sigma_2) < \epsilon$ and $y_r(\sigma_2) > \epsilon + \Delta$ implies $y_r(\sigma_2) - y(\sigma_2) > \Delta > 0$. This results in $a_r(\sigma_2) \geq 0$ if $\alpha \geq \mid \frac{\beta}{\Delta} \frac{dy_r}{ds}(\sigma_2) \mid$ which gives a contradiction. Hence, $y(s) \geq \epsilon$ for all $s \geq s_0$.

We have shown $\epsilon \leq y(s) \leq y_{max}$ for all $s \geq s_0$. Since $v(s_0) > 0$, and $y(s) = \frac{1}{2} v(s)^2 > 0$ for all $s \geq s_0$, $\dot{s} = v(s)$ will continue to be positive, i.e. $v(s) > 0$ for all $s \geq s_0$. Furthermore, the function sat_1 has the property

$| \; sat_1(x, s, v, e, \dot{e}) \; | \leq | \; x \; |$ for all x, s, v, e, and $\dot{e}$, which gives the bound on $\ddot{s} = \frac{dy}{ds}$ as

$$| \; \frac{dy}{ds} \; | = | \; sat_1(\beta \frac{dy_r}{ds} + \alpha(y_r - y), s, v, e, \dot{e}) \; | \leq | \; \beta \frac{dy_r}{ds} + \alpha(y_r - y) \; | \leq$$
$$| \; \beta \frac{dy_r}{ds} \; | + | \; \alpha(y_r - y) \; | \leq | \; \beta \frac{dy_r}{ds} \; | + | \; \alpha y_{max} \; |$$

□

B.3 Stability of the Controlled Robot

A result showing stability and bounds on the tracking error for a computed torque controller without torque limits, will be derived. The result is found in (Craig, 1988) and the derivation here is similar. We will use the following notations. The vector norm of a vector, $y(t)$, is denoted $| \; y(t) \; |$, and defined as $\max_i | \; y_i \; |$, where y_i are the elements of the vector y. The corresponding induced matrix norm of a matrix, $A(t)$, is denoted $\|A(t)\|$. The notation $\|A\|_t$ means $\sup_{0 \leq x \leq t} \|A(x)\|$ if A is a matrix and $\sup_{0 \leq x \leq t} | \; A(x) \; |$ if A is a vector or a scalar. The robot dynamics are written as

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

The computed torque controller is given by

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

We will assume boundedness of the real and estimated quantities in the dynamic equations, i.e. we assume that $\|H(q)\|_\infty$, $\|\hat{H}(q)\|_\infty$, $\|d(q)\|_\infty$, $\|\hat{d}(q)\|_\infty$, $\|g(q)\|_\infty$, and $\|\hat{g}(q)\|_\infty$ are finite. The elements of the vector of coriolis and centrifugal forces, $v(q, \dot{q})$, can be written

$$v_i(q, \dot{q}) = \sum_{j,k} h_{ijk}(q)\dot{q}_j\dot{q}_k$$

and we assume $\|h_{ijk}(q)\|_\infty$ finite for all i, j, and k, such that $1 \leq i, j, k \leq n$, where n is the number of joint variables. We also assume that $H^{-1}(q)$

and $\hat{H}^{-1}(q)$ exists and that $\|H^{-1}(q)\|_\infty$ and $\|\hat{H}^{-1}(q)\|_\infty$ are finite. Finally, the reference trajectory is assumed to be bounded, i.e. there are numbers $\dot{q}_r^{max}$ and $\ddot{q}_r^{max}$ such that

$$\|\dot{q}_r\|_\infty \leq \dot{q}_r^{max}, \quad \|\ddot{q}_r\|_\infty \leq \ddot{q}_r^{max} \tag{B.4}$$

The equations for the controlled robot can be written as

$$\ddot{e} + K_v\dot{e} + K_p e = u$$

$$u = (I - H^{-1}\hat{H})(\ddot{q}_r + K_v\dot{e} + K_p e) + H^{-1}(v - \hat{v} + (d - \hat{d})\dot{q} + g - \hat{g})$$

where the arguments q and $\dot{q}$ have been left out. Suppose that K_v and K_p are diagonal, and that $k_{p_i} = {k_{v2}}^2/4$. If the motion starts with zero tracking error, $e(0) = \dot{e}(0) = 0$, the tracking errors are bounded by

$$\|e\|_t \leq \beta_1\|u\|_t, \quad \|\dot{e}\|_t \leq \beta_2\|u\|_t$$

where

$$\beta_1 = \max_i \frac{1}{k_{p_i}}$$

$$\beta_2 = \max_i \frac{4}{k_{vi}\exp(1)}$$

We can bound $\|u\|_t$ as

$$\begin{aligned}\|u\|_t \leq &\|I - H^{-1}\hat{H}\|_\infty(\|\ddot{q}_r\|_\infty + \|K_v\|_\infty\|\dot{e}\|_t + \|K_p\|_\infty\|e\|_t) + \\ &\|H^{-1}\|_\infty(\|\tilde{v}\|_t + \|\tilde{d}\|_\infty(\|\dot{q}_r\|_\infty + \|\dot{e}\|_t) + \|\tilde{g}\|_\infty)\end{aligned}$$

where $\tilde{v} = v - \hat{v}$, $\tilde{d} = d - \hat{d}$, and $\tilde{g} = g - \hat{g}$. The elements of the vector $\tilde{v}$ can be written as

$$\begin{aligned}\tilde{v}_i &= \sum_{j,k} h_{ijk}(q)\dot{q}_j\dot{q}_k - \hat{h}_{ijk}(q)\dot{q}_j\dot{q}_k \\ &= \sum_{j,k} \tilde{h}_{ijk}(q)(\dot{q}_{rj}\dot{q}_{rk} - \dot{q}_{rj}\dot{e}_k - \dot{q}_{rk}\dot{e}_j + \dot{e}_j\dot{e}_k) \\ &= \dot{q}_r^T v_{i1}\dot{q}_r + \dot{q}_r^T v_{i2}\dot{e} + \dot{e}^T v_{i3}\dot{e}\end{aligned}$$

Each element of $\tilde{v}$ is bounded by

$$\|\tilde{v}_i\|_t \leq \|v_{i1}\|_\infty \|\dot{q}_r\|_\infty^2 + \|v_{i2}\|_\infty \|\dot{q}_r\|_\infty \|\dot{e}\|_t + \|v_{i3}\|_\infty \|\dot{e}\|_t^2$$

The norm of $\tilde{v}$ is then bounded by

$$\|\tilde{v}\|_t \leq \nu_1 \|\dot{q}_r\|_\infty^2 + \nu_2 \|\dot{q}_r\|_\infty \|\dot{e}\|_t + \nu_3 \|\dot{e}\|_t^2$$

where

$$\nu_j = \max_i \|v_{ij}\|_\infty, \quad 1 \leq j \leq 3$$

This results in bounds on $\|u\|_t$ given by

$$\|u\|_t \leq \alpha_1 + \alpha_2 \|e\|_t + \alpha_3 \|\dot{e}\|_t + \alpha_4 \|\dot{e}\|_t^2$$

where $\alpha_i, 1 \leq i \leq 4$ are constants, given by

$$\begin{aligned}
\alpha_1 =& \|I - H^{-1}\hat{H}\|_\infty \|\ddot{q}_r\|_\infty + \\
& \|H^{-1}\|_\infty (\nu_1 \|\dot{q}_r\|_\infty^2 + \|\tilde{d}\|_\infty \|\dot{q}_r\|_\infty + \|\tilde{g}\|_\infty) \\
\alpha_2 =& \|I - H^{-1}\hat{H}\|_\infty \|K_p\|_\infty \\
\alpha_3 =& \|I - H^{-1}\hat{H}\|_\infty \|K_v\|_\infty + \|H^{-1}\|_\infty (\nu_2 \|\dot{q}_r\|_\infty + \|\tilde{d}\|_\infty) \\
\alpha_4 =& \|H^{-1}\|_\infty \nu_3
\end{aligned}$$

The norms of the tracking errors now satisfy the inequalities

$$\begin{aligned}
\|e\|_t \leq & \frac{\beta_1 \alpha_4}{1 - \beta_1 \alpha_2} \|\dot{e}\|_t^2 + \frac{\beta_1 \alpha_3}{1 - \beta_1 \alpha_2} \|\dot{e}\|_t + \frac{\beta_1 \alpha_1}{1 - \beta_1 \alpha_2} \\
\|e\|_t \geq & -\frac{\alpha_4}{\alpha_2} \|\dot{e}\|_t^2 + \frac{1 - \beta_2 \alpha_3}{\beta_2 \alpha_2} \|\dot{e}\|_t - \frac{\alpha_1}{\alpha_2}
\end{aligned} \tag{B.5}$$

where we have assumed $1 - \beta_1 \alpha_2 > 0$. If

$$\beta_2 \alpha_3 + \beta_1 \alpha_2 < 1 \tag{B.6}$$

and

$$\beta_2 \alpha_3 + \beta_1 \alpha_2 + 2\beta_2 \sqrt{\alpha_1 \alpha_4} < 1 \tag{B.7}$$

then the two quadratic functions of $\|\dot{e}\|_t$, equation (B.5), intersect and the tracking errors $\|e\|_t$ and $\|\dot{e}\|_t$ are bounded by a closed region in the $\|\dot{e}\|_t$-$\|e\|_t$-plane. A bound on $\|\dot{e}\|_t$ is, for example, given by the intersection point where

$$\|\dot{e}\|_t = \frac{1}{2}\frac{1-\beta_2\alpha_3-\beta_1\alpha_2}{\beta_2\alpha_4} - \sqrt{\frac{1}{4}\frac{(1-\beta_2\alpha_3-\beta_1\alpha_2)^2}{\beta_2^2\alpha_4^2} - \frac{\alpha_1}{\alpha_4}} \qquad \text{(B.8)}$$

and the corresponding bound on $\|e\|_t$ is given by replacing inequality by equality in one of the equations (B.5). We can now state the following result.

LEMMA B.1—Stability of the Controlled Robot
If the motion starts with zero tracking error, $e(0) = \dot{e}(0) = 0$, and the reference trajectory is bounded by (B.4), and the model uncertainty is such that (B.6) and (B.7) are satisfied, then the tracking errors $\|e\|_t$ and $\|\dot{e}\|_t$ are bounded by a region in the $\|\dot{e}\|_t$-$\|e\|_t$-plane given by the inequalities (B.5). Explicit bounds on $\|e\|_t$ and $\|\dot{e}\|_t$ can be computed by (B.8) and by (B.5) with inequality replaced by equality in one of the equations (B.5). □

B.4 Stability of the Closed Loop System

Motivated by the stability result for the controlled robot, Lemma B.1, the procedure for obtaining a stable closed loop system is now formalized. Assume that the modified Algorithm 1, equation (4.4), is used for trajectory execution. We can then state the following result.

THEOREM B.2—Stability of the Closed Loop System
If

1. The nominal velocity profile $v_n(s)$ satisfies $\Delta+\epsilon < \frac{1}{2}v_n^2(s) < y_{max} - \Delta$, $\Delta, \epsilon > 0$, for all $s \geq s_0$, and $\mid \beta\frac{dy_r}{ds} \mid$ is bounded for all $s \geq s_0$, and the feedback gain α satisfies $\alpha \geq \mid \frac{\beta}{\Delta}\frac{dy_r}{ds}(s) \mid$ for all $s \geq s_0$.
2. The actual velocity profile $\dot{s} = v(s)$ satisfies the initial conditions $\epsilon \leq \frac{1}{2}v(s_0)^2 \leq y_{max}$, $v(s_0) > 0$.
3. The motion starts at $s = s_0$ with zero tracking error, i.e. $s(0) = s_0$, $e(0) = \dot{e}(0) = 0$.

4. The primary controller has the property that $\mid e \mid \leq e_{max}$ and $\mid \dot{e} \mid \leq \dot{e}_{max}$ for all $s \geq s_0$, if $\mid \ddot{s} \mid \leq \mid \beta \frac{dy_r}{ds} \mid + \mid \alpha y_{max} \mid$ and $\mid \dot{s} \mid \leq \sqrt{2y_{max}}$ for all $s \geq s_0$, and $e(0) = \dot{e}(0) = 0$.

5. The nominal trajectory is such that $\dot{s}$ is admissible if $\dot{s} \leq \sqrt{2y_{max}}$, $\mid e \mid \leq e_{max}$, and $\mid \dot{e} \mid \leq \dot{e}_{max}$.

then

* The closed loop system is stable in the sense that $\mid e \mid \leq e_{max}$ and $\mid \dot{e} \mid \leq \dot{e}_{max}$ for all $s \geq s_0$, and τ is admissible for all $s \geq s_0$.

Proof: Assumptions 1 and 2, together with Theorem B.1, give $0 < \dot{s} \leq \sqrt{2y_{max}}$ for all $s \geq s_0$, and $\mid \ddot{s} \mid \leq \mid \beta \frac{dy_r}{ds} \mid + \mid \alpha y_{max} \mid$ for all $s \geq s_0$. Assumptions 3 and 4 then give $\mid e \mid \leq e_{max}$ and $\mid \dot{e} \mid \leq \dot{e}_{max}$ for all $s \geq s_0$. Assumption 5 together with the bound on $\mid \dot{s} \mid$ then implies that $\dot{s}$ will be admissible during the motion, i.e. for $s \geq s_0$, which in turn implies that τ will be admissible for all $s \geq s_0$.

Remark. Note that the result holds for any primary controller, satisfying Assumption 4. Assuming the path functions $f(s)$, $f'(s)$, and $f''(s)$ are bounded, $\dot{q}_r$ and $\ddot{q}_r$ will then also be bounded if $\dot{s}$ and $\ddot{s}$ are bounded. We can then use Lemma B.1, to construct a primary computed torque controller that satisfies Assumption 4, provided the model uncertainty is small enough, i.e. equations (B.6) and (B.7) must be satisfied. □

Motivation of Assumption 5

Assumption 5 will now be motivated by an analysis of the controller parametrization $\tau = b_1 \ddot{s} + b_2$ for a computed torque controller, where b_1 and b_2 are given by (4.2). We will assume that the torque constraints can be written as $\mid \tau \mid \leq \tau_{max}$, and that the path parametrization has the property $f'(s) \neq 0$ for all s. Since the inertia matrix $H(q)$ is invertible, this implies $b_1 \neq 0$ for all s and q. From Lemma 4.1, we then get $\dot{s}$ admissible if and only if $\mid b_2 \mid \leq \tau_{max}$. The elements of the vector $\hat{v}$ can be written as

$$\begin{aligned}
\hat{v}_i &= \sum_{j,k} \hat{h}_{ijk}(q)\dot{q}_j\dot{q}_k \\
&= \sum_{j,k} \hat{h}_{ijk}(q)(\dot{q}_{rj}\dot{q}_{rk} - \dot{q}_{rj}\dot{e}_k - \dot{q}_{rk}\dot{e}_j + \dot{e}_j\dot{e}_k) \\
&= f'^T \hat{v}_{i1} f' \dot{s}^2 + f'^T \hat{v}_{i2} \dot{e}\dot{s} + \dot{e}^T \hat{v}_{i3} \dot{e}
\end{aligned}$$

Each element of $\hat{v}$ is then bounded by

$$| \hat{v}_i | \leq \|\hat{v}_{i1}\|_\infty \|f'\|_\infty^2 \dot{s}^2 + \|\hat{v}_{i2}\|_\infty \|f'\|_\infty \|\dot{e}\|_\infty \dot{s} + \|\hat{v}_{i3}\|_\infty \|\dot{e}\|_\infty^2$$

Then there are numbers $\hat{\nu}_1$, $\hat{\nu}_2$, and $\hat{\nu}_3$, such that

$$| \hat{v} | \leq \hat{\nu}_1 \|f'\|_\infty^2 \dot{s}^2 + \hat{\nu}_2 \|f'\|_\infty \|\dot{e}\|_\infty \dot{s} + \hat{\nu}_3 \|\dot{e}\|_\infty^2$$

We can now bound $| b_2 |$ as $| b_2 | \leq b_{21} + b_{22}$ where

$$\begin{aligned} b_{21} =& \|\hat{H}\|_\infty (\|K_v\|_\infty \|\dot{e}\|_\infty + \|K_p\|_\infty \|e\|_\infty) + \hat{\nu}_3 \|\dot{e}\|_\infty^2 + \\ & \|\hat{d}\|_\infty \|\dot{e}\|_\infty + \|\hat{g}\|_\infty \\ b_{22} =& (\|\hat{H}\|_\infty \|f''\|_\infty + \hat{\nu}_1 \|f'\|_\infty^2) \dot{s}^2 + (\hat{\nu}_2 \|f'\|_\infty \|\dot{e}\|_\infty + \|\hat{d}\|_\infty \|f'\|_\infty) \dot{s} \end{aligned}$$

and we see that if $b_{21}^{max} < \tau_{max}$, where b_{21}^{max} is defined as b_{21} with $\|e\|_\infty$ and $\|\dot{e}\|_\infty$ replaced by e_{max} and $\dot{e}_{max}$, then there exists $\dot{s}_{max}$, e.g. given by

$$\begin{aligned} \tau_{max} =& b_{21}^{max} + (\|\hat{H}\|_\infty \|f''\|_\infty + \hat{\nu}_1 \|f'\|_\infty^2) \dot{s}_{max}^2 + \\ & (\hat{\nu}_2 \|f'\|_\infty \dot{e}_{max} + \|\hat{d}\|_\infty \|f'\|_\infty) \dot{s}_{max} \end{aligned}$$

such that $| b_2 | \leq \tau_{max}$, i.e. $\dot{s}$ is admissible, when $\dot{s} \leq \dot{s}_{max}$, $\|e\|_\infty \leq e_{max}$, and $\|\dot{e}\|_\infty \leq \dot{e}_{max}$.