# Torque Limited Path Following by On-Line Trajectory Time Scaling

Dahl, Ola; Nielsen, Lars

1989

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

CODEN: LUTFD2/(TFRT-7415)/1-08/(1989)

# Torque Limited Path Following by On-line Trajectory Time Scaling

Ola Dahl
Lars Nielsen

Department of Automatic Control
Lund Institute of Technology
February 1989

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

*Document name*

*Date of issue*
February 1989

*Document Number*
CODEN: LUTFD2/(TFRT-7415)/1–8/(1989)

*Author(s)*
Ola Dahl
Lars Nielsen

*Supervisor*

*Sponsoring organisation*
The Swedish National Board for Technical Development (STU-contract 87-01384P)

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

*Abstract*

Paper at the 1989 IEEE International Conference on Robotics and Automation

*Key words*

*Classification system and/or index terms (if any)*

*Supplementary bibliographical information*

*ISSN and key title*

*ISBN*

*Language*
English

*Number of pages*
8

*Recipient's notes*

*Security classification*

# Torque Limited Path Following by On-line Trajectory Time Scaling

Ola Dahl and Lars Nielsen

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

**Abstract** — A feedback scheme for path following by trajectory time scaling is presented. The scheme is used in execution of fast trajectories along a geometric path, where the motion is limited by torque constraints. The time scaling is done by using a secondary controller that modifies a nominal trajectory during motion. 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, closed loop action is possible. 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 functionality of the secondary controller is verified by simulations and experiments. We believe that the concept of on-line feedback time scaling for torque limited path following is new, and hence that the design and analysis are new contributions.

## 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 nominal 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 our work is to execute such fast trajectories, along a given path, by taking errors into account in a feedback scheme for on-line time scaling.

The work is presented by first reviewing some well known trajectory planning algorithms in Section 2. Section 3 describes problems that may occur in the execution of these trajectories, and some earlier attempts to cope with these problems. Our work is based on a secondary feedback loop, and the new idea is to have a feedback scheme that modifies only the time scaling properties 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, thus keeping the desired basic dynamic behavior. In Section 4, the primary controller is written in a form well suited for devising the secondary loop. The main results, algorithms for feedback trajectory scaling to obtain path following, are designed and analyzed in Section 5. Sections 6 and 7 present simulations and experimental results, and the conclusions are given in Section 8.

## 2. Trajectory Planning

Trajectory planning is a calculation of robot trajectories from a given path. Good candidate methods are 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 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, and thus they pose the most challenging problems for trajectory execution. We have therefore restricted the presentation to minimum time even though our work on trajectory execution is not limited to that case, and this section gives as a background a short review of minimum time trajectory planning (Bobrow, Dubowsky, and Gibson, 1983, 1985; Shin and McKay, 1985; Pfeiffer and Johanni, 1986).

### Robot Dynamics and Torque Constraints

The rigid body dynamics of a robot can be written as

$$H(q)\ddot{q} + v(q,\dot{q}) + d(q)\dot{q} + g(q) = \tau \tag{1}$$

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 parameterized form as a vector function $f(s) \in R^n$ of the scalar path parameter $s \in R$, $s_0 \le s \le s_1$, 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 $q = f(s)$, $\dot{q} = f'(s)\dot{s}$, and $\ddot{q} = f''(s)\dot{s}^2 + f'(s)\ddot{s}$. 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 (1) can now be written as

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

where

$$
\begin{aligned}
a_1(s) &= H(f(s))f'(s) \\
a_2(s) &= H(f(s))f''(s) + v(f(s), f'(s)) \\
a_3(s) &= d(f(s))f'(s) \\
a_4(s) &= g(f(s))
\end{aligned} \tag{3}
$$

The torque constraints are in general 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 $E$. These 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 $\tau$, the admissible values of $\ddot{s}$ are easily obtained from eq. (2). Now given only $s$ and bounds on $\tau$, the admissible values of $\dot{s}$ are those where there exist at least one admissible $\ddot{s}$ from eq. (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 $s$, and one example is seen in Figure 3. 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, $A$. It is a necessary condition on a feasible trajectory that the $\dot{s}(s)$-curve is inside the admissible region $A$.

### Minimum time optimization

The minimum time trajectory planning problem can now 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 objective function is the traversal time

$$
t_f = \int_0^{t_f} dt = \int_{s_0}^{s_1} \frac{dt}{ds} ds = \int_{s_0}^{s_1} \frac{1}{\dot{s}} ds \tag{4}
$$

The states $s$ and $\dot{s}$ are constrained by the admissible region $A$ and the input $\ddot{s}$ is constrained by eq. (2). This optimization problem can of course be solved by general methods like dynamic programming, but there exist more convenient ways (Bobrow, Dubowsky, and Gibson, 1983, 1985; Shin and McKay, 1985; Pfeiffer and Johanni, 1986). These methods are also numerical, but has the advantage of a simple interpretation of the result in an $s$-$\dot{s}$-diagram. 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.

### 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 $\tau(t)$. 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, see Figure 3. For example, if the trajectory approaches a boundary point with maximum deceleration (Figure 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.

## 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. This means that the feedforward part of the controller output is at the torque limit, hence leaving no room for feedback action to compensate for e.g. model errors and disturbances. This may then give rise to large tracking errors, that lead to large path deviations. 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 Section 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 paper propose a scheme for on-line modification of the time scaling properties 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. The modification of only a scalar variable, the time scaling $s$, will give a consistent view, and it will give valuable insight into the design and analysis of the secondary loop.

## 4. 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). 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

$$\tau = b_1 \ddot{s} + b_2 \tag{5}$$

where $b_1$ and $b_2$ depend on measured quantities. Compare with equation (2), that is used to compute off-line bounds on $\ddot{s}$. We will use eq. (5) 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 (5). A feedforward controller with position and velocity feedback is given by

$$\tau = \hat{H}(q_r)\ddot{q}_r + \hat{v}(q_r, \dot{q}_r) + \hat{d}(q_r)\dot{q}_r + \hat{g}(q_r) + K_p e + K_v \dot{e} \tag{6}$$

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 (6) gives

$$\begin{aligned} b_1(s) =& \hat{H}(f(s))f'(s) \\ b_2(s,\dot{s},q,\dot{q}) =& (\hat{H}(f(s))f''(s) + \hat{v}(f(s), f'(s))\dot{s}^2 \\ &+ (\hat{d}(f(s))f'(s))\dot{s} + \hat{g}(f(s)) + K_p e + K_v \dot{e} \end{aligned} \tag{7}$$

A computed torque controller is given by

$$\tau = \hat{H}(q)(\ddot{q}_r + K_p e + K_v \dot{e}) + \hat{v}(q, \dot{q}) + \hat{d}(q)\dot{q} + \hat{g}(q) \tag{8}$$

resulting in

$$\begin{aligned} b_1(s,q) =& \hat{H}(q)f'(s) \\ b_2(s,\dot{s},q,\dot{q}) =& \hat{H}(q)(f''(s)\dot{s}^2 + K_p e + K_v \dot{e}) + \hat{v}(q,\dot{q}) \\ &+ \hat{d}(q)\dot{q} + \hat{g}(q) \end{aligned} \tag{9}$$

### On-line Constraints

We can compute on-line constraints on $\dot{s}$ and $\ddot{s}$, in analogy with the off-line constraints in Section 2. The on-line admissible values of $\ddot{s}$ are the values that result in on-line admissible $\tau$, i.e. $\tau \in E$, the set of admissible torques, when $\tau$ is now computed from equation (5). 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 (5) where $b_1$ and $b_2$ depends on measured quantities. Constant bounds on the input torques are written as

$$\tau_i^{min} \leq \quad \tau_i = b_{1i}\ddot{s} + b_{2i} \quad \leq \tau_i^{max}, \quad 1 \leq i \leq n \tag{10}$$

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

$$c_i = \begin{cases} (\tau_i^{max} - b_{2i})/b_{1i}, & b_{1i} > 0 \\ (\tau_i^{min} - b_{2i})/b_{1i}, & b_{1i} < 0 \\ \infty, & b_{1i} = 0 \end{cases} \tag{11}$$

and

$$d_i = \begin{cases} (\tau_i^{min} - b_{2i})/b_{1i}, & b_{1i} > 0 \\ (\tau_i^{max} - b_{2i})/b_{1i}, & b_{1i} < 0 \\ -\infty, & b_{1i} = 0 \end{cases} \tag{12}$$

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

$$\ddot{s}_{max} = \min_i c_i, \quad \ddot{s}_{min} = \max_i d_i \tag{13}$$

Observe that these on-line bounds are dependent of the measured quantities $q$ and $\dot{q}$.

## 5. On-line Trajectory Scaling

The purpose is to modify the time scaling of the reference trajectory $q_r(t)$. From now on, let the nominal trajectory be denoted by $s_n$ so that the nominal reference trajectory is $q_r(t) = f(s_n(t))$. We will in this section present two algorithms for time scaling of the reference trajectory $q_r(t)$. The scaling is done by on-line modification of the scalar function $s_n(t)$ to a new function $s(t)$. The new reference trajectory is then $q_r(t) = f(s(t))$. The first algorithm uses bounds on the path acceleration $\ddot{s}$ together with feedback from the nominal path speed $\dot{s}_n$, 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 as in the previous section, $\tau = b_1(s,\dot{s},q,\dot{q})\ddot{s} + b_2(s,\dot{s},q,\dot{q})$. In order to avoid computing derivatives of measured quantities, the modification of $s(t)$ is done by modifying the path acceleration $\ddot{s}$. 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 $s$, $\dot{s}$, and $\ddot{s}$.

### Execution of the Nominal Trajectory

The following identities will be useful

$$\frac{1}{2}\frac{d}{ds}\dot{s}^2(s) = \dot{s}'(s)\dot{s}(s) \tag{14}$$

$$\frac{1}{2}\frac{d}{ds}\dot{s}^2(s) = \ddot{s}(s) \tag{15}$$

These formulas are of course also true if $s_n$ is substituted for $s$. The nominal trajectory, given by $s_n(t)$, $\dot{s}_n(t)$, and $\ddot{s}_n(t)$, can equivalently be represented by a velocity profile $\dot{s}_n = \dot{s}_n(s_n)$. The nominal path acceleration is obtained by combining eqs. (14) and (15)

$$\ddot{s}_n = \ddot{s}_n(s_n) = \dot{s}_n'(s_n)\dot{s}_n(s_n) \tag{16}$$

The nominal trajectory can thus be obtained as the solution to the differential equation $\ddot{s}_n = \dot{s}_n'(s_n)\dot{s}_n(s_n)$. A simple trajectory execution method is now to integrate the differential equation (16) on line, and to use the result to obtain $q_r$. In order to compensate for errors, e.g. roundoff in the integration of $\ddot{s}$, we introduce feedback from the nominal path velocity, i.e. $\ddot{s} = \ddot{s}_n(s) + h(\dot{s}_n, \dot{s})$. As a first try $h$ proportional to $\dot{s}_n - \dot{s}$ was used. However, better performance can be obtained by the

trajectory execution method

$$\ddot{s} = \ddot{s}_n(s) + \frac{1}{2}\alpha(\dot{s}_n^2(s) - \dot{s}^2) \tag{17}$$

The reason for this method is revealed by inserting (15) into (17)

$$\frac{d}{ds}(\dot{s}_n^2(s) - \dot{s}^2) + \alpha(\dot{s}_n^2(s) - \dot{s}^2) = 0 \tag{18}$$

By interpreting $s$ as a transformed time scale, we see that $\dot{s}^2 \to \dot{s}_n(s)^2$ with a "time constant" $\frac{1}{\alpha}$, which in turn means that the velocity profile $\dot{s}_n$ is approached. Note that using the $s$-time scale makes it easy to tune the gain $\alpha$ by inspection in the $s$-$\dot{s}$ diagram for the nominal trajectory. Another idea is to leave out the nominal path acceleration $\ddot{s}_n(s)$, and only use the velocity profile $\dot{s}_n$. This gives the trajectory execution method

$$\ddot{s} = \frac{1}{2}\alpha(\dot{s}_n^2(s) - \dot{s}^2) \tag{19}$$

which rewritten with use of eq. (18) gives

$$\dot{s}^2 = \frac{\alpha}{\frac{d}{ds} + \alpha}\dot{s}_n^2(s) \tag{20}$$

and we see that $\dot{s}^2$ is a low-pass filtered version of $\dot{s}_n^2(s)$. The time constant of the filter, again in the transformed time scale, is $\frac{1}{\alpha}$. The two methods (17) and (19) for trajectory execution by velocity profile tracking without regard to the torque limits can be summarized in the equation

$$\ddot{s} = \beta\ddot{s}_n(s) + \frac{1}{2}\alpha(\dot{s}_n^2(s) - \dot{s}^2) \tag{21}$$

The parameter $\beta$ equal to one gives (17), and equal to zero gives (19).

### 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 velocity profile tracking, eq. (21), with on-line bounds on $\ddot{s}$

$$\begin{aligned} a_r &= \beta\ddot{s}_n(s) + \frac{1}{2}\alpha(\dot{s}_n^2(s) - \dot{s}^2) \\ \ddot{s} &= sat(a_r, \ddot{s}_{min}, \ddot{s}_{max}) \end{aligned} \tag{22}$$

where $a_r$ is a filter variable, and 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 (13). If the bounds are in conflict then $\ddot{s} = a_r$ is used. The feedback signal $\alpha(\dot{s}_n^2(s) - \dot{s}^2)$ is here used to achieve the nominal path speed $\dot{s}_n(s)$. As long as the $\ddot{s}$-limits are not active, the gain $\alpha$ has the same interpretation as in the case of nominal trajectory execution. This algorithm works well as long as $\dot{s}$ is in the on-line admissible region $A$. The effect is that the saturation function limits the slope of the velocity profile $\dot{s}_n(s)$. The slope may be too large due to modeling errors, e.g. the inertia may be underestimated in some robot configuration. A block diagram of the system with the secondary control loop is shown in Figure 1, and an example using algorithm 1 is simulated in Section 6 and shown in Figure 2.

### Including velocity profile constraints: Algorithm 2

It is not sufficient to consider only the path acceleration constraint when executing minimum time trajectories. Also the constraints on the velocity profile has to be taken into account. Already the nominal trajectory may, at some points along the path, be at the boundary of the nominal admissible region, $A_n$, i.e. the path speed $\dot{s}$ is at the boundary of being admissible,

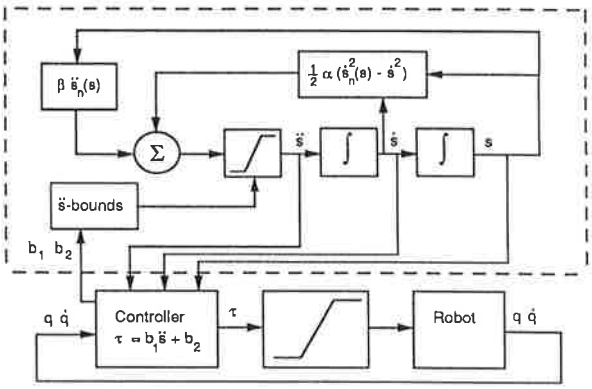


**Figure 1.** A block diagram of the robot with primary and secondary control loop. The primary controller is parameterized in $s$. Observe that $b_1$ and $b_2$ depend on measured quantities.

see Figure 3. Further, uncertainties in the modeling result in an admissible region $A$ that is different from the nominal. Velocity profile tracking in such a case may lead to velocities outside $A$ further along the path especially at the critical switching points.

Algorithm 1 is extended with a modification of the nominal velocity profile in order to handle this problem by the following reasoning. If for some reason the initial acceleration is too high, it will be limited by the algorithm. However, the limitation of the path acceleration may be an indication that the nominal trajectory is too fast, i.e. at some point along the path, the path speed may be outside the on-line-admissible region. The purpose of the modification is to ensure that the path speed $\dot{s}$ is on-line admissible during the rest of the motion, i.e. to ensure that the on-line bounds on $\ddot{s}$ continue to be admissible. We choose to implement this modification as a scaling of the nominal velocity, i.e. to use $\gamma\dot{s}_n(s)$ instead of $\dot{s}_n(s)$ as the nominal path velocity, where $\gamma$ is a scaling factor. Recall that the relation between path acceleration and the velocity profile is $\ddot{s}_n = \dot{s}_n'(s_n)\dot{s}_n(s_n)$, eq. (16). Motivated by this, the nominal acceleration, $\ddot{s}_n(s)$ is replaced by $\gamma^2\ddot{s}_n(s)$. This results in the algorithm

$$\begin{aligned} a_r &= \beta\gamma^2\ddot{s}_n(s) + \frac{1}{2}\alpha(\gamma^2\dot{s}_n^2(s) - \dot{s}^2) \\ \ddot{s} &= sat(a_r, \ddot{s}_{min}, \ddot{s}_{max}) \end{aligned} \tag{23}$$

where the saturation function, the parameters $\alpha$ and $\beta$ have the same meaning as in algorithm 1, equation (21). The completion of this algorithm requires that the scaling factor $\gamma$ is specified. This is done by the following update law

$$\begin{aligned} \gamma &= 1 + kx_f \\ \dot{x}_f &= \begin{cases} \dot{s}(-ax_f + 1 - \gamma\dot{s}_n(s)/\dot{s}), & \gamma\dot{s}_n(s)/\dot{s} \geq 1 \\ \dot{s}(-ax_f), & \gamma\dot{s}_n(s)/\dot{s} < 1 \end{cases} \end{aligned} \tag{24}$$

where $x_f$ is a scalar filter variable, and $k$ is a scalar constant. The update law is intended to adjust the velocity profile during the acceleration parts of the nominal minimum time trajectory, i.e. when $\ddot{s}$ is nominally maximum. Suppose that during the execution of an acceleration phase, the on-line bounds on $\ddot{s}$ are activated, and this results in a path speed $\dot{s}$ that is lower that the current velocity profile $\gamma\dot{s}_n(s)$. The state variable $x_f$

of the filter, equation (24), is then decreased, thereby reducing $\gamma$, i.e. the nominal velocity profile $\gamma \dot{s}_n(s)$ approaches the actual velocity profile given by $\dot{s}$. Tuning rules for the parameters $k$ and $a$ can be derived from the following approximate analysis. Note that the differential equation (24) for $\dot{x}_f$ can be interpreted as a filter in the transformed time scale $s$. Suppose that during an acceleration phase, the on-line bounds on $\ddot{s}$ are activated, and that this results in $\dot{s} = \tilde{\gamma}\dot{s}_n$. Inserting this into the update law (24) for $\gamma$ gives

$$\gamma = 1 + \frac{k}{\frac{d}{ds} + a}(1 - \frac{\gamma}{\tilde{\gamma}}) \quad \Rightarrow$$

$$(\frac{d}{ds} + a)(\gamma - 1) = (\frac{d}{ds} + a)\gamma - a = k(1 - \frac{\gamma}{\tilde{\gamma}}) \quad \Rightarrow$$

$$\gamma = \frac{k + a}{\frac{d}{ds} + a + \frac{k}{\tilde{\gamma}}}$$

If $k$ is chosen much larger than $a$, we get in stationarity $\gamma = \tilde{\gamma}$. Furthermore, assuming small modifications of the nominal trajectory, $\tilde{\gamma} \approx 1$, the parameter $k$ represents a time constant of $\frac{1}{k}$ in the $s$-time scale. If the nominal trajectory does not result in the activation on the $\ddot{s}$-bounds, we get $\dot{s} \approx \dot{s}_n(s)$, and $(\frac{d}{ds}+a)(\gamma-1) = 0$, i.e. $\gamma$ approaches one with the time constant $\frac{1}{a}$ in the transformed time scale. Note that both $k$, giving the time constant for the $\gamma$-update, and $a$, giving the time constant for resetting $\gamma$ to one, can be tuned by using the $s$-$\dot{s}$ diagram for the nominal trajectory.

## 6. 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 7. 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

$$m_i\ddot{x}_i + d_i\dot{x}_i = \tau_i, \quad i = 1, 2 \tag{25}$$

Assuming $x_i$ and $\tau_i$ 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 $|\tau_1| \leq 0.2$, $|\tau_2| \leq 0.2$, and the path is $f_1(s) = 0.4(1 - \cos(s))$, $f_2(s) = 0.8\sin(s)$.

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

$$\dot{s}_n = \begin{cases} \dot{s}_1, & s_0 \leq s \leq s_1 \\ \dot{s}_2, & s_1 \leq s \leq s_2 \\ \dot{s}_2 - \frac{\dot{s}_2}{s_3 - s_2}(s - s_2), & s_2 \leq s \leq s_3 \end{cases}$$

where $\dot{s}_1 = 2.1$, $\dot{s}_2 = 0.7$, $s_1 = 3$, $s_2 = 5$, 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 trajectory is executed by algorithm 1 with $\beta = 0$ and $\alpha = 30$. 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 feedback gains were chosen as $k_{p_1} = k_{p_2} = 81$, $k_{v_1} = k_{v_2} = 18$. The result is shown in Figure 2. The upper left plot shows the path tracking, which is excellent. The slope of the velocity profile has been limited, resulting in a followable reference trajectory, as can be seen in the middle left plot. This has been achieved at a very small delay as can be seen from the difference in $s$ and $s_n$ 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. A minimum time trajectory for the nominal model, equation (25), was computed, and the nominal trajectory and the corresponding torques are shown in Figure 3. Executing the nominal trajectory with a robot model with $m_1$ and $m_2$ 5 % larger than the nominal values, and using the same controller as in the previous example results in large path deviations, see Figure 4 especially the upper left plot. Since the actual robot model is heavier than the robot model used in the trajectory planning, this is expected. Using algorithm 2 with $\beta = 1$, $\alpha = 20$, $k = 4$, and $a = 0.05$ results in accurate path following. The result is shown in Figure 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 increase in the path traversal time is minor as seen in the lower right 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 4 and 5, with respect to the traversal time. Large path following errors have been eliminated by a small increase in traversal time.

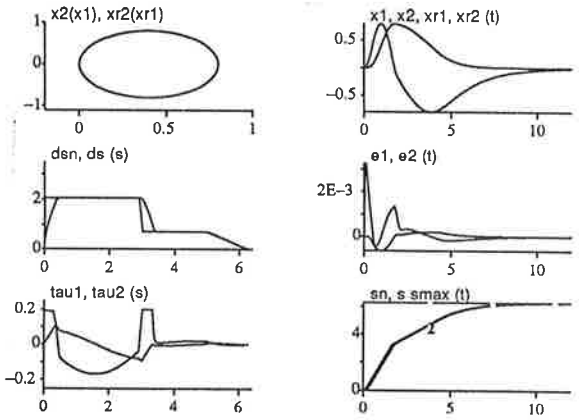


**Figure 2.** The result of using algorithm 1 on a nominally unfollowable reference trajectory. The path acceleration is limited by the algorithm, resulting in path following with limited torques.

## 7. Experimental Results

The same minimum time trajectory as in the previous section has been tried in reality, using a laboratory system controlled by an IBM-AT computer. The path and the primary controller were the same as in the simulation, and the sampling time was 0.017 seconds. The controller parameters were $k_{p_1} = 225$, $k_{p_2} = 100$, $k_{v_1} = 21$, $k_{v_2} = 14$. The nominal trajectory is shown in Figure 3. Algorithm 2 was used, however discretized.

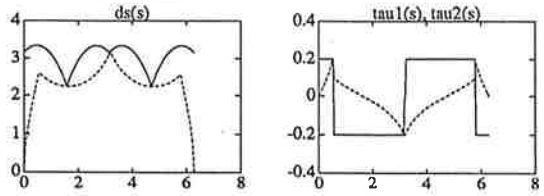


**Figure 3.** An $s$-$\dot{s}$-diagram is shown in the left plot. The nominal minimum time trajectory (dashed) is represented as a velocity profile $\dot{s}(s)$. The solid curve is the boundary of the admissible region. The right plot shows the nominal minimum time torques as functions of the path parameter.

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 6-9. The x-axis is scaled in seconds. The sampling time for the data logging was 0.07 seconds. Figure 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 and in Figure 6, the lower right side plot. In Figure 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 8 shows the result of using algorithm

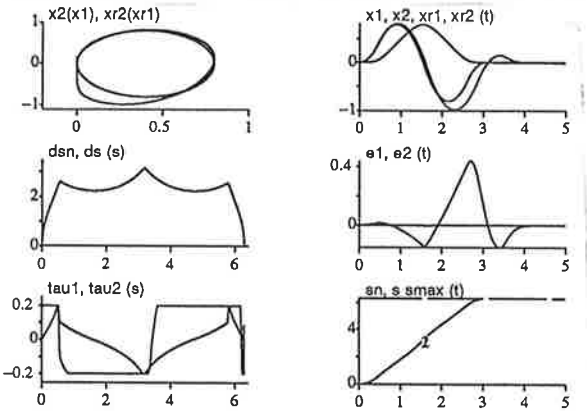


**Figure 4.** An example where nominal trajectory execution is unsuccessful due to an erroneous model. The actual parameters $m_1$ and $m_2$ are 5 % larger than the nominal parameters.

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, 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 6, 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

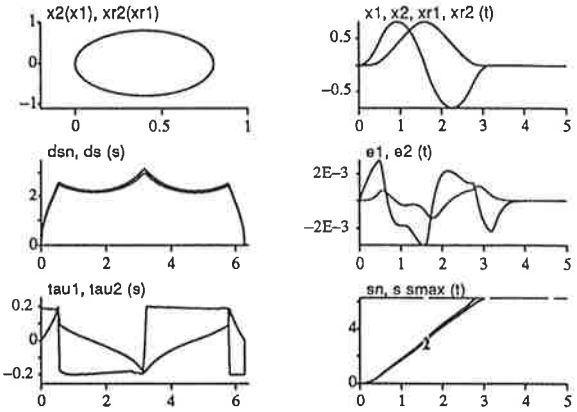


**Figure 5.** Using algorithm 2 on the nominal minimum time trajectory, shown in Figure 3, leads to tracking. The increase in path traversal time is shown in the lower right diagram.

case here. The model used is derived from physical modeling and identification experiments. The model accuracy is good enough for controller design, see Figure 6, but not for minimum time trajectory planning, see Figure 7. A crude measure of the model uncertainty could be the increase in traversal time, see Figure 9, where the nominal velocity $\dot{s}_n$ and the actual velocity $\dot{s}$ are shown as functions of time.

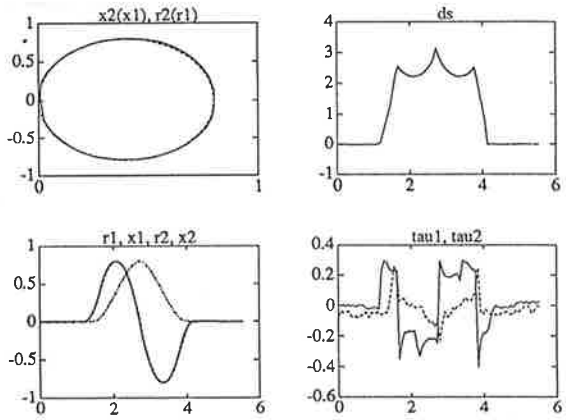


**Figure 6.** An artificial experiment where the torque limits have been removed. The purpose is to show that the primary controller is properly tuned.

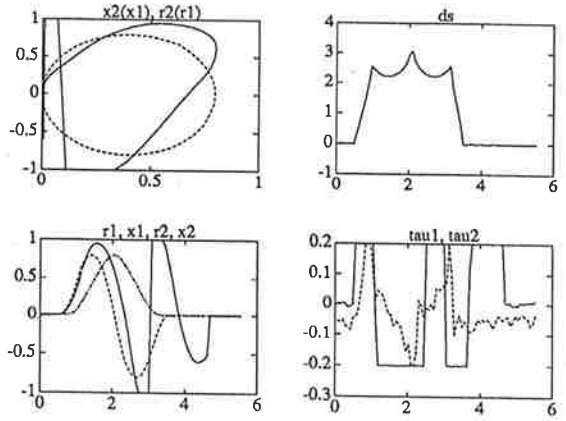


**Figure 7.** Execution of the nominal trajectory when the torques are bounded leads to large path deviations.

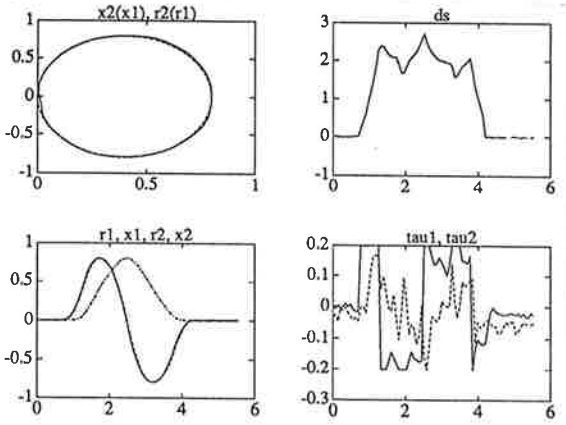


**Figure 8.** The result of using a secondary feedback loop, based on algorithm 2, on the nominal minimum time trajectory. The modified trajectory leads to path following with limited torques.

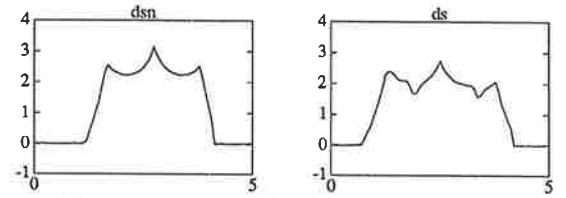


**Figure 9.** The nominal (left) and the modified velocity

## 8. Conclusions

High performance path tracking is an important problem in robotics. A major problem is that minimum time trajectories or other high speed trajectories nominally reaches the torque limits, and hence that there is no control authority left to cope with uncertainties. It is reasonable to modify the speed of the trajectory if problems occur, but there has been a lack of concepts on how to do it.

The contribution of this paper is to propose a feedback scheme for trajectory time scaling. The concept leaves the primary controller unchanged, i.e. a well tuned control behavior is kept. The primary controller is parametrized in the scalar path coordinate $s$. A secondary loop is devised based on the same scalar path coordinate, thus providing a consistent concept for high performance trajectory execution. A basic algorithm uses limitations of the path-acceleration $\ddot{s}$ based on controller saturation. An improved algorithm is based on more explicit knowledge about the nominal, high performance trajectory. Specifically, the interpretation of $s$-$\dot{s}$-charts inspire control structures and filters in the variable $s$ for the secondary loop. We believe that the concept of on-line feedback time scaling for torque limited path following is new, and hence that the design and analysis are new contributions.

### Acknowledgements

The work was financially supported by the Swedish National Board for Technical Development under contract 87-01384P.

## 9. References

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.

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.

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