HOW TO ASSIGN TIME-OPTIMAL TRAJECTORIES TO PARALLEL ROBOTS - AN ADAPTIVE JERK-LIMITED APPROACH

Ingo Pietsch  
Robert Bosch GmbH, Stuttgart, Germany  
e-mail: ingo.pietsch@de.bosch.com

Carlos Bier  
Institut für Werkzeugmaschinen und Fertigungstechnik, TU Braunschweig, Langer Kamp 19b, Germany  
e-mail: c.bier@tu-bs.de

Oliver Becker  
Robert Bosch GmbH, Stuttgart, Germany  
e-mail: oliver.becker@de.bosch.com

Jürgen Hesselbach  
Institut für Werkzeugmaschinen und Fertigungstechnik, TU Braunschweig, Langer Kamp 19b, Germany  
e-mail: j.hesselbach@tu-bs.de

Abstract. This paper provides a new algorithm for time-optimal adaptive trajectory planning for parallel robots. The approach enhances existing solutions by including the parameter information obtained for adaptive control schemes in the trajectory planning process. Furthermore, an explicit limitation of the path jerk is realized. A new algorithm to identify reachable points of the maximum velocity curve (MVC), which leads to smooth trajectories, is derived. Experimental results using the plane parallel robot PORTYS demonstrate the high performance of the proposed algorithm.

Keywords: Machine Tools, Trajectory Planning, Parallel Kinematics

1. Introduction

Time-optimal trajectory planning is an effective way to increase the productivity of robotic manipulators for a variety of tasks especially in the areas of handling and assembly. A time-optimal trajectory describes the maximal velocity and acceleration for each pose along a pre-specified path, where the dynamic properties of the robot, limited drive forces or torques as well as different task-specific boundary conditions are met.

A basic algorithm for time-optimal trajectory planning has been developed almost simultaneously by the groups of Boborow, Dubowsky and Gibson (1985), Shin and McKay (1985) and Pfeiffer and Johanni (1986). The basic idea of these algorithms is to describe the robot dynamics in dependency on a path description in parameterized form. Thus, an optimization problem independent of the degrees of freedom (DOF) of the robot is obtained. It has been shown, that an intuitive solution of this problem can be found, eventually based on Pontryagins maximum principle. The approaches differ e.g. in the way reachable points on the maximum velocity curve (MVC) are determined. The MVC describes the maximal velocity for each point on the path which could be reached if no dependencies between position, velocity and acceleration along the path would occur. Thus, it defines the boundary velocity for each trajectory and can only be reached at certain (“reachable”) points. The main drawback of these approaches is the generation of a hard switching between minimal and maximal acceleration on the trajectory causing increasing waste of the drives and robot mechanics.

Pledel and Bestaoui (1995) propose to include the drive dynamics in the trajectory generation using a cost functional, which weights travelling time as well as drive energy in order to achieve a “softer” trajectory characteristic. A similar approach has been developed by Shiller (1996). The difficulty of these approaches is the determination of the cost functional. An implicit jerk limitation is achieved by the algorithm of Constantinescu and Croft (2001) by limiting the drive force rate, leading to reduced strain, improved tracking accuracy and speed. In contrast, the proposed algorithm limits the trajectory jerk explicitly while the drive force rate is implicitly limited. So far, all algorithms can be used for serial as well as for parallel robots. Generally the ratio of payload to moved robot mass of serial robot manipulators amounts typically 1/10 (Hirzinger et al., 2001). Therefore the influence of changing payloads on the dynamic robot behavior is small. On the other hand high structural stiffness of parallel robots allows the construction of light-weight links and joints. As a result, the ratio of payload to moved robot mass decreases. Therefore, payload changes significantly influence the dynamic behavior of the robot.

This changing dynamic behavior has to be taken into account for control as well as for the generation of time-optimal trajectories in order to exploit the full dynamic robot capabilities. Therefore, an algorithm for time-optimal trajectory planning is derived which can be combined with adaptive control approaches. The algorithm is formulated in parameter-linear form, allowing the direct use of parameter estimates of the adaption laws of standard adaptive control
2. Adaptive time-optimal trajectory planning

The aim of time-optimal trajectory planning, as it is commonly understood in this context, is the determination of the maximum velocity profile along a given path that complies with all given dynamic and kinematic robot constraints. The problem can be described mathematically as an optimal control problem aiming to minimize the traversal time

\[ t_c = \min_{s_{\text{path}}} \int_{0}^{1} \frac{1}{v_p} ds_p \]

with the strictly increasing path parameter \( s_p \) and the path velocity \( v_p \) underlying the following constraints

\[ \frac{ds_p}{dt} = v_p, \quad \frac{dv_p}{dt} = a_p, \quad \frac{da_p}{dt} = j_p, \]

where \( a_p \) and \( j_p \) are path acceleration and path jerk respectively. \( q = q(s_p) \) forms the path constraint in joint-coordinates. The drive torque limitations are given as \( \tau_{\text{min}} \leq \tau(s_p, v_p, a_p) \leq \tau_{\text{max}}. \)

The drive velocity constraints are formulated as (3a), the path velocity limitation as (3b), where forward motion along the path is assumed and the jerk limitation as (3c).

\[ \begin{align*}
   (a) \quad v_{q_{\text{min}}} & \leq v_q \leq v_{q_{\text{max}}} \quad \text{with} \quad v_{q_{\text{min}}} \leq 0 < v_{q_{\text{path}}}, \\
   (b) \quad 0 & \leq v_p \leq v_{p_{\text{max}}}, \\
   (c) \quad j_{p_{\text{min}}} & \leq j_p \leq j_{p_{\text{max}}}. 
\end{align*} \]

The admissible torques can be obtained from the robot dynamics. Dynamic modelling of parallel robots is a complex task because of additional kinematic constraints imposed by closure conditions of the kinematic loops (Dasgupta and Mruthyunjaya, 1998). The robot dynamics can always be expressed in parameter-linear form as \( \tau = Y(q, \dot{q}, \ddot{q}) \pi \) with the regressor matrix \( Y \in \mathbb{R}^{l \times n} \) and the dynamic parameter vector \( \pi \in \mathbb{R}^n \), where \( n \) determines the DOF of the robot (Sciavicco and Siciliano, 2000). The usage of the parameter-linear form has two advantages: Firstly, the dynamic parameter estimates of adaptive control algorithms can directly be used to improve the results of the trajectory planning process. Secondly, it needs less computational time (Hesselbach et al., 2003). From (1) it is obvious that the optimal control problem can be solved in the \((s_p, v_p)\)-state-space by finding the maximum admissible value of \( v_p \) for each \( s_p \).

By inserting the parameterized path \( q = q(s_p) \) and its time derivatives as a function of the path parameter \( s_p \) in the robot dynamics, the dimension of the optimization problem can be reduced from 2n to 2 for rigid robots (Dahl, 1992). In order to determine the admissible acceleration from the robot dynamics, \( \tau = Y(q, \dot{q}, \ddot{q}) \pi \) can be reformulated as

\[ \tau = (Y_a(q(s_p), \dot{q}(s_p))a_p + Y_{v_p}(q(s_p), \dot{q}(s_p), \ddot{q}(s_p), v_p))\pi, \]

where \( \dot{q}(s_p) = \dot{q}(s_p) \dot{s}_p, \quad \ddot{q}(s_p) = q''(s_p) \dot{s}_p^2 + \ddot{q}(s_p) a_p \) have been used. With (4) the maximum and minimum dynamically allowed acceleration for each drive \( i \) can be calculated analytically as:
The drive, which is maximal stressed, is responsible for the acceleration limit 
\[ a_{p_{\text{max}}}(s_p, v_p) = \min_i \left( a_{i_{\text{max}}}(s_p, v_p) \right) \],
\[ a_{i_{\text{min}}}(s_p, v_p) = \max_i \left( a_{i_{\text{min}}}(s_p, v_p) \right) \].
Thus, the extreme path accelerations yield
\[ a_{p_{\text{max}}} = \min(a_{p_{\text{max}}}(s_p, v_p), a_{p_{\text{max}}}(t)), \quad a_{p_{\text{min}}} = \max(a_{p_{\text{max}}}(s_p, v_p), a_{p_{\text{min}}}(t)). \] (6)

As a result of the different constraints, for each path parameter \( s_p \), an upper limit for the reachable velocity \( v_{p_{\text{max}}} \) can be calculated which cannot be exceeded without violating at least one constraint. The set of all \( v_{p_{\text{max}}} \) for \([s_0, s_f]\) determines the maximum velocity curve (MVC). Given a pair of state-space variables \((s_p, v_p)\) the constraints (3) can easily be checked:

\( (a) \quad v_q = q'(s_p)v_p \leq v_{\text{max}}, \quad (b) \quad v_p \leq v_{p_{\text{max}}}, \quad (c) \quad j_{p_{\text{min}}} \leq j_p = \frac{d^2v_p}{dt^2} \leq j_{p_{\text{max}}}. \) (7)

With the following considerations the maximum velocity that depends on the drive torque constraints \( \tau_{\text{min}} \leq \tau(s_p, v_p, a_p) \leq \tau_{\text{max}} \) can be calculated. The boundary condition for the drive torque constraints is \( a_{\text{min}} = a_{\text{max}}. \)

With regard to (4), if \( Y_k(s_p) \pi \neq 0 \), this case only occurs if \( a_{p_{\text{max}}} = a_{p_{\text{min}}}. \)

Introducing the abbreviations \( Y_{\text{max}}(s_p, v_p) = (a_1(s_p)v_p^2 + a_2(s_p)) \), the last condition can be checked analytically using (5), thus leading towards

\[ v_{p_{\text{max}}} = \begin{cases} \frac{Y_k(\tau_{\text{min}} - a_1\pi) + Y_k(a_2\pi - \tau_{\text{max}})}{\pi Y_k(a_1\pi - a_2\pi)}, & \text{if } j \land \{ (Y_k(\pi > 0 \land Y_k(\pi > 0) \lor Y_k(\pi < 0 \land Y_k(\pi < 0)) \} \} \quad (8) \\
\frac{Y_k(\pi - a_1\pi) + Y_k(\pi - a_2\pi - \tau_{\text{min}})}{\pi Y_k(a_1\pi - a_2\pi)}, & \text{if } j \land Y_k(s_p) > 0 \land Y_k(s_p) < 0, \quad (9) \\
\frac{Y_k(\pi - a_1\pi) + Y_k(\pi - a_2\pi - \tau_{\text{min}})}{\pi Y_k(a_1\pi - a_2\pi)}, & \text{if } j \land Y_k(s_p) < 0 \land Y_k(s_p) > 0. \quad (10) \]

If at least one element of \( Y_k(s_p) \pi \) equals zero, \( v_{p_{\text{max}}} \) can be calculated according to

\[ v_{p_{\text{max}}} = \frac{\tau_{\text{max}} - a_1'(s_p)}{a_2'(s_p) \pi}, \quad v_{p_{\text{min}}} = \frac{\tau_{\text{min}} - a_1'(s_p)}{a_2'(s_p) \pi}. \] (11)

The minimum value of \( v_{p_{\text{min}}} \) yields the maximum admissible velocity \( v_{p_{\text{max}}} = \min v_{p_{\text{max}}} \), \( \forall v_{p_{\text{max}}} \in \mathbb{R} \).

In general, a trajectory can reach the MVC at single points or on single intervals only because of its dynamic constraints (2). Reachable MVC points are characterized by one of the four following conditions:

\( a) \) Points, where the MVC results from constraints (7) and the acceleration is below the limits defined by (6).
\( b) \) Tangency points where the slope of the MVC is equal to the slope of the trajectory. The condition can be checked by detecting a sign change of

\[ \kappa(s_p) = v_p'(s_p) - v_{\text{max}}'(s_p) = \frac{a_{p_{\text{max}}}(s_p)}{v_{p_{\text{max}}}(s_p)} - \frac{dv_{p_{\text{max}}}(s_p)}{ds_p}. \] (12)

\( c) \) Points, where \( Y_k(s_p) \pi = 0. \)
\( d) \) Discontinuity points of the MVC.
Thus, the algorithm can be broken down into eight steps. It is formulated in a way suited for direct implementation:

**Algorithm 1: Jerk-limited, time-optimal trajectory planning**

A1.1. Integrate forward from $s_1$ with maximum acceleration according to (6) until either $s_p \geq s_f(t_f) = s_c$ or the trajectory exceeds the MVC. In the latter case set $s_1 = s_p$. Go on with step A1.2.

A1.2. Integrate backward from $s_s$ with minimum acceleration until either the curve intersects the forward trajectory or vanishes at the MVC. In the first case go to A1.8, in the second case set $s_s = s_p$, and go on with A1.3.

A1.3. Calculate the MVC once and all reachable points on the MVC between $s_1$ and $s_s$. Continue with A1.4.

A1.4. Proceed from $s_p = s_1$ on the MVC as long as the points on the MVC are reachable or until the backward trajectory is intersected. In the first case attach the new trajectory segment to the forward trajectory and set $s_3$ the last reachable point. If $s_3 = s_1$ continue with A1.7, else with A1.5.

A1.5. Go to the next reachable point on the MVC and set this point $s_1$. If no reachable point is left, the algorithm fails, otherwise proceed with A1.6.

A1.6. Integrate backward from $s_1$ until the forward trajectory is met and connect the new trajectory segment with the forward trajectory. Continue with A1.7. Otherwise go back to A1.5. If $s_1 = s_2$ the trajectory is closed. Proceed with A1.8.

A1.7. Integrate forward from $s_1$ until the backward trajectory is reached or the trajectory vanishes at the MVC. In the first case go to A1.8, in the latter case attach the new trajectory segment to the forward trajectory, set $s_1$ the last point of the new forward trajectory and go back to A1.4.

A1.8. Search for points $s_i$ on the trajectory, where the jerk limitation is violated. This case can occur at intersection points of different algorithm steps, i.e. if trajectory segments from forward or backward integration meet or at intersection points of trajectory segments calculated either with forward or backward integration with trajectory segments of former MVC segments (A1.4). Divide the trajectory at $s_i$ in one front segment with $s_1 \leq s_3 \leq s_1$ and one back segment with $s_3 \leq s_1 \leq s_i$. From $s_i = s_1$ go stepwise backwards on the front trajectory segment, start a new trajectory segment considering the jerk limitation starting at this point. Repeat this procedure until the new trajectory segment meets the back trajectory segment tangentially or $s_3 = s_0$. In the first case the jerk limitation for this point was successful, and the new trajectory segment can be inserted between the actual $s_3$ and the meeting point at the backward trajectory segment. In the latter case the jerk limit cannot be adhered. Thus, either the jerk limitation is relaxed until the algorithm succeeds or the algorithm fails and stops.

![Figure 2a](image-a) ![Figure 2b](image-b) ![Figure 2c](image-c) ![Figure 2d](image-d)

**Figure 2. Time-optimal, jerk-limited and smoothed trajectory for a circular path**

Figure 2a demonstrates the proceeding. Step A1.1 - forward integration - is active between point 0 and point 1. Between point 9 and point 8 the algorithm integrates backwards in step A1.2. In the third step A1.3 the MVC is calculated. Step A1.4 leads to $s_3 = s_1$, and the algorithm identifies the next reachable point on the MVC at point 2 in step A1.5. From point 2 integrate backward and forward in step A1.6 and A1.7, respectively. Again skipping step 4, point 4 is found in step 5. Go on by integrating backward and forward in steps A1.6 and A1.7 and so on until the trajectory is closed at point 8. Finally, the jerk limitation algorithm of step A1.8 is active at points 1, 3, 5, 7 and 8.

The jerk-limiting algorithm is illustrated in Fig. 2b and 2d. Figure 2b depicts how the jerk at point 1 is limited by an iterative search for a jerk-limited trajectory in area 2. It is found within trajectory segment 4. Figure 2d proves that the
Algorithm 2: Identification of reachable MVC points for smooth trajectories

A2.1 A forward or backward trajectory section (FTS or BTS) is calculated according to step A1.6 or A1.7 in algorithm 1. The start value for a reachable point \( x_1 = \left[ s_{p}^{1}, v_{p}^{1} \right] \) is determined using (12). The maximum value of the numerical deviation between the true and the numerically determined reachable point \( \Delta_{lim}^{\pm \left( s_{p}^{1} \right)} > 0 \) is defined.

a. Backward integration: The intersection between FTS and the recent BTS serves as starting point \( x_1 = \left[ s_{p}^{1}, v_{p}^{1} \right] \) for the smoothing algorithm.

b. Forward integration: If the FTS intersects the BTS, the intersection point is the starting point \( x_1 \). If the FTS exceeds the FTS, the BTS is intersected, choose the next higher point on the FTS, \( x_1 \), and compute \( x_{new}^{1} \).

A2.2 Defining the starting conditions for the iterative smoothing algorithm:

a. Backward integration: \( i = i+1 \). Beginning from \( x_i \) integrate forward until \( s_{p} < s_{p}^{1} \). Compute \( \Delta v_{p}^{i} = v_{p}^{i} - v_{p}^{\prime} \). If \( \Delta v_{p}^{i} > \Delta_{lim} v_{p} \), \( x_i \) was too high in the \( s_{p} - v_{p} \) plane. Thus, the next lower point on the FTS is defined as \( x_{new}^{i} \). If \( \left\lfloor \Delta v_{p}^{i} \right\rfloor = \left\lfloor \Delta v_{p}^{i-1} \right\rfloor \) set \( x_i = x_{new}^{i} \) and repeat A2.2. If \( \left\lfloor \Delta v_{p}^{i} \right\rfloor \neq \left\lfloor \Delta v_{p}^{i-1} \right\rfloor \), the starting point is between \( x_{new}^{i} \) and \( x_i \). In this case \( k > 2 \) equidistant points \( x_{j}^{i} \), \( j=1\ldots k \), \( x_{1}^{i}(1) = x_{new}^{i} \), \( x_{1}^{i}(k+1) = x_i \) are inserted in the trajectory segment between \( x_{new}^{i} \) and \( x_i \). Go on with step A2.3. If \( \Delta v_{p}^{i} < \Delta_{lim} v_{p} \), a sufficiently smooth trajectory is found.

b. Forward integration: \( i = i+1 \). Beginning from \( x_i \) integrate forward until \( s_{p} > s_{p}^{1} \) and compute \( \Delta v_{p}^{i} = v_{p}^{i} - v_{p}^{\prime} \). If \( \Delta v_{p}^{i} > \Delta_{lim} v_{p} \), \( x_i \) was too high in the \( s_{p} - v_{p} \) plane. If the BTS in A1.7 was intersected, choose the next lower point on the BTS as \( x_{new}^{i} \); otherwise define \( x_{new}^{i} = \left[ s_{p}^{i}, v_{p}^{i+1} \right] \), where \( v_{p}^{i+1} \) is the velocity of the next lower point on the FTS. If \( \left\lfloor \Delta v_{p}^{i} \right\rfloor = \left\lfloor \Delta v_{p}^{i-1} \right\rfloor \), \( x_i = x_{new}^{i} \) and repeat A2.2. If \( \left\lfloor \Delta v_{p}^{i} \right\rfloor \neq \left\lfloor \Delta v_{p}^{i-1} \right\rfloor \), the starting point is between \( x_{new}^{i} \) and \( x_i \). In this case \( k > 2 \) equidistant points \( x_{j}^{i} \), \( j=1\ldots k \), \( x_{1}^{i}(1) = x_{new}^{i} \), \( x_{1}^{i}(k+1) = x_i \) are inserted in the trajectory segment between \( x_{new}^{i} \) and \( x_i \). Go on with step A2.3. If \( \Delta v_{p}^{i} < \Delta_{lim} v_{p} \), a sufficiently smooth trajectory is found.

A2.3. Iterative search for a smooth trajectory:

a. Backward integration: \( i = i+1 \) and \( j = j+1 \). Starting from \( x_{1}^{j} \) integrate forward until \( s_{p} > s_{p}^{j} \). Compute \( \Delta v_{p}^{j} = v_{p}^{j} - v_{p}^{\prime} \). If \( \Delta v_{p}^{j} < \Delta_{lim} v_{p} \), repeat A2.3. If \( \Delta v_{p}^{j} > \Delta_{lim} v_{p} \), \( k > 2 \) equidistant points \( x_{j}^{i} \), \( j=1\ldots k \), \( x_{1}^{i}(1) = x_{new}^{i} \), \( x_{1}^{i}(k+1) = x_i \) are inserted in the trajectory segment between \( x_{new}^{i} \) and \( x_i \), set \( j = 1 \), repeat A2.3. If \( \Delta v_{p}^{i} < \Delta_{lim} v_{p} \), a sufficiently smooth trajectory is found.

b. Forward integration: \( i = i+1 \) and \( j = j+1 \). Starting from \( x_{1}^{j} \) integrate backward until \( s_{p} < s_{p}^{j} \). Compute \( \Delta v_{p}^{j} = v_{p}^{j} - v_{p}^{\prime} \). If \( \Delta v_{p}^{j} < \Delta_{lim} v_{p} \), repeat A2.3. If \( \Delta v_{p}^{j} > \Delta_{lim} v_{p} \), \( k > 2 \) equidistant points \( x_{j}^{i} \), \( j=1\ldots k \), \( x_{1}^{i}(1) = x_{new}^{i} \), \( x_{1}^{i}(k+1) = x_i \) are inserted in the trajectory segment between \( x_{new}^{i} \) and \( x_i \), set \( j = 1 \), repeat A2.3. If \( \Delta v_{p}^{i} < \Delta_{lim} v_{p} \), a sufficiently smooth trajectory is found.
In Fig. 3a the proceeding of algorithm 2 is depicted. The reachable point 2 on the MVC is firstly determined using the analytic algorithm of Shin and McKay (1985), according to a sign change of equation (12). Because of the finite sample time (0.0005 s for this example), the reachable point 2 is detected with a slight deviation from its true value. According to steps A1.6 and A1.7 it defines the starting conditions for the adjacent backward and forward integration, respectively. These inaccuracies of the integration starting conditions cause the oscillations in the vicinity of point 2. These oscillations result in strong peaks in the acceleration (Fig. 3b) and in the drive forces (Fig. 3c) accordingly. Using the intersection points 1 and 3 of the (oscillatory) trajectory sections with the BTS and FTS, as starting points for steps A2.2 and A2.3 for an iterative accuracy improvement of the computation of the reachable MVC point leads to a smooth trajectory as shown in Fig. 3a. Thus, also the undesired peaks in the path acceleration signal and the drive forces signal are removed, as depicted in Fig. 3b and 3c, respectively.

3. Experimental Results

The robot PORTYS (Fig. 1) is used to evaluate the performance of the proposed approach for adaptive time-optimal path planning. In Tab. 1 some important robot parameters are given:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>moved robot mass</td>
<td>29.2 kg</td>
</tr>
<tr>
<td>max. payload</td>
<td>10.0 kg</td>
</tr>
<tr>
<td>max. drive force</td>
<td>550.0 N</td>
</tr>
<tr>
<td>max. drive velocity</td>
<td>6.0 m/s</td>
</tr>
</tbody>
</table>

Again, it is noteworthy, that the ratio of payload to moved robot mass is high (1/3), thus payload changes are expected to influence significantly the robot dynamic behavior.

Two different test paths are used for the experiments: a circular (Fig. 4a and 4b) and a triangular path (Fig. 4c and 4d). The results of the adaptive, time-optimal trajectories are compared to the results obtained using conventionally planned trajectories based on a trapezoidal acceleration profile. The conventional trajectories are iteratively planned with respect to the maximum payload such that the maximum drive forces are fully exploited but do not exceed the drive force limitations.

An adaptive composite control approach developed by Slotine and Li with bounded-gain forgetting is used to control the PORTYS robot (Slotine and Li, 1989):

\[
\tau_d = B(q, \dot{q}, \ddot{q})\dot{q}_s + C(q, \dot{q}) + g(q) = Y_s(q, \dot{q}, \ddot{q}) \hat{\Pi},
\]

with the estimated mass matrix \( B \in \mathbb{R}^{n \times n} \), the matrix of the estimated centrifugal and coriolis terms \( C \in \mathbb{R}^{n \times n} \), the gravitational force vector \( g \in \mathbb{R}^n \). \( \dot{q} = \dot{q}_d - \Lambda \ddot{q} \) describes a reference velocity, where \( \dot{q}_d \) is the desired position, \( \Lambda \in \mathbb{R}^{n \times n} \) is a weighting matrix and the position deviation is described as \( \ddot{q} = q - \dot{q}_d \). The parameter estimation is performed using a least-squares approach:

\[
\dot{\Pi} = -\Gamma(t)(Y^T_s s + W^T \ddot{y}(t)),
\]

where \( s = \ddot{q} + \Lambda \ddot{q} \) is the weighted position and velocity error, \( W(q, \dot{q}) \) is a low-pass filtered version of \( Y(q, \dot{q}, \ddot{q}) \) (in order to avoid error-prone measurement of \( \ddot{q} \)) and \( R \) weights the ratio between \( s \) and the prediction error \( \ddot{y}(t) = W \Pi - \Pi \). The time-variant adaption gain is calculated according to

\[
\dot{\Gamma}(t) = -\chi(t)\Gamma(t) + \Gamma(t)W^T(t)W(t)\Gamma(t),
\]

with the forgetting factor \( \chi(t) \), which is defined as

\[
\chi(t) = c_0 \left(1 - \frac{\|\Gamma(t)\|}{k_0}\right).
\]

\( k_0 \) describes an upper bound of \( \chi(t) \).
A dSpace DS1103 Rapid-Prototyping System based on a DSP (Motorola PowerPC 604e, 400 MHz) is used as controller hardware. A control sample frequency of 2 kHz could be realized. During the experiments the traversal time \( t_e \) as well as the mean tracking accuracy \( \Delta H^5 \) and drive utilization rate \( \eta_u \) are investigated. We define the mean tracking accuracy in operational space after passing \( N \) times the test paths as

\[
\Delta H^N = \frac{1}{N} \sum_{i=1}^{N} \| p_i(t) - p(t) \| \quad p = [x, y]
\]

For the described experiments we chose \( N = 5 \) in order to compare the tracking error when parameter adaption of the controller is completed. Payloads of \( m_c = 0.0 \) kg and \( m_v = 7.5 \) kg are used to show the influence of payload changes on the dynamic behavior of the robot. Both, traversal times and mean tracking error for the two test paths for conventionally planned (conv.) as well as for time-optimal trajectories (t.o., generated using the approach described above) are compared in Tab. 2. In the case of the time-optimal trajectories, the payload estimate of the adaptive control law (13)-(16) is used to adapt the trajectory to changing dynamic robot properties. For conventionally planned information cannot be used directly to optimize the trajectory. Therefore, in this case for both payloads the same trajectory has been used.

Obviously, the traversal time can significantly be reduced up to 40 % if the time-optimal trajectory is used instead of the conventional trajectory.

<table>
<thead>
<tr>
<th>Trajectory</th>
<th>( m_c ) [kg]</th>
<th>( t_e ) [s]</th>
<th>( \Delta H^5 ) [mm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>circular, conv.</td>
<td>0.0; 7.5</td>
<td>1.04</td>
<td>0.06; 0.08</td>
</tr>
<tr>
<td>circular, t.o.</td>
<td>0.0</td>
<td>0.62</td>
<td>0.62</td>
</tr>
<tr>
<td>circular, t.o.</td>
<td>7.5</td>
<td>0.68</td>
<td>0.51</td>
</tr>
<tr>
<td>triangular, conv.</td>
<td>0.0; 7.5</td>
<td>1.45</td>
<td>0.07; 0.08</td>
</tr>
<tr>
<td>triangular, t.o.</td>
<td>0.0</td>
<td>1.22</td>
<td>0.37</td>
</tr>
<tr>
<td>triangular, t.o.</td>
<td>7.5</td>
<td>1.28</td>
<td>0.35</td>
</tr>
</tbody>
</table>

This result can be explained according to Fig. 2a using equation (1). The traversal time is inversely proportional to the area beyond the trajectory in state-space formulation. Because the time-optimal trajectory better exploits the drive capabilities especially during acceleration and deceleration phases it is located higher in the \( s_y - v_y \)-plane, thus covering a larger area, hence reaching lower traversal times. Furthermore, if the payload information is included, the traversal time is further decreased by app. 10 %.

At the same time, the mean tracking error increases. But even if the relative increase is significant, the absolute tracking error remains considerably small. Figure 5a shows the tracking error and the drive forces for the circular path using the time-optimal trajectory without additional payload.

Figure 4. Circular and triangular test path

Figure 5. Tracking error and drive force for circular testpath with, (a) time-optimal trajectory and no additional payload after five trajectory passes, (b) conventional trajectory and no additional payload after five trajectory passes.
In order to refute the assumption that similar traversal time reductions could be obtained using conventional trajectories which are modified in a way that similar path deviations occur, Figure 5b depicts tracking error and drive forces for a modified conventionally planned trajectory.

Showing a similar maximum tracking error as obtained using the time-optimal trajectory, the drive forces exceed their limitations clearly, whereas the traversal time is only slightly reduced to 1.01 s.

4. Conclusions

The adaptive time-optimal trajectory planning approach presented in this paper enhances existing algorithms by integrating information of dynamic parameter changes, including a jerk limitation and an improved algorithm to determine reachable MVC points. Because of its adaptive properties it is especially suited to the needs of parallel robots obtaining a high ratio of payload to moved robot mass. Experimental results indicate a significant decrease in the reachable traversal times thus leading to increasing productivity of the robotic system. A drawback of the proposed approach are slightly increasing tracking errors. A possibility to overcome this problem may be the application of a path velocity control algorithm as proposed by Dahl (1992).

5. Acknowledgements

This work is supported by the German Research Foundation DFG within the scope of the Collaborative Research Center 562.

6. References


