Chapter 2
Efficient Online Computation of Smooth Trajectories Along Geometric Paths for Robotic Manipulators

Lukas Messner, Hubert Gattringer, and Hartmut Bremer

Abstract This paper presents a fast computation method for time-optimal robot state trajectories along specified geometric paths. A main feature of this new algorithm is that joint positions can be generated in realtime. Hence, not only joint velocities and accelerations limits but also constraints on joint jerks and motor torques can be considered. Jerk limits are essential to avoid vibrations due to (not-modeled) gear or structure flexibilities. For the limitation of motor torques a complete dynamic robot model including Coulomb and viscous friction is used. The underlying optimal control problem is found by projecting the problem onto the geometric path. The resulting state vector contains path position, speed and acceleration while path jerk is used as input. From optimal control theory it follows that the path jerk has to be chosen at its boundaries, which can be computed for each state in each step. Continuous state progress is assured via so called test trajectories which are additionally computed in each step. As an example the algorithm is applied to a six-axis industrial robot moving along a straight line in Cartesian space.

2.1 Introduction

Despite the fact that in the last decades big effort has been put into the development of path planning techniques, most of these methods are not used in industry and the capabilities of robots are often not fully exploited. One reason for this is, that most algorithms are designed for an offline computation of a desired trajectory for the robot's joints $\mathbf{q}_d(t)$ although an online computation is desirable for mainly two reasons: (1) A movement described with a robotic program should start without computation delays and (2) a flexible production system should be able to react on unforeseeable events in realtime.

L. Messner (✉) • H. Gattringer • H. Bremer
Johannes Kepler University, Altenbergerstr. 69, Linz 4040, Austria
e-mail: lukas.messner@jku.at

H. Gattringer and J. Gerstmayr (eds.), *Multibody System Dynamics, Robotics and Control*,
DOI 10.1007/978-3-7091-1289-2_2, © Springer-Verlag Wien 2013

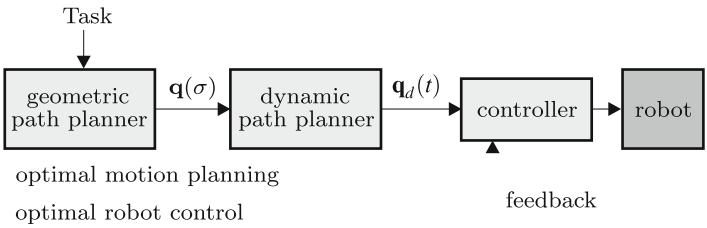

Fig. 2.1 The decoupled approach. A geometric path $\mathbf{q}(\sigma)$ from a geometric path planner is the basis for a time parametrization in the dynamic path planner. The feedback controller is then tracking so computed reference trajectories $\mathbf{q}_d(t)$

Typically, the general online path planning problem leads to a realtime optimal control problem [5]. Unfortunately, current industrial realtime environments are not powerful enough to solve the general problem (e.g. for a six axis robot) using state of the art techniques, like model predictive control (MPC) [9]. However, in this paper a typical industrial case with a predefined geometric path $\mathbf{q}(\sigma)$ is considered (e.g. defined by a robotic program). Therefore, it is known that the optimal control problem can be greatly reduced by projecting it onto the path parameter σ. This separation of the geometric path planning problem, including the inverse kinematics, and the dynamic path planning problem for finding a course in time $\sigma(t)$ was firstly introduced by Bobrow et al. [3] and is called *decoupled approach* [15]. The resulting control setup is also depicted in Fig. 2.1.

There exist many well studied methods for solving the optimal control problem arising from the decoupled approach which can be divided into dynamic programming [2, 11], direct solution methods [8, 13, 18] and indirect solution methods [3, 17]. However, only few existing methods are designed for realtime usage and they often do not fulfill all industrial demands. For example vibration avoiding smoothing techniques like jerk limitations are not considered in Pardo-Castellote's approach [15]. Other methods do not consider a dynamic robot model for the consideration of actuator torques (e.g. [7] or [1]) or they are limited to special geometric paths like straight lines [14].

This paper presents an algorithm which overcomes the realtime problem by not solving the optimal control problem for the whole path at once, but only for a short distance in each step. Firstly, the problem is projected onto the geometric path by introducing a path position, speed and acceleration state vector (Sect. 2.2). With path jerk as input, it is shown that desired joint limits can be formulated as pure state or mixed input state inequality conditions for that system. In Sect. 2.3, possible input values are derived from Pontryagin's Maximum Principle [16]. Thereafter, a time discretization is introduced and continuable states are defined. Finally, in Sect. 2.4 test trajectories are used to check for continuable states. These trajectories

form the key computation for the presented discrete online algorithm. To demonstrate the performance of the method, results for a six-axis industrial robot are shown in Sect. 2.5.

2.2 Problem Statement

In the following, a geometric path

$$\mathbf{q}(\sigma) = [q_1(\sigma), \ldots, q_n(\sigma)]^{\mathrm{T}} \qquad \sigma \in [\sigma_0, \sigma_e] \tag{2.1}$$

is given for a fully actuated robotic manipulator with n joints $q_1, \ldots, q_n$. It is assumed that $\mathbf{q}(\sigma)$ is at least three times differentiable with respect to the arbitrarily chosen path parameter σ.

For the limitation of actuator torques or forces $\boldsymbol{\tau}$, a nonlinear dynamic model is used in the general form

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) + \mathbf{D}\,\dot{\mathbf{q}} + \mathbf{D}_c\,\mathrm{sign}(\dot{\mathbf{q}}) \tag{2.2}$$

with the mass matrix $\mathbf{M}$, the Coriolis and centrifugal force matrix $\mathbf{C}$, which is linear in $\dot{\mathbf{q}}$ and the gravity force vector $\mathbf{G}$. Viscous friction is considered with $\mathbf{D}\,\dot{\mathbf{q}}$ and $\mathbf{D}_c\,\mathrm{sign}(\dot{\mathbf{q}})$ are Coulomb friction torques and forces [4].

2.2.1 Optimal Control Problem

For a given geometric path (2.1) the whole path planning problem is reduced to the task of finding a function $\sigma(t)$. It will be shown that constraints and an optimization criterion for this function lead to an optimal control formulation in the form

$$\min_{u(t)} J := \int_0^{t_e} l(\mathbf{x}(t), u(t))\mathrm{d}t \tag{2.3}$$

$$\text{subject to } \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u),\ \mathbf{x}(0) = \mathbf{x}_0,\ \mathbf{x}(t_e) = \mathbf{x}_e, \tag{2.4}$$

$$\left.\begin{aligned} \mathbf{h}(\mathbf{x}) &\le 0 \\ \mathbf{g}(\mathbf{x}, u) &\le 0 \end{aligned}\right\}\ t \in [0, t_e], \tag{2.5}$$

where J is a cost functional, $\mathbf{f}$ denotes a dynamic system and $\mathbf{h}$ and $\mathbf{g}$ consider pure state and mixed input-state inequality conditions [5].

Herein, the pure time optimal case will be considered by choosing $l(\mathbf{x}(t), u(t)) = 1$. Furthermore, limitations on joint speeds, accelerations, jerks and motor torques will be taken into account with

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} \dot{\mathbf{q}}(x_1, x_2) - \dot{\mathbf{q}}_{max} \\ -\dot{\mathbf{q}}(x_1, x_2) + \dot{\mathbf{q}}_{min} \\ \ddot{\mathbf{q}}(x_1, x_2, x_3) - \ddot{\mathbf{q}}_{max} \\ -\ddot{\mathbf{q}}(x_1, x_2, x_3) + \ddot{\mathbf{q}}_{min} \\ \boldsymbol{\tau}(x_1, x_2, x_3) - \boldsymbol{\tau}_{max} \\ -\boldsymbol{\tau}(x_1, x_2, x_3) + \boldsymbol{\tau}_{min} \end{bmatrix}, \tag{2.6}$$

$$\mathbf{g}(\mathbf{x}, u) = \begin{bmatrix} \dddot{\mathbf{q}}(x_1, x_2, x_3, u) - \mathbf{j}_{max} \\ -\dddot{\mathbf{q}}(x_1, x_2, x_3, u) + \mathbf{j}_{min} \end{bmatrix}, \tag{2.7}$$

by selecting path position, speed and acceleration as components of the state vector $\mathbf{x} = [x_1, x_2, x_3]^{\mathrm{T}} = [\sigma, \dot{\sigma}, \ddot{\sigma}]^{\mathrm{T}}$ and $u = \dddot{\sigma}$ as input of the dynamic system $\mathbf{f}(\mathbf{x}, u) = [x_2, x_3, u]^{\mathrm{T}}$. Dependencies on system input and state components in (2.6), and (2.7) can be easily seen by expanding the time derivatives

$$\dot{\mathbf{q}}(x_1, x_2) = \mathbf{q}'(x_1)\, x_2 \tag{2.8}$$

$$\ddot{\mathbf{q}}(x_1, x_2, x_3) = \mathbf{q}''(x_1)\, x_2^2 + \mathbf{q}'(x_1)\, x_3 \tag{2.9}$$

$$\dddot{\mathbf{q}}(x_1, x_2, x_3, u) = \mathbf{q}'''(x_1)\, x_2^3 + \mathbf{q}''(x_1) 3 x_2 x_3 + \mathbf{q}'(x_1)\, u\,, \tag{2.10}$$

using the differential operator $(.)' := \frac{\partial(.)}{\partial \sigma}$. For the actuator torques and forces, the dynamic equation (2.2) is rewritten to

$$\boldsymbol{\tau}(x_1, x_2, x_3) = \mathbf{a}_0(x_1) + \mathbf{a}_1(x_1)\, x_2 + \mathbf{a}_2(x_1)\, x_2^2 + \mathbf{a}_3(x_1)\, x_3\,, \tag{2.11}$$

with purely σ-depending parameters

$$\begin{aligned} \mathbf{a}_0(x_1) &= \mathbf{G}(\mathbf{q}(x_1)) + \mathbf{D}_c\, \mathrm{sign}(\mathbf{q}'(x_1)) \\ \mathbf{a}_1(x_1) &= \mathbf{D}\, \mathbf{q}'(x_1) \\ \mathbf{a}_2(x_1) &= \mathbf{M}(\mathbf{q}(x_1))\, \mathbf{q}''(x_1) + \mathbf{C}(\mathbf{q}(x_1), \mathbf{q}'(x_1))\, \mathbf{q}'(x_1) \\ \mathbf{a}_3(x_1) &= \mathbf{M}(\mathbf{q}(x_1))\, \mathbf{q}'(x_1)\,. \end{aligned}$$

For an efficient computation of the inequality conditions (2.6), and (2.7) the existence of purely σ-depending parameters is advantageous because a preparation (e.g. using a spline approximation) is possible in advance.

2.3 Optimal System Inputs

For solving an optimal control problem (2.3), (2.4), and (2.5) various techniques exist [5]. Similar to indirect solution methods optimality conditions derived from Pontryagin's Maximum Principle give useful information for the choice of optimal system inputs. Therefore, a Hamiltonian is introduced as

$$H(\mathbf{x}, u, \lambda) := \lambda_0 l + \lambda^{\mathrm{T}} \mathbf{f}(\mathbf{x}, u) = \lambda_0 + \lambda_1 x_2 + \lambda_2 x_3 + \lambda_3 u\,, \tag{2.12}$$

for the problem (2.3), (2.4), and (2.5) with Lagrangian Multipliers $\lambda_0, \ldots, \lambda_3$ [5].

For an optimal solution $(\mathbf{x}^*, u^*, \lambda^*)$ of a problem (2.3), (2.4), and (2.5) the first necessary condition is given by [6]

$$H(\mathbf{x}^*, u^*, \lambda^*) \leq H(\mathbf{x}^*, u, \lambda^*)\,. \tag{2.13}$$

This means that for an optimal point $(\mathbf{x}^*, \lambda^*)$ on the trajectory, the optimal input $u = u^*$ has to minimize the Hamiltonian $H(\mathbf{x}^*, u, \lambda^*)$. Hence, the optimality condition (2.13) for the Hamiltonian (2.12) can be rewritten to

$$u^* = \begin{cases} u_{max}(\mathbf{x}^*), & \text{if } \lambda_3^* < 0 \\ u_{min}(\mathbf{x}^*), & \text{if } \lambda_3^* > 0 \\ \text{undefined}, & \text{if } \lambda_3^* = 0 \end{cases}, \tag{2.14}$$

assuming upper and lower bounds $u_{max}(\mathbf{x})$ and $u_{min}(\mathbf{x})$ for the input u can be defined according to the inequality conditions (2.5) and the system equation (2.4). Additional optimality conditions could give more information about the Lagrangian Multiplier λ_3, but it is known that pure state constraints lead to complex switching point analysis [5]. However, assuming no singular arcs [5] exist with $\lambda_3^* = 0$ on a non empty time interval $(\tau_1, \tau_2) \subset [t_0, t_e]$, the information given by the optimality condition (2.14) is already very useful because it means that u^* can only be either $u_{max}(\mathbf{x})$ or $u_{min}(\mathbf{x})$.

2.3.1 Discretization

In this section a discretization of the original problem (2.3), (2.4), and (2.5) is used to subsequently define a discrete online algorithm.

With sample times $t_k = \sum_{i=0}^{k} T_i, k = 0, \ldots, n$, a piecewise constant input $u(t) = u_k$, $t \in [t_k, t_{k+1}]$ and discrete states $\mathbf{x}_k = \mathbf{x}(t_k)$, a time discretization of the system equation (2.4) is given by [10]

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{b}_k u_k \,,$$
$$\mathbf{A}_k = \begin{bmatrix} 1 & T_k & \frac{T_k^2}{2} \\ 0 & 1 & T_k \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{b}_k = \begin{bmatrix} \frac{T_k^3}{6} \\ \frac{T_k^2}{2} \\ T_k \end{bmatrix}. \tag{2.15}$$

Inequality conditions (2.5) can be written as

$$\begin{matrix} \mathbf{g}(\mathbf{x}_k, u_k) \leq \mathbf{0} \\ \mathbf{h}(\mathbf{x}_k) \leq \mathbf{0} \end{matrix}, k = 0, \ldots, n \,, \tag{2.16}$$

which means that the original inequality conditions (2.5) must hold at discrete time steps t_k, $k = 0, \ldots, n$, but not in between. This simplification is common for many methods and in practice small enough sample times T_k deliver satisfying results.

2.3.2 Input Bounds

In this section a computation rule for the input bounds $u_{max}(\mathbf{x})$ and $u_{min}(\mathbf{x})$, defined in the beginning of Sect. 2.3, shall be found.

The set of allowed input values u_k is clearly limited by the mixed input-state constraints $\mathbf{g}(\mathbf{x}, u)$. In the discrete case pure state constraints $\mathbf{h}(\mathbf{x})$ can be taken into account by considering the next step $\mathbf{h}(\mathbf{x}_{k+1}(\mathbf{x}_k, u_k))$ where u_k automatically appears. To avoid time consuming computations for a lower and upper bound of a so defined set

$$S_k = \{u_k \in \mathbb{R},\, \mathbf{g}(\mathbf{x}_k, u_k) \leq \mathbf{0},\, \mathbf{h}(\mathbf{x}_{k+1}(\mathbf{x}_k, u_k)) \leq \mathbf{0}\} \,,$$

it is useful to introduce a u_k independent Taylor series estimation of the next state $\tilde{\mathbf{x}} = \mathbf{A}_k \mathbf{x}_k + \mathbf{b}_k 0$. Together with inequality conditions (2.6), and (2.7), this approximation lead to a set of linear inequality conditions for u_k, namely

$$\tilde{S}_k = \left\{u_k \in \mathbb{R},\ \boldsymbol{\beta}_{k,min} \leq \boldsymbol{\alpha}_k\, u_k \leq \boldsymbol{\beta}_{k,max}\right\}$$

with

$$\boldsymbol{\alpha}_k = \begin{bmatrix} \mathbf{q}' \\ \tilde{\mathbf{a}}_3\, T_s \\ \tilde{\mathbf{q}}' T_s \\ \tilde{\mathbf{q}}' \frac{T_s^2}{2} \end{bmatrix}, \quad \boldsymbol{\beta}_{k,\,\substack{min \\ max}} = \begin{bmatrix} \mathbf{j}_{\substack{min \\ max}} - (\mathbf{q}''' x_{2,k}^3 + \mathbf{q}'' 3 x_{2,k} x_{3,k}) \\ \boldsymbol{\tau}_{\substack{min \\ max}} - (\tilde{\mathbf{a}}_0 + \tilde{\mathbf{a}}_1\, \tilde{x}_2 + \tilde{\mathbf{a}}_2\, \tilde{x}_2^2 + \tilde{\mathbf{a}}_3\, x_{3,k}) \\ \ddot{\mathbf{q}}_{\substack{min \\ max}} - (\tilde{\mathbf{q}}'' x_{2,k}^2 + \tilde{\mathbf{q}}' x_{3,k}) \\ \dot{\mathbf{q}}_{\substack{min \\ max}} - \tilde{\mathbf{q}}' (x_{2,k} + x_{3,k} T_s) \end{bmatrix}$$

and $\tilde{(.)}$ denoting values computed with the estimated state $\tilde{\mathbf{x}}$. This directly gives a simply to compute estimation for $u_{k,max}$ and $u_{k,min}$ with $u_{k,min} \approx \inf\{\tilde{S}_k\}$ and $u_{k,max} \approx \sup\{\tilde{S}_k\}$.

2.3.3 Not Allowed States

In addition to the rule, how to compute $u_{k,max}$ and $u_{k,min}$ (see Sect. 2.3.2) it has to be clarified when to use $u_{k,max}$ and when $u_{k,min}$. When looking at the objective function, which can be rewritten as

$$J = \sum_{k=0}^{n-1} T_k = \sum_{k=0}^{n-1} \int_{x_{1,k}}^{x_{1,k+1}} \frac{1}{x_2} \mathrm{d}x_1, \tag{2.17}$$

choosing $u_{k,max}$ would be obviously the best to maximize the path speed x_2.

Additionally, it has to be considered that a sequence $\mathbf{x}_k, u_k \in S_k,\ k = j, j+1, \ldots,$ can potentially end in an empty set $S_k = \{\}$, which would mean that a not allowed state is reached and one of the inequality conditions (2.16) is violated.

States $\mathbf{x}_j$ belonging to a special subset $N \subset \mathbb{R}^3$ which can be continued with a sequence $u_k, k = j, j+1, \ldots,$ without reaching a not allowed state shall be called continuable states. Clearly, only continuable states should be considered but it is difficult to find a simple condition for $\mathbf{x}_j \in N$ due to the infinite horizon of the previous condition. However, if a finite trajectory $\mathbf{x}_k, u_k \in S_k, k = j, \ldots, m-1,$ which ends in a rest position $\mathbf{x}_m = [x_{1,m}\ \ 0\ \ 0]^{\mathrm{T}}$ can be found, it means that with $u_k = 0$, $\mathbf{x}_k = \mathbf{x}_m$ and $S_k = S_{m-1} \neq \{\}$ for all $k \geq m$ this rest position can be continued and it is a sufficient condition for a continuable state $\mathbf{x}_j$. A method for computing such short trajectories for testing if a state is continuable or not, is introduced in the subsequent section.

2.4 A Discrete Online Algorithm

In the previous Sect. 2.3 conditions for optimal system inputs u_k are found. Based on those conditions, the idea for the online algorithm is to test in each step $u_{k,max}$ as input and to check if the resulting next state is continuable. Therefore, in the subsequent section, so called test trajectories are introduced.

2.4.1 Test Trajectories

As described in Sect. 2.3.3, if a test trajectory $\xi = (\bar{\mathbf{x}}_0, \ldots, \bar{\mathbf{x}}_m, \bar{u}_0, \ldots, \bar{u}_m)$ with a given start state $\bar{\mathbf{x}}_0$ and a desired rest position $\bar{\mathbf{x}}_{m,d} = [\sigma_{r,d}\ \ 0\ \ 0]^{\mathrm{T}}$ can be found, it means that all states $\bar{\mathbf{x}}_j,\ j = 0, \ldots, m$ of such a test trajectory are continuable states. Choosing the input at the boundaries in three time intervals τ_1, τ_2 and τ_3 in the form

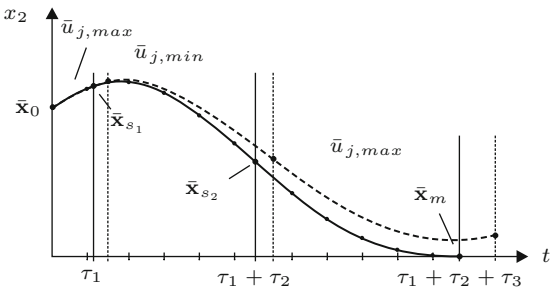

Fig. 2.2 Test trajectory for an initial state $\bar{\mathbf{x}}_0$. The solution of the 2PBVP is depicted with the solid line. Inappropriately chosen time intervals τ_1, τ_2 and τ_3 lead to a final state which is not a rest position (*dashed line*)

$$\bar{u}_j = \begin{cases} \bar{u}_{j,max} & \text{if } t_j \in [0, \tau_1) \\ \bar{u}_{j,min} & \text{if } t_j \in [\tau_1, \tau_1 + \tau_2) \\ \bar{u}_{j,max} & \text{if } t_j \in [\tau_1 + \tau_2, \tau_1 + \tau_2 + \tau_3) \end{cases} \tag{2.18}$$

result in a trajectory with three degrees of freedom τ_1, τ_2 and τ_3 (Fig. 2.2). Finding correct values for $\boldsymbol{\tau} = [\tau_1, \tau_2, \tau_3]^{\mathrm{T}}$ to end in the desired rest position $\bar{\mathbf{x}}_{m,d}$ is a two point boundary value problem (2PBVP). Therefore, known methods like single shooting [5] can be applied. In case of a free rest position $\sigma_{r,d}$ and fixed interval time τ_1, the 2PBVP reduces to a unknown $\bar{\boldsymbol{\tau}} = [\tau_2, \tau_3]^{\mathrm{T}}$.

It is clear that such a test trajectory does not exist for a not continuable state $\bar{\mathbf{x}}_0$ (see Sect. 2.3.3). But if it exists, all states $\bar{\mathbf{x}}_0, \ldots, \bar{\mathbf{x}}_m$ are known to be continuable. This test for continuable states is the basis for the online algorithm defined in the following section.

2.4.2 The Algorithm

Now the idea for an algorithm solving the optimal control problem (2.3), (2.4), and (2.5) is to use a maximum path jerk $u_{k,max}$ in each step as input for the discrete system (2.15) except the test trajectory computation fails. A detailed algorithm is described in the following.

1. Start with $\mathbf{x}_0 = [\,\sigma_0 \;\; 0 \;\; 0\,]^{\mathrm{T}}$ and $k = 0$.
2. Try to compute a test trajectory $\xi_k = (\bar{\mathbf{x}}_{k,0}, \ldots, \bar{\mathbf{x}}_{k,m_k}, \bar{u}_{k,0}, \ldots, \bar{u}_{k,m_k})$ with $\bar{\mathbf{x}}_{k,0} = \mathbf{x}_k$, fixed interval $\tau_1 = T_s$ and a free end position $\sigma_{r,d}$.
3. If the test trajectory ξ_k exceeds the end position σ_e, recompute the test trajectory with fixed end position $\bar{\mathbf{x}}_{k,m_k,d} = [\,\sigma_e \;\; 0 \;\; 0\,]^{\mathrm{T}}$ and free interval length τ_1. If successful, go to 7.
4. If ξ_k was not computed successfully, take the previous test trajectory but without the first state, thus

$$\xi_k = (\bar{\mathbf{x}}_{k-1,1}, \ldots, \bar{\mathbf{x}}_{k-1,m_{k-1}}, \bar{u}_{k-1,1}, \ldots, \bar{u}_{k-1,m_{k-1}}).$$

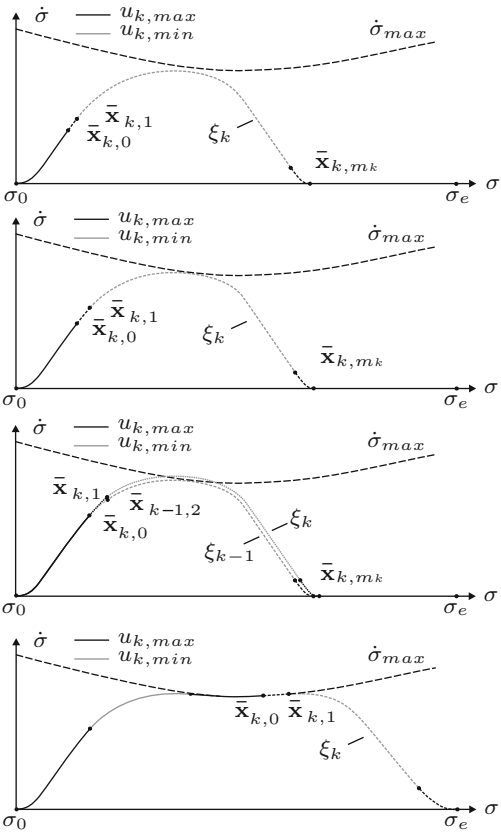

Fig. 2.3 Three consecutive iterations and the last step of the algorithm. The presence of dynamic limits is represented by the *dashed line* $\dot{\sigma}_{max}$. In the first two steps it can be seen that a test trajectory is computed successfully without violating the limit. Therefore, the state $\bar{\mathbf{x}}_{k,1}$ is continuable and can be outputted. In the third step this is not the case and a state from the previous test trajectory $\bar{\mathbf{x}}_{k-1,2}$ is taken instead. The figure on the bottom shows the last step in which the test trajectory reaches the end of the path

5. Output $\mathbf{x}_k = \bar{\mathbf{x}}_{k,0}$, $u_k = \bar{u}_{k,0}$ and choose $\mathbf{x}_{k+1} = \bar{\mathbf{x}}_{k,1}$.
6. Increase k by one and go to 2.
7. Output the whole test trajectory until the end of the path $\mathbf{x}_{k+j} = \bar{\mathbf{x}}_{k,j}, u_k = \bar{u}_{k,j}$ for $j = 0, \ldots, m_k$.

The evolution of the algorithm is also shown in Fig. 2.3. An iteration step of the algorithm always includes a test trajectory computation. Therefore, it is useful to know that, due to small iteration steps, time intervals τ_2 and τ_3 from a previous step $k-1$ can be assumed to be good starting values for the 2PBVP in the actual step k.

In each iteration step, a path jerk $u_{k,max}$ is used as long as test trajectories are computed successfully. If the computation fails, the algorithm stays on the last successfully computed test trajectory until a new test trajectory is found. Therefore, $u_{k,max}$ is used whenever possible, otherwise $u_{k,min}$ is taken as input for the discrete system (2.15). This means that the necessary condition for optimality (Sect. 2.3) is fulfilled.

2.4.3 Realtime Capability

The fact that each iteration step of the algorithm presented in Sect. 2.4.2 outputs a new path position $\sigma_k = x_{1,k}$ (and therefore also joint positions $\mathbf{q}(\sigma_k)$) makes this algorithm capable for realtime usage. Therefore, it is important to guarantee a maximum computation time for each step which is smaller than the chosen cycle time T_s. This is possible if the maximum computation time for a single test trajectory is limited by choosing a maximum trajectory length and a limitation of iterations for the 2PBVP. Exceeding these limits can be treated like a not successful test trajectory computation in the algorithm.

Experiments show, that for typical robotic applications (see Sect. 2.5) small enough sample times can be reached. Even smaller sample times, potentially needed for feedback controller, can be realized by simply resampling the result using (2.15), see [10]. Additionally, feed forward strategies [12] can be easily provided with set speeds, accelerations and torques resulting from (2.8), (2.9) and (2.11).

Due to the fact that the last point of each test trajectory $\bar{\mathbf{x}}_{k,m_k}$ is at the same time the most advanced point on the path, it is possible to change any information beyond that point like speed limits or the geometric path $\mathbf{q}(\sigma)$, $\sigma > \bar{x}_{k,m_k}$, without influencing the result. This means, that a geometric path can be prepared in the background while the algorithm is running and features like online speed adjustment (override) can be easily realized. In practice, it is also advantageous that in each time step a test trajectory is present which can bring the robot as fast as possible to a stop if required (e.g. emergency stop).

2.5 Example

In this section the presented algorithm is applied to a six-axis robot controlled by a standard industrial PC with a 1.4 GHz processor. For limiting motor torques, a full dynamic model with identified parameters is used. Velocity limits are taken from manufacturer's data sheet, acceleration limits are not needed (due to torque limitations) and jerk limits are derived from experimental results such that vibrations are extensively avoided.

A C-implementation of the algorithm for the standard controller reveals possible sample times smaller than $T_s = 2.4$ ms. This means that the realtime condition for the computation time of each iterations step is never violated.

In the following, a MATLAB-implementation of the presented algorithm is used to be able to compare the results with other state of the art (offline) methods. As an example, a geometric path is computed from an inverse kinematic transformation of a straight line in Cartesian space with fixed orientation of the tool center point (see Fig. 2.4). The resulting speed profile $\dot{\sigma}(t)$ and all consecutive test trajectories computed with a sample time $T_s = 10$ ms are depicted in Fig. 2.5.

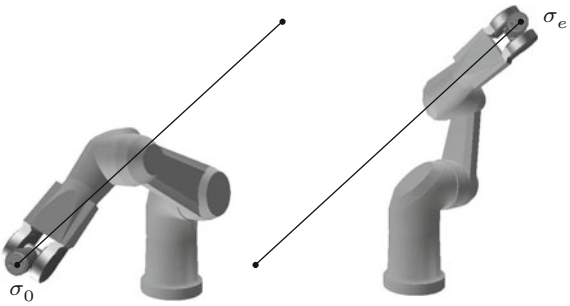

Fig. 2.4 A geometric path for a six-axis robot moving along a straight line

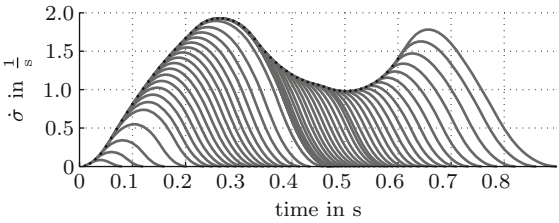

Fig. 2.5 Path speed $\dot{\sigma} = x_2$ and all consecutively generated test trajectories over time

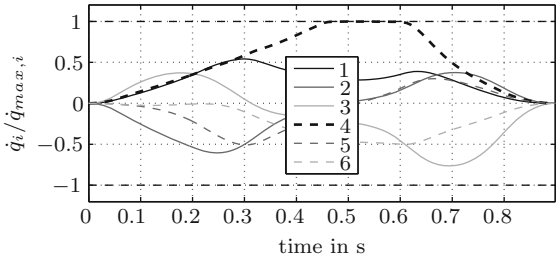

Fig. 2.6 Normalized axis velocities

Corresponding axis speeds, torques and jerks are shown in Figs. 2.6, 2.7, and 2.8. Comparing the results reveals that always one of the limits is active (Sect. 2.3). Now, two state of the art offline methods are applied to the same problem to proof the performance of the presented algorithm. For the first method a B-Spline approximation of the function $\sigma(t)$ is used similar to Ref. [18]. For minimizing the equidistantly chosen time intervals of the knot vector, this results in a parametrization of the original optimal control problem (2.3), (2.4), and (2.5). The second method, a modified implementation of the multiple shooting method as described in Ref. [13], is also a direct method. Time optimization is achieved

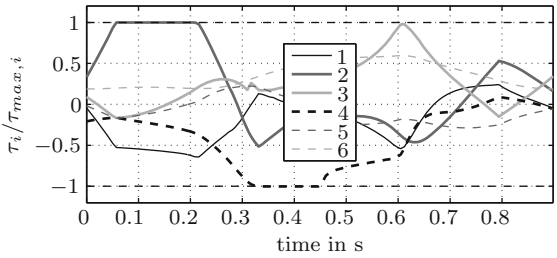

Fig. 2.7 Normalized motor torques

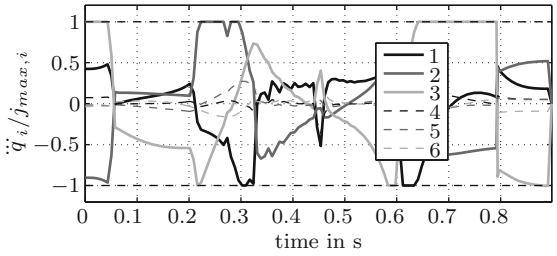

Fig. 2.8 Normalized axis jerks

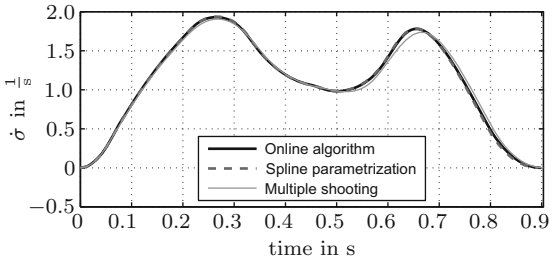

Fig. 2.9 Comparison of path speed $\dot{\sigma} = x_2$ over time for three solution methods: (1) the presented algorithm, (2) a direct method using a spline parametrization and (3) a multiple shooting method

by minimizing the time interval between each multiple shooting node. Both methods are implemented in MATLAB and an SQP method is used to solve the nonlinear optimization problems.

The comparison of the three methods in Fig. 2.9 shows only small deviations which can be explained with different parameterizations and discretizations. A detailed comparison is given in Table 2.1. Although, due to MATLAB implementations, computation times have to be interpreted carefully, the comparison shows that the presented method seems to be very efficient and delivers satisfying results in realtime. Furthermore, the offline methods tend to fail for long geometric paths,

Table 2.1 Comparison of optimization algorithms

	Online alg.	Spline	Mult. shoot.
Final time	0.8983 s	0.8935 s	9033 s
Num. points	92	70	70
CPU time	0.49 s	1.59 s	34.3 s
T_s	0.01 s	0.0129 s	0.0127 s

Online alg. is the presented algorithm, *Spline* is a method using a B-Spline parametrization and *Mult. shoot.* is a multiple shooting technique. *Final time* is the time t_e to be minimized. *Num. points* are the number of discrete points (chosen for spline and multiple shooting). *Iterations* is the number of test trajectories for the presented algorithm and the number of SQP iterations for the other methods respectively. *CPU time* is the computation time for the whole path in MATLAB. The sampling time T_s is chosen for the presented algorithm and a result for the other methods

whereas the presented algorithm is just running longer according to longer travel times.

2.6 Conclusion

The algorithm presented in this paper is an iterative online method for solving the problem of finding trajectories for the joint positions $\mathbf{q}(t)$ for a predefined geometric path. Therefore, time optimality and dynamic limits on joint speeds, accelerations, jerks and torques have to be considered. The method is directly derived from an optimality condition (Sect. 2.3) leading to lower and upper bounds for the path jerk (Sect. 2.3.2) as input for a discrete time system (Sect. 2.3.1). Further analysis on the existence of not allowed states show that a test for continuable states is necessary. Therefore, so called test trajectories are introduced, which form the basis of the presented algorithm (see Sect. 2.4.1). The main advantage of this method compared to other approaches is that a new path position (and therefore joint positions) can be outputted in each iteration step, which allows a realtime usage. An example shows that excellent results can be achieved, which is confirmed by a comparison with two other state of the art methods (Sect. 2.5).

For a future work deeper analysis on the optimality of the algorithm is planned.

Acknowledgements Support of the Austrian Center of Competence in Mechatronics (ACCM) is gratefully acknowledged.

References

1. Bazaz SA, Tondu B (1999) Minimum time on-line joint trajectory generator based on low order spline method for industrial manipulators. Robot Autom Syst 29(4):257–268
2. Bellman RE, Dreyfus SE (1962) Applied dynamic programming. Princeton University Press, Princeton

3. Bobrow JE, Dubowsky S, Gibson JS (1985) Time-optimal control of robotic manipulators along specified paths. Int J Robot Res 4:3–17
4. Bremer H (2008) Elastic multibody dynamics: a direct Ritz approach. Springer-Verlag GmbH, Linz
5. Bryson AE, Ho YC (1975) Applied optimal control: optimization, estimation, and control. Hemisphere, Washington, DC
6. Chachuat B (2007) Nonlinear and dynamic optimization: from theory to practice. http://lawww.epfl.ch/page4234.html
7. Chand S, Doty K (1985) Online polynomial trajectories for robot manipulators. Int J Robot Res 4:38–48, Summer
8. Constantinescu D (2000) Smooth and time-optimal trajectory planning for industrial manipulators along specified paths. J Robot Syst 17:233–249
9. Diehl M, Bock HG, Diedam H (2006) Fast direct multiple shooting algorithms for optimal robot control. Control 1:65–93
10. Franklin G, Powell JD, Workman ML (1998) Digital control of dynamic systems. Addison Wesley, Menlo Park
11. Hollerbach JM (1983) Dynamic scaling of manipulator trajectories. Am Control Conf 1983:752–756
12. Khalil W, Dombre E (20002) Modeling, identification & control of robots. Kogan Page Science, London
13. Leineweber DB, Bauer I, Bock HG, Schlöder JP (2003) An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part 1: theoretical aspects. Comput Chem Eng 27(2):157–166
14. Macfarlane S, Croft E (2003) Jerk-bounded robot trajectory planning-design for real-time applications. IEEE Trans Robot Autom 19(1):42–52
15. Pardo-Castellote G, Cannon RH (1996) Proximate time-optimal algorithm for on-line path parameterization and modification. In: Proceedings of the IEEE international conference robotics automation, Minneapolis, USA, April 1996
16. Pontryagin LS, Boltyanskii VG, Gamkrelidze RV, Mishchenko E (1962) The mathematical theory of optimal processes, International series of monographs in pure and applied mathematics. Interscience, New York
17. Shin K, McKay N (1985) Minimum-time control of robotic manipulators with geometric path constraints. IEEE Trans Autom Control 30(6):531–541
18. Verscheure D, Demeulenaere B, Swevers J, De Schutter J, Diehl M (2009) Time-optimal path tracking for robots: a convex optimization approach. IEEE Trans Autom Control 54 (10):2318–2327

http://www.springer.com/978-3-7091-1288-5

Multibody System Dynamics, Robotics and Control
Gattringer, H.; Gerstmayr, J. (Eds.)
2013, X, 314 p., Hardcover
ISBN: 978-3-7091-1288-5