Path-Accurate Online Trajectory Generation for Jerk-Limited Industrial Robots

Friedrich Lange[1] and Alin Albu-Schäffer[12]

***Abstract*—Standard industrial controllers of robot arms define constraints on the velocity, the acceleration, and the jerk and abort execution in case any of them is violated. In addition to satisfying these constraints, the presented method tries to generate a trajectory that is path-accurate, i.e., that exactly tracks the 6-dof shape of the desired path in axis space. Furthermore, the computed trajectory is as close as possible to the original robot program. This is reached by forward scaling and backtracking until a feasible trajectory is obtained. In contrast to previous work, blending of subsequent segments of the desired path is prevented by interpolation of the arc length instead of direct position interpolation. Because of its time-efficiency the algorithm can be applied in each sampling step, e.g. every 4 ms for a standard KUKA robot with RSI interface. Experimental results demonstrate the approach.**

***Index Terms*—Trajectory Generation, Motion Control of Manipulators, Industrial Robots**

I. Introduction

TRAJECTORY generation is required for all robotic motion. Commonly it is executed offline, optimizing both, the geometrical path that solves the task and the velocity profile of the tool center point (tcp) when tracking this path [1], [2], [3], [4], [5]. Such a computation may be time consuming. In contrast, for online methods the computing time is restricted, resulting in simpler methods. Online trajectory generation is required if the robot program is interpreted during its execution or if the desired path has changed, e.g. because a workpiece is misplaced.

For online trajectory generation, first the path is computed, i.e., a sequence of positions and orientations, or axis positions, and a rule on how to interpolate between them, e.g. by straight lines or circles. In order to be generic, in this paper we assume that the path is represented by a sequence of axis positions, where the distances between them are so small that the intermediate course does not matter. Thus we do not assume a differentiable or continuous path.

Second, a feasible trajectory is generated, which means that a velocity profile is added to the path. This step is treated in this paper. *Feasible* means that the constraints on the velocity, the acceleration, and the jerk are met. Most robot manufacturers define allowed ranges for the axis positions and their derivatives. For safety, tolerable load, moderate wear, and minimal excitation of oscillations, the robot motion is aborted whenever any of these constraints is violated. Otherwise the commanded position, i.e., the first sampling step of the generated trajectory, is sent to the position controllers of the robot.

Most trajectory generation methods minimize the execution time of the robot, as e.g. Refs. [6], [7], [8], [9], [10], [11], [12], [13], [14], denoted as time-optimal path parametrization (TOPP). In contrast, here the goal is a trajectory that, perhaps after a temporary slowdown, is synchronized with the original robot program, as in Ref. [15]. This is crucial when multiple robots or devices share the same working space.

In addition, it is assumed that the updated desired path has to be tracked exactly since otherwise a collision may happen or the contact force may increase excessively. This means that the generated motion has to be *path-accurate*. For safety not only the Cartesian positions are tracked but also the orientation. This is done by acting in the axis space with e.g. 6 degrees of freedom (dof). It should be mentioned that path accuracy is not always reachable without prediction. For example, high curvature of the path is only possible at low speed since otherwise acceleration limits are exceeded. Besides, without exact knowledge of the robot dynamics, it is not guaranteed that the desired path is really tracked. Therefore the compensation of deviations caused by the robot's dynamics, as in [7], [16], [17], [18], is left for future work.

The authors have previously presented a method that in most cases satisfies both, the robot constraints as well as the path accuracy [19]. However, in special cases there might be undesired blending of corners of the desired path, i.e., a large corner cut. This is a violation of path accuracy. Therefore, in this paper the method is modified by a different interpolation method denoted as arc length interpolation (ALI), to be distinguished from the direct position interpolation (DPI) of [19].

Instead of the 6-dof positions with DPI, ALI interpolates a scalar parameter s which is denoted as *arc length*. Its use for the generation of path-accurate trajectories is well known [2], [6], [7], [8], [11], [13], [16], [20], [21]. However, in these methods, in general, the kinematic constraints of the robot along the desired path are converted to constraints of s. Then the trajectory generation is solved as a one-dimensional problem whose solution is finally mapped to the axis space. The conversion of the constraints is not directly possible with a desired path that is given by the sampled positions. Thus in this paper another approach is presented that iteratively combines

Manuscript received: August, 12, 2015; Revised November, 16, 2015; Accepted November, 23, 2015.

This paper was recommended for publication by Editor Paolo Rocco upon evaluation of the Associate Editor and Reviewers' comments.

[1]Both authors are with the Robotics and Mechatronics Center, German Aerospace Center (DLR), Germany. {friedrich.lange, alin.albu-schaeffer}@dlr.de

[2]Alin Albu-Schäffer is also with the Department of Informatics, Technical University Munich, Germany.

Digital Object Identifier (DOI): please see at the top of this page

forward scaling and backtracking of the axis positions.

Forward scaling with ALI has already been presented for the dual problem of stopping along a given path [22]. That task is simpler, as it does without prediction and thus without backtracking. Now that method is extended to the generic case.

The paper is organized as follows: Next, the task is formulated. Section III outlines the method of [19]. Then Section IV replaces the interpolation which is used there. Finally Section V compares the different approaches in experiments using a KUKA industrial robot.

II. Notation and Problem Formulation

Notation: Derivatives are computed by backward differences, omitting the sampling time T_0. This results in a simple but unusual notation which is explained in the Appendix, similar to [15]. Vectors of all axes are denoted by bold face letters whereas single axes are in normal face with the index as last subscript, e.g. $q_{di}(k)$ with $i \in \{1, \cdots, 6\}$.

Task: The task is to modify a sequence of given *desired* positions $\mathbf{q}_d(k) = (q_{d1}(k), \cdots, q_{d6}(k))^T$ of sampling steps k in such a way that at the current time step k the resulting *commanded* positions $\mathbf{q}_c(k) = (q_{c1}(k), \cdots, q_{c6}(k))^T$ preferably satisfy the following conditions:

1) Feasibility: For all axes $i \in \{1, ..., 6\}$, the velocity $v_{ci}(k) = q_{ci}(k) - q_{ci}(k-1)$, the acceleration $a_{ci}(k) = v_{ci}(k) - v_{ci}(k-1)$, and the jerk $j_{ci}(k) = a_{ci}(k) - a_{ci}(k-1)$ satisfy the constraints

$$|v_{ci}(k)| = |q_{ci}(k) - q_{ci}(k-1)| \le \bar{v}_i \tag{1}$$

$$|a_{ci}(k)| = |q_{ci}(k) - 2q_{ci}(k-1) + q_{ci}(k-2)| \le \bar{a}_i \tag{2}$$

$$|j_{ci}(k)| = |q_{ci}(k) - 3q_{ci}(k-1) + 3q_{ci}(k-2) - q_{ci}(k-3)| \le \bar{j}_i, \tag{3}$$

where $\pm\bar{v}_i$, $\pm\bar{a}_i$, and $\pm\bar{j}_i$ are the (symmetric) limits for the velocity, the acceleration, and the jerk.

2) Path Accuracy: If a position $\mathbf{q}_c(k)$ is on the desired path, the next position $\mathbf{q}_c(k+1)$ will be on the desired path as well. The desired path is defined by the sampled positions $\mathbf{q}_d(\cdot)$ and the straight lines between them, i.e.,

$$\mathbf{q}_c(k) = (1-\beta)\mathbf{q}_d(k+\kappa-1) + \beta\mathbf{q}_d(k+\kappa) \tag{4}$$

with $\kappa \in \mathcal{Z}$ and $0 \le \beta < 1$.

3) Synchronization: Even if $\mathbf{q}_d(k)$ is not feasible, there is at least a single $\kappa_1 \in \mathcal{N}$ for which the commanded trajectory reaches the desired trajectory, i.e.,

$$\mathbf{q}_c(k+\kappa_1) = \mathbf{q}_d(k+\kappa_1). \tag{5}$$

This denotes that $\mathbf{q}_c$ hangs $\mathbf{q}_d$ at time step $k+\kappa_1$.

Equation (5) implies that the sequence $\mathbf{q}_d(k)$ does not only represent the desired path but also the time steps k in which the respective positions are desired.

III. Iterative Trajectory Generation

A. Trajectory Generation by Forward Scaling

1) Velocity Constraint: At the current time step k, first of all it is checked whether the desired position can directly be taken as command $\mathbf{q}_c(k) = \mathbf{q}_d(k)$. Otherwise the velocity is scaled. This applies if the desired position $\mathbf{q}_d(k)$ violates (1) in at least one axis i, i.e., $|v_{dci}(k)| > \bar{v}_i$ with $v_{dci}(k) = q_{di}(k) - q_{ci}(k-1)$. Scaling the velocity between time step $(k-1)$ and k probably results in a path-accurate trajectory since the path is not modified but the time profile of its execution.

The modification is executed in two steps. First, the scaling factor α is computed from all limited components as

$$\alpha = \min_{i,\ |v_{dci}(k)|>\bar{v}_i} (\bar{v}_i/|v_{dci}(k)|). \tag{6}$$

This computation is always possible and results in $0 < \alpha < 1$. Then the velocity of all axes is scaled using

$$\mathbf{q}_c(k) = \mathbf{q}_c(k-1) + \alpha\mathbf{v}_{dc}(k). \tag{7}$$

2) Acceleration Constraint: The velocity is scaled as well if the desired position violates (2) in at least one axis i, i.e., $|a_{dci}(k)| > \bar{a}_i$ with $a_{dci}(k) = q_{di}(k) - 2q_{ci}(k-1) + q_{ci}(k-2)$. (7) gives

$$|q_{ci}(k-1) + \alpha v_{dci}(k) - 2q_{ci}(k-1) + q_{ci}(k-2)| \le \bar{a}_i \tag{8}$$

and hence the scaling factor is

$$\alpha = \min_{i,\ |a_{dci}(k)|>\bar{a}_i} (\frac{\pm\bar{a}_i + v_{ci}(k-1)}{v_{dci}(k)}). \tag{9}$$

The sign of $\pm\bar{a}_i$ is positive if $a_{dci}(k) > \bar{a}_i$, it is negative if $a_{dci}(k) < -\bar{a}_i$. Otherwise axis i does not contribute to the computation of α since the component i of (2) is satisfied.

If the result is $0 \le \alpha \le 1$, (7) satisfies (2) in all axes. Then the modification is path-accurate.

Unfortunately, a scaling in an axis $i_1 \in \{1, \cdots, 6\}$ may cause a limitation in another axis $i_2 \in \{1, \cdots, 6\}$, since the elements of $\mathbf{q}_c(k)$, which are modified according to (7), have to be used in (2) for verification. Simply speaking, if the velocity has to be reduced because of a big desired acceleration in axis i_1, the reduction of the velocity of axis i_2 may exceed the maximum deceleration. Therefore all scaling is repeated iteratively, where the $q_{di}(k)$ for the computations in the next iteration step are taken from the $q_{ci}(k)$ of the previous iteration step. The iterative execution of (7) and (9) ends when a solution with $0 \le \alpha \le 1$ is found that satisfies the constraint (2) for all axes i, or when α is not within the allowed range. The number of steps of this iteration is not greater than the number of axes. The modification of the desired trajectory will not be path-accurate if a solution with $0 \le \alpha \le 1$ cannot be found.

3) Jerk Constraint: Scaling is required also if (3) is exceeded by at least a single axis, i.e., $|j_{dci}(k)| > \bar{j}_i$ with $j_{dci}(k) = q_{di}(k) - 3q_{ci}(k-1) + 3q_{ci}(k-2) - q_{ci}(k-3)$. As with the acceleration, it is first tried to modify the desired trajectory in a path-accurate way, i.e., by scaling the velocity. (7) gives

$$\begin{aligned}|q_{ci}(k-1) &+ \alpha v_{dci}(k)\\ &-3q_{ci}(k-1) + 3q_{ci}(k-2) - q_{ci}(k-3)| \le \bar{j}_i.\end{aligned} \tag{10}$$

Thus the scaling factor

$$\alpha = \min_{i,\ |j_{dci}(k)|>\bar{j}_i} (\frac{\pm\bar{j}_i + v_{ci}(k-1) + a_{ci}(k-1)}{v_{dci}(k)}) \tag{11}$$

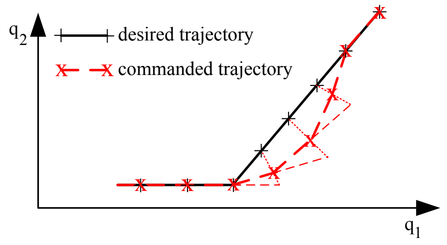

Fig. 1. Modification of a 2 dof path by direct scaling (DS) the acceleration at the corner of the desired path in order to comply with acceleration limits (not path-accurate). The markers denote the sampled positions.

can be computed and inserted in (7). As with the acceleration, this has to be repeated until a solution with $0 \le \alpha \le 1$ is found that satisfies all constraints.

Similar to (9), in (11) the maximum jerk $\pm\bar{j}_i$ has a positive sign if $j_{dci}(k) > \bar{j}_i$ and it has a negative sign if $j_{dci}(k) < -\bar{j}_i$. Otherwise axis i does not contribute to the determination of the scaling factor.

B. Direct Scaling as a Last Resort

If no feasible axis positions $\mathbf{q}_c(k)$ exist from Section III-A, an alternative solution is computed. It satisfies all kinematic constraints mentioned in Section III-A, but the resulting trajectory is not path-accurate. This algorithm is denoted as direct scaling (DS) of the acceleration and the jerk.

If an acceleration limit (2) is exceeded, the scaling is done for all axes using

$$\mathbf{q}_c(k) = \mathbf{q}_c(k-1) + \mathbf{v}_c(k-1) + \alpha\mathbf{a}_{dc}(k) \tag{12}$$

(cf. (33) with $i = 1$), where the scaling factor α is computed from the constrained axes by

$$\alpha = \min_{i,\ |a_{dci}(k)|>\bar{a}_i} (\bar{a}_i/|a_{dci}(k)|). \tag{13}$$

This modification of the desired position is always possible, since $0 < \alpha < 1$, provided that an acceleration limit is violated. Scaling the acceleration results in a position between that one which would be reached with zero acceleration and the desired position which is not reachable, see Fig. 1.

If a jerk limit (3) is exceeded, the scaling is done for all axes using

$$\mathbf{q}_c(k) = \mathbf{q}_c(k-1) + \mathbf{v}_c(k-1) + \mathbf{a}_c(k-1) + \alpha\mathbf{j}_{dc}(k) \tag{14}$$

(cf. (35) with $i = 1$), and

$$\alpha = \min_{i,\ |j_{dci}(k)|>\bar{j}_i} (\bar{j}_i/|j_{dci}(k)|) \tag{15}$$

computed from the constrained axes only. This results in $0 < \alpha < 1$, provided that a jerk constraint is violated before the direct scaling (DS).

C. Inhibition of Overshooting by Backtracking

Direct scaling and the corresponding path inaccuracies can be avoided in the case that the desired trajectory is known in advance. Then the forward scaling of Section III-A can be done predictively. If no feasible solution is found for a time step $k+\kappa$, a modification of preceding positions is possible, e.g. $\mathbf{q}_c(k+\kappa-1)$, instead of computing $\mathbf{q}_c(k+\kappa)$ by direct scaling. This generally inhibits overshooting after reaching the desired trajectory with a too high velocity or acceleration. The consideration of preceding time-steps is called backtracking. In contrast to direct scaling it is path-accurate.

For brevity, $k+\kappa$ is denoted as k' in the following.

Constraints of the velocity according to (1) need no backtracking, since (6) and (7) always give a feasible commanded position.

Constraints on the acceleration according to (2) can be resolved by modifying $q_{di}(k')$ and $q_{ci}(k'-1)$, in order to result in a feasible $a_{ci}(k')$. This is done using

$$q_{bai}(k') = q_{ci}(k'-1) + \alpha(q_{di}(k') - q_{ci}(k'-1)) \tag{16}$$

$$\begin{aligned} q_{bai}(k'-1) &= q_{ci}(k'-2) \\ &\quad + (1+\alpha)(q_{ci}(k'-1) - q_{ci}(k'-2))/2. \end{aligned} \tag{17}$$

$q_{di}(k')$ is the desired position or a position that has been computed previously by backtracking. $q_{ci}(k'-1)$ has been computed by forward scaling or previous iteration steps of backtracking.

Equation (17) has been chosen such that with α from (13), (16)-(17) result in $|a_{bai}(k')| = |q_{bai}(k') - 2q_{bai}(k'-1) + q_{ci}(k'-2)| \le \bar{a}_i$ with $|a_{bai}(k')| = \bar{a}_i$ for the most constrained axis.

Similarly, constraints on the jerk according to (3) can be resolved by modifying $q_{di}(k')$, $q_{ci}(k'-1)$, and $q_{ci}(k'-2)$, in order to result in a feasible $j_{ci}(k')$. This is done by

$$q_{bji}(k') = q_{ci}(k'-1) + \alpha(q_{di}(k') - q_{ci}(k'-1)) \tag{18}$$

$$\begin{aligned} q_{bji}(k'-1) &= q_{ci}(k'-2) \\ &\quad + (1+2\alpha)(q_{ci}(k'-1) - q_{ci}(k'-2))/3 \end{aligned} \tag{19}$$

$$\begin{aligned} q_{bji}(k'-2) &= q_{ci}(k'-3) \\ &\quad + (2+\alpha)(q_{ci}(k'-2) - q_{ci}(k'-3))/3, \end{aligned} \tag{20}$$

where (19) and (20) have been chosen such that with α from (15), (18)-(20) result in $|j_{bji}(k')| = |q_{bji}(k') - 3q_{bji}(k'-1) + 3q_{bji}(k'-2) - q_{ci}(k'-3)| \le \bar{j}_i$ with $|j_{bji}(k')| = \bar{j}_i$ for the most constrained axis.

D. Iterative Procedure

In order to result in feasible and path-accurate trajectories, forward scaling and backtracking are executed iteratively. The procedure is as follows:

- Try $\mathbf{q}_c = \mathbf{q}_d$ and check the constraints (1), (2), and (3) predictively, i.e., for $k+\kappa$ instead of k, with $\kappa = 0, \cdots, \bar{\kappa}$, where $\bar{\kappa}$ denotes the number of future time-steps for which the desired trajectory $\mathbf{q}_d(k+\kappa)$ is available.[1]
  If needed, do forward scaling according to Sect. III-A. If this is successful, continue with the next value of κ.
- Otherwise do backtracking according to Sect. III-C at time-step $k+\kappa$. Then continue at the first modified sampling step, i.e., time-step $k+\kappa-1$ or $k+\kappa-2$, depending on the kind of backtracking. For this, all modified values

[1] $\bar{\kappa}$ is at least in the order of magnitude of the number of time steps that are needed to stop down from the current velocity $\mathbf{v}(k)$. A bigger value does not compromise the procedure but needs additional computing power.

q_{bai} or q_{bji} from backtracking are transferred to the q_{di} of the respective time steps, since these are required for a subsequent or future check of the constraints or a forward scaling.

The iteration ends when

- $\bar{\kappa}$ is reached (success),
- $k{+}\kappa{-}1 < 0$ respectively $k{+}\kappa{-}2 < 0$ is reached (failure),
- or when the maximum computing time is exceeded (failure).

In the case of a failure, a feasible solution is reached nevertheless by direct scaling according to Section III-B. This is required e.g. if $\bar{\kappa}$ is too small or if the desired trajectory is recomputed during high speed motion.

E. Discussion

The method presented so far always results in a feasible trajectory. But it is not ensured that a *path-accurate* trajectory is found within the given computing time. This can be improved heuristically by multiplying the scaling factor α in forward scaling by $c_f \approx 0.9$, provided that $\alpha < 1$.

The reached path accuracy may be inferior to what is expected, especially in the following cases:

1) If the desired trajectory has been lost, subsequent positions are not guaranteed to reach the desired path. (This case is not comprised in the above definition of path accuracy.)
2) The position resulting from scaling $\mathbf{v}_{dc}(k)$ lies always on the straight line between $\mathbf{q}_c(k-1)$ and $\mathbf{q}_d(k)$. So it may be off the path if the latter is curved and these two points are far from each other. Fig. 2 shows an example.
3) But even if the interpolation is done within single steps of $\mathbf{q}_d$, the desired path can be lost by repeated forward and backward interpolation during an iteration. Finally the commanded path is significantly smoothed with respect to the desired path.

Section IV provides a solution for the last two points, as shown in Fig. 2.

IV. Arc Length Interpolation (ALI)

The desired path is given as sampled positions $\mathbf{q}_d(k+\kappa)$, which are connected by straight lines. In order to better preserve the desired path, a scalar path coordinate $s(k+\kappa)$ is computed, also denoted as arc length. When the scalar trajectory has been finally determined as a sequence $s(k+\kappa)$ with $\kappa = 0, \cdots, \bar{\kappa}$, the spatial trajectory $\mathbf{q}_c(k+\kappa)$ can be computed from $\mathbf{q}_d(k+\kappa)$ by a single interpolation. This is denoted as arc length interpolation (ALI).

The result is illustrated in Fig. 2. It shows that all sampling points of the commanded trajectory are on the desired path. The blending appears only between adjacent sampling points of the final trajectory. This cannot be prevented.

Arc length interpolation may be applied always when an interpolation is done in the preceding sections. This concerns (7), (16)-(17), and (18)-(20). Then, if a constraint of an axis i is violated, the following steps are executed:

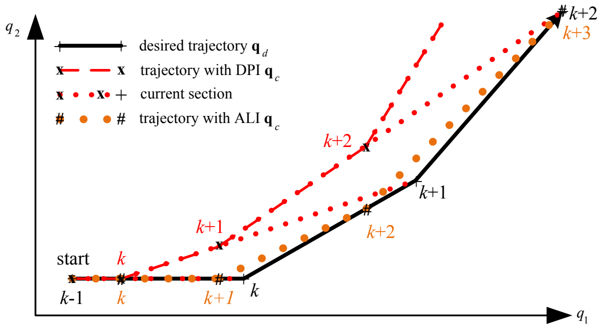
q_2
desired trajectory $\mathbf{q}_d$
trajectory with DPI $\mathbf{q}_c$
current section
trajectory with ALI $\mathbf{q}_c$
start
k
k-1
k
k+1
k
k+1
k+2
k+2
k+1
k+2
k+3
q_1

Fig. 2. Example with a path error due to a delay during nonlinear motion: After starting in time step k 1, $\mathbf{q}_d(k)$ (black +) cannot be reached because of the acceleration limit. Instead, $\mathbf{q}_c(k)$ is computed. In the next step, DPI computes $\mathbf{q}_c(k+1)$ (red x) by scaling the line from $\mathbf{q}_c(k)$ to $\mathbf{q}_d(k+1)$ (black +). In the same way, $\mathbf{q}_c(k+2)$ lies between $\mathbf{q}_c(k+1)$ and $\mathbf{q}_d(k+2)$. With ALI, $\mathbf{q}_c(k)$ is computed as with DPI, using (27) for $q_{fa1}(k)$ and (31). Then (29) for $q_{fa1}(k+1)$ and (32) computes $\mathbf{q}_c(k+1)$ (orange #) between the previous executed position $\mathbf{q}_c(k)$ and the next desired position, which in this case is $\mathbf{q}_d(k)$ (black +) and not $\mathbf{q}_d(k+1)$. Then $\mathbf{q}_c(k+2)$ is computed by (27) for $q_{fa1}(k+2)$ and/or $q_{fa2}(k+2)$ and (31) between two desired positions $\mathbf{q}_d(k)$ and $\mathbf{q}_d(k+1)$. $\mathbf{q}_d(k+2)$ has no influence. Similarly, $\mathbf{q}_d(k+3)$ is determined. Thus there is no path error at the sampled positions $\mathbf{q}_c$. For simplicity, jerk limits are not considered in this example.

1) The reachable axis position $q_{\#*i}(k')$ is computed that satisfies the limit, with $\# \in \{f, b\}$ and $* \in \{v, a, j\}$. Other than in Section III, (7) is replaced by

$$q_{fvi}(k') = q_{ci}(k'-1) \pm \bar{v}_i \quad (21)$$
$$q_{fai}(k') = q_{ci}(k'-1) + v_{ci}(k'-1) \pm \bar{a}_i \quad (22)$$
$$q_{fji}(k') = q_{ci}(k'-1) + v_{ci}(k'-1) + a_{ci}(k'-1) \pm \bar{j}_i, \quad (23)$$

if the respective constraint is violated by $q_{di}(k')$. In this way the singularity of $v_{dci}(k') = 0$ does not appear. The sign of the limit $\pm\bar{v}_i$, $\pm\bar{a}_i$, or $\pm\bar{j}_i$ is such that the modification with respect to $q_{di}(k')$ is minimal.
Equations (16)-(17) and (18)-(20) are adopted without change, but the α are not the minimal values from (13) and (15) respectively, but individually

$$\alpha = \bar{a}_i / |a_{dci}(k)| \quad (24)$$

or

$$\alpha = \bar{j}_i / |j_{dci}(k)|. \quad (25)$$

2) A scalar parameter $s_{*i}(k')$ is computed from this axis position $q_{\#*i}(k')$, as explained in Section IV-A. For simplicity, s is defined here by the elapsed time of the desired trajectory, meaning that a position $\mathbf{q}_d(k')$ has the value $s = k'$. The exact definition is given in Section IV-A. It is a linear interpolation between subsequent sampling points of the desired trajectory or between the previously executed position and the next sampled desired position.
3) It is checked whether $s_{*i}(k') < s(k')$. In this case the new value is taken as $s(k')$. So with ALI, the minimum of the computed $s_{*i}(k')$ is selected.

$$s(k') = \min_i(s_{vi}(k'), s_{ai}(k'), s_{ji}(k')), \quad (26)$$

This corresponds to the minimum of the factor α with DPI, according to (6), (9), or (11).

4) The resulting position $\mathbf{q}_c(k')$ is computed. $\mathbf{q}_c(k')$ is not used for a recursive computation as with DPI, since all forward scaling and backtracking steps are done on the basis of s and not of previously computed $\mathbf{q}_c$. So, except for the final computation of $\mathbf{q}_c(k)$, the $\mathbf{q}_c(k')$ serve only for checking the constraints in further steps of 1), using $\mathbf{q}_c(k')$ instead of $\mathbf{q}_d(k')$.

A. Computation of the Arc Length and the Resulting Position

The difference between ALI and DPI is that interpolation of $s(\cdot)$ is not simply done by $s(k) = s(k-1)+\alpha(s(k)-s(k-1))$, similar to (7), but by an equation that computes $s(k)$ from the reachable positions $q_{\#*i}(k)$.

In order to comprise forward scaling as well as backtracking, the task is formulated to interpolate $q_{\#*i}(k'-k_1)$ in the interval between $q_{ci}(k'-k_2)$ and $q_{di}(k'-k_1)$. Then for forward scaling, $k_2 = 1$, whereas backtracking the acceleration or the jerk needs $k_2 = 2$ and $k_2 = 3$ respectively. $k_1 = 0$ is used for (21)-(23), (16), and (18). Equations (17) and (19) correspond to $k_1 = 1$, whereas $k_1 = 2$ is taken for (20).

$s_{*i}(k'-k_1)$ is computed from the $q_{\#*i}(k'-k_1)$ using

$$\begin{aligned} s_{*i}(k'-k_1) \;&=\; \underline{s}_{*i}(k'-k_1) \\ &+ \frac{q_{\#*i}(k'-k_1) - q_{di}(\underline{s}_{*i}(k'-k_1))}{q_{di}(\underline{s}_{*i}(k'-k_1)+1) - q_{di}(\underline{s}_{*i}(k'-k_1))}, \end{aligned} \tag{27}$$

where $\underline{s}$ is the largest integer not greater than s.

Equation (27) is computed by trial and error, trying $\underline{s}_{*i}(k'-k_1) = \underline{s}(k'-k_1), \underline{s}(k'-k_1)-1, \cdots, \underline{s}(k'-k_2)+1$, until a $s_{*i}(k'-k_1)$ fits to the interval

$$\underline{s}_{*i}(k'-k_1) \leq s_{*i}(k'-k_1) < \min(s(k'-k_1), \underline{s}_{*i}(k'-k_1)+1). \tag{28}$$

In the beginning, all values $s(k')$ are initialized by k'. Before each computation of (27), $s(k'-k_1)$ is reduced to $s(k'-k_1+1)$, whenever the latter is lower.

If no solution is found by (27) and (28),

$$\begin{aligned} s_{*i}(k'-k_1) \;&=\; s(k'-k_2) \\ &+ \frac{(q_{\#*i}(k'-k_1) - q_{ci}(k'-k_2)) \cdot (\underline{s}(k'-k_2)+1-s(k'-k_2))}{q_{di}(\underline{s}(k'-k_2)+1) - q_{ci}(k'-k_2)} \end{aligned} \tag{29}$$

may give a feasible $s_{*i}(k'-k_1)$, with

$$s(k'-k_2) \leq s_{*i}(k'-k_1) < \min(s(k'-k_1), \underline{s}(k'-k_2)+1). \tag{30}$$

At most $k'-k_1-\underline{s}(k'-k_2)+1$ equations (27) or (29) have to be tested for computing $s_{*i}(k'-k_1)$. In case multiple solutions of $s_{*i}(k'-k_1)$ exist, the biggest one is taken, since it represents the smallest change with respect to the desired trajectory, which corresponds to $s_{*i}(k'-k_1) = k'-k_1$.

The axis positions of the arc length $s(k'-k_1)$ can be computed using

$$\begin{aligned} \mathbf{q}_c(k'-k_1) \;&=\; \mathbf{q}_d(\underline{s}(k'-k_1)) + \\ (s(k'-k_1)-\underline{s}(k'-k_1))&\cdot(\mathbf{q}_d(\underline{s}(k'-k_1)+1)-\mathbf{q}_d(\underline{s}(k'-k_1))), \end{aligned} \tag{31}$$

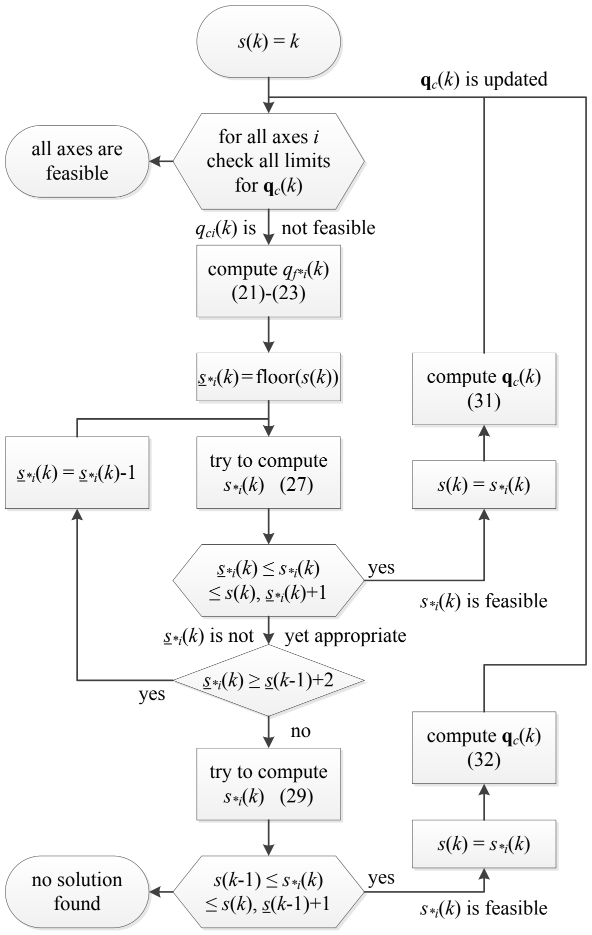
$s(k) = k$
$\mathbf{q}_c(k)$ is updated
for all axes i check all limits for $\mathbf{q}_c(k)$
all axes are feasible
$q_{ci}(k)$ is not feasible
compute $q_{f*i}(k)$ (21)-(23)
$\underline{s}_{*i}(k)=\text{floor}(s(k))$
compute $\mathbf{q}_c(k)$ (31)
$\underline{s}_{*i}(k) = \underline{s}_{*i}(k)$-1
try to compute $s_{*i}(k)$ (27)
$s(k) = s_{*i}(k)$
$\underline{s}_{*i}(k) \leq s_{*i}(k) \leq s(k), \underline{s}_{*i}(k)+1$
yes
$s_{*i}(k)$ is feasible
$\underline{s}_{*i}(k)$ is not yet appropriate
$\underline{s}_{*i}(k) \geq \underline{s}(k\text{-}1)+2$
yes
no
compute $\mathbf{q}_c(k)$ (32)
try to compute $s_{*i}(k)$ (29)
$s(k) = s_{*i}(k)$
no solution found
$s(k\text{-}1) \leq s_{*i}(k) \leq s(k), \underline{s}(k\text{-}1)+1$
yes
$s_{*i}(k)$ is feasible

Fig. 3. Flow chart of ALI during forward scaling.

or

$$\begin{aligned} \mathbf{q}_c(k'-k_1) \;&=\; \mathbf{q}_c(k'-k_2) + \\ &\frac{(s(k'-k_1)-s(k'-k_2))\cdot(\mathbf{q}_d(\underline{s}(k'-k_2)+1)-\mathbf{q}_c(k'-k_2))}{\underline{s}(k'-k_2)+1-s(k'-k_2)}, \end{aligned} \tag{32}$$

if $s(k) < \underline{s}(k'-k_2)+1$. This results in $q_c(k'-k_1) = q_{\#*i}(k'-k_1)$ for the most limiting constraint $*i$.

The simplified procedure with $k' = k$, $k_1 = 0$, and $k_2 = 1$ is illustrated in Fig. 3.

B. Discussion

Other than with previous work, ALI is not a complete trajectory generation method but an add-on to the approach of Section III, improving the path accuracy.

The main features of ALI are as follows:

- Whereas DPI always interpolates between $\mathbf{q}_c(k'-1)$ and $\mathbf{q}_d(k')$, ALI selects the appropriate segment and interpolates there. So no corner cut exists, as in 2) of the list of Section III-E. The only blending appears between subsequent sampled positions $\mathbf{q}_c(k'-1)$ and $\mathbf{q}_c(k')$. ALI and DPI are identical if there is only a single segment of $\mathbf{q}_d(\cdot)$ between $s(k'-1)$ and k'. The different cases are illustrated in Fig. 2.

- For integral $s(k')$, $\mathbf{q}_c(k') = \mathbf{q}_d(s(k'))$. Otherwise $\mathbf{q}_c(k')$ is on the straight line between neighboring points of $\mathbf{q}_d$ or the previously executed position $\mathbf{q}_c(k' - k_2)$ and $\mathbf{q}_d(\underline{s}(k' - k_2) + 1)$.
- All iterative scaling only modifies the scalar arc length $s(\cdot)$. Thus the original desired trajectory is not affected. Consequently, no smoothing is present, as in 3) of the list in Section III. The final commanded trajectory $\mathbf{q}_c(\cdot)$ is computed from the $s(\cdot)$ and the $\mathbf{q}_d(\cdot)$ and not iteratively updated from preliminary $\mathbf{q}_c(\cdot)$.
- In contrast to Section III-C, backtracking with ALI does not always result in a feasible trajectory. This is explained in the following and a solution is found.

Backtracking with DPI according to (16) - (17) or (18) - (20) guarantees that $\mathbf{q}_{ba}(k')$ or $\mathbf{q}_{bj}(k')$ are feasible. In particular, $\alpha = 0$ results in $\mathbf{a}_{bac}(k') = \mathbf{q}_{ba}(k') - 2\mathbf{q}_{ba}(k'-1) + \mathbf{q}_c(k'-2) = 0$ or $\mathbf{j}_{bjc}(k') = \mathbf{q}_{bj}(k') - 3\mathbf{q}_{bj}(k'-1) + 3\mathbf{q}_{bj}(k'-2) - \mathbf{q}_c(k'-3) = 0$, since the individual axes are all backtracked similarly. But the same factor α may give different values $s_{*i}(k'-1)$ (see Fig. 4) or $s_{ji}(k'-2)$. Consequently, the updated $s(k'-1)$ and $s(k'-2)$ may result in $a_{ci}(k') \neq a_{baci}(k')$ or $j_{ci}(k') \neq j_{bjci}(k')$.

Therefore, backtracking with ALI is repeated, as forward scaling is repeated in Section III whenever scaling because of an axis i_1 has caused a new limitation in an axis i_2. Convergence can be speeded up by multiplying α with a decreasing factor, e.g. $c_b = 1 - 0.03l$, where $l \in \mathcal{N}$ is the iteration index. In this way, feasible backtracking is usually achieved rapidly.

If, nevertheless, repeated backtracking does not converge to a feasible set of $s(k')$, $s(k'-1)$, and $s(k'-2)$, a solution is enforced by several scalings with $\alpha = 0$, until the result is feasible. This is true at the latest for $\underline{s}(k') = \underline{s}(k'-1) = \underline{s}(k'-2) = \underline{s}(k'-3)$, since then the interpolation is within a single interval, as in Section III. The number of such backtracking steps is in the order of magnitude of $s(k') - s(k'-3)$.

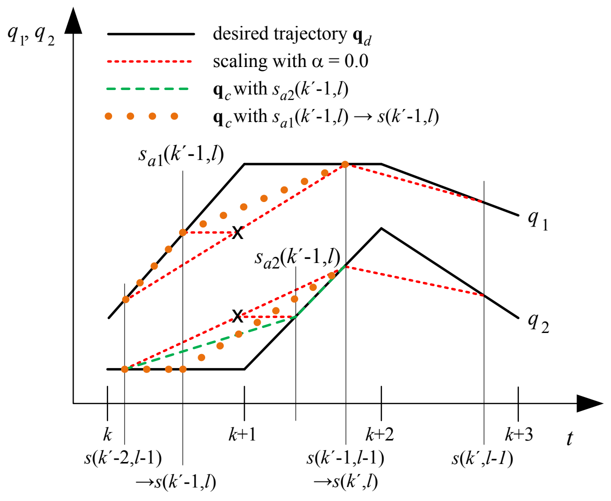

Fig. 4. Example of backtracking in an iteration step l in which the $s(k'$ k_1, l $1)$ are not given by the sampling steps anymore. By backtracking the acceleration with $\alpha = 0$, $\mathbf{q}_{bai}(k', l)$ according to (16) and thus $s(k', l)$ and $\mathbf{q}_c(k', l)$ are adopted from $s(k'$ $1, l$ $1)$ and $\mathbf{q}_c(k'$ $1, l$ $1)$ of the previous iteration step l 1. But there is no unique $s_{ai}(k'$ $1, l)$ from the two $\mathbf{q}_{bai}(k'$ $1, l)$ according to (17), marked by x. The lesser $s_{ai}(k'$ $1, l)$ is adopted as updated $s(k'$ $1, l)$ and results in the updated $\mathbf{q}_c(k'$ $1, l)$. DPI would result in updated positions marked by x, thus leaving the path.

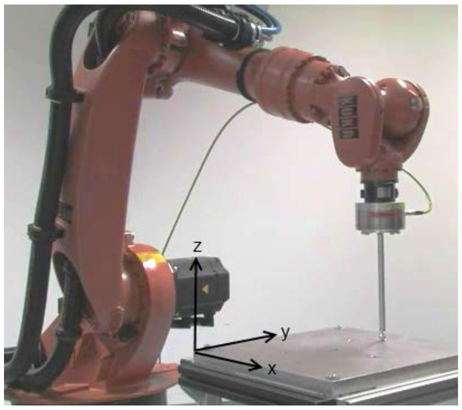

Fig. 5. KUKA robot with force sensor and pin which is in contact.

V. Experiments

In the first experiment the robot tool moves horizontally, overlayed by a vertical motion until a reaction force is measured when touching the table (see setup in Fig. 5). This happens in time step 3184 in Fig. 6, much earlier than expected. Then the future desired trajectory is recomputed such that the desired force will be executed with the original desired horizontal motion. This approach is explained in detail in [23]. It results in a step function since the actual position at which the force is sensed is delayed with respect to the so far desired trajectory.

Fig. 6 shows the experimental results of the first three axes of a KUKA KR16 robot that is controlled at 250 Hz via RSI Ethernet from an external computer. It displays that the desired velocity of axis 1 remains constant whereas the desired trajectories of axes 2 and 3 are significantly changed after the impact. Fig. 7 shows the paths in the plane of axes 2 and 3. Table I displays the used limits of velocities, accelerations and jerks.

When the desired trajectory is changed (because of a sensed contact), all tested methods succeed in a feasible trajectory. In the first steps they are even identical. The steps 3184 to 3186 are not path-accurate, because the axes 2, 3, and 5 cannot be accelerated fast enough.

TABLE I
ASSUMED LIMITS OF THE INDIVIDUAL AXES IN RAD/(SAMPLING STEP), RAD/(SAMPLING STEP)^2, AND RAD/(SAMPLING STEP)^3 (BEFORE FILTERING).

i	$\bar{v}_i$	$\bar{a}_i$	$\bar{j}_i$
1	0.014	0.000074	0.000061
2	0.014	0.000037	0.000030
3	0.014	0.000085	0.000069
4	0.029	0.000250	0.000204
5	0.030	0.000252	0.000206
6	0.055	0.000450	0.000368

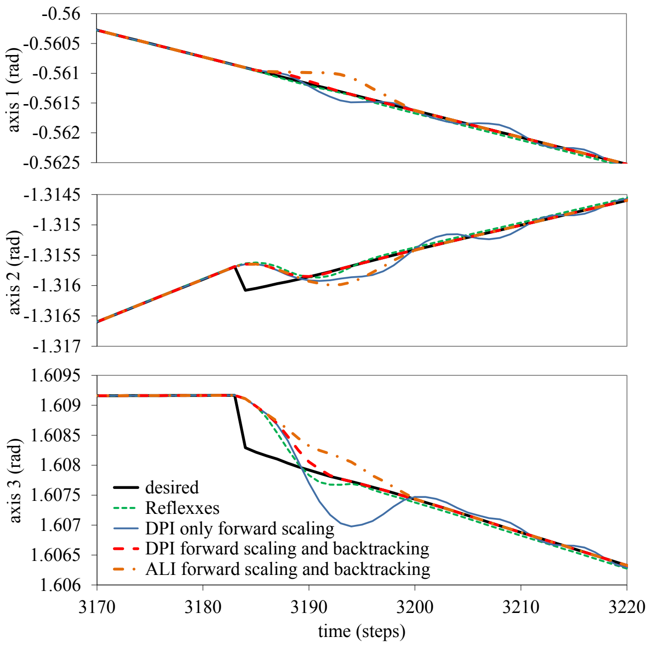

Fig. 6. Desired and commanded trajectories using different methods.

Then the method without backtracking, i.e., DPI with only forward scaling according to Sections III-A and III-B, executes maximum acceleration until the desired trajectory is reached. Thereafter, as far as the jerk limit allows, the deceleration is maximum. In this way a poorly damped oscillation around the desired path is executed. The robot motion may even become unstable if the limits of the jerk are very restrictive. So this “classical” approach is not suitable. The resulting path is shown in Fig. 7.

In contrast, with DPI with backtracking there is no overshooting since the acceleration is reduced in time. Tracing the experiment discloses that time steps 3184 to 3186 require direct scaling (DS). In step 3187 the following steps of forward scaling (f) and backtracking (b) are done in order to comply with the constraints in brackets: f($a_2(3187)$), f($a_1(3188)$, $a_2(3188)$, $a_3(3188)$), f($a_3(3189)$), b($a_2(3190)$, $a_3(3190)$, $a_5(3190)$), b($a_2(3189)$), b($j_2(3189)$, $j_3(3189)$), f($a_2(3191)$), b($a_3(3191)$), f($a_1(3192)$, $a_2(3192)$), f($a_2(3193)$), and b($j_2(3194)$, $j_3(3194)$). Then the modified trajectory is feasible until time step $3208 = 3187 + \bar{\kappa}$ and $\mathbf{q}_c(3187)$ is executed. Similarly, backtracking is performed in each of the next sampling steps.

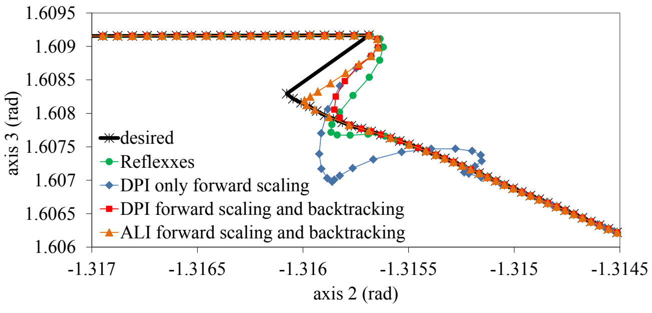

Fig. 7. Desired and commanded paths using a force sensor. Each symbol marks a sampled position.

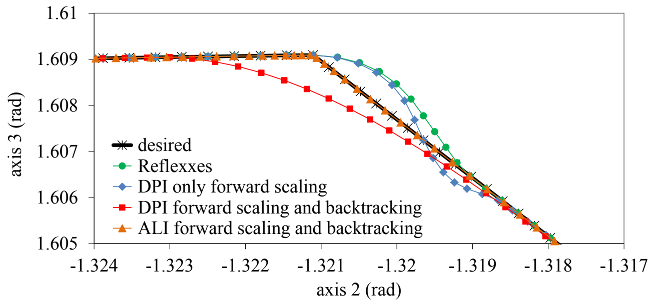

Fig. 8. Desired and commanded paths using a predictive sensor (5 times as fast as Fig. 7). Each symbol marks a sampled position.

The result of the Reflexxes Motion Library [24] is similar to this. It is obtained by computing trajectories from $\mathbf{q}_c(k-1)$, $\mathbf{v}_c(k-1)$, and $\mathbf{a}_c(k-1)$ to $\mathbf{q}_d(k+\kappa)$ and $\mathbf{v}_d(k+\kappa)$. This is done by tentatively calling the function *RMLPosition* for $\kappa = 0, \cdots, \bar{\kappa}$ until the target is reached within at most κ time steps.[2] The intermediate time steps $k, \cdots, k+\kappa-1$ are not considered, i.e., path accuracy does not matter.

Forward scaling and backtracking with ALI decelerates significantly and thus reaches the desired trajectory later than with the other methods (Fig. 6), but at an earlier position of the desired path (Fig. 7).

In a second experiment the task is repeated 5 times faster, where, instead of the force-torque sensor, a distance sensor is simulated. This sensor can measure the table position in advance, thus predict a changed desired trajectory that keeps a minimum distance. The predictive approach decelerates early enough, such that there is no overshooting (Fig. 8). This cannot be exploited when using any of the two methods without backtracking.

In this experiment, backtracking with DPI shows substantial blending at the vertex, caused by up to 30 iteration steps. In contrast, ALI reproduces the desired path exactly, which is only possible by a stricter deceleration, resulting in a delay $k - s(k)$ of up to 9.5 sampling steps, using not more than 56 iteration steps for each trajectory.

The experiment verifies that, in combination with a predictive sensor with little noise, the presented approach with ALI can preserve the shape of the desired path according to the definition of path accuracy in Section II.

The video attachment gives an impression of both experiments. However, the different methods cannot be distinguished there.

VI. Conclusion

The paper presents a trajectory generator that in each sampling step continues the so far commanded robot motion on the currently desired path, complying with constraints on the velocity, the acceleration, and the jerk, and finally synchronizing with the desired trajectory. The method is applicable to industrial robots with standard robot programs, which in this way are online-modified locally.

[2]Note that this is different to the intended usage. In addition, the used backward computation of the derivatives is not adequate for the approach of the library. Therefore the apparent overshooting and offset when using the Reflexxes Motion Library are no deficiency of that library.

Trajectory generation is done by forward scaling and backtracking, using either direct position interpolation (DPI) or arc length interpolation (ALI), where the latter results in better path accuracy. If no other solution is found, direct scaling (DS) always gives a feasible position command.

The method can be extended to comply with other constraints, e.g. torque constraints or limitations on Cartesian level. This can be treated similarly, but with time-variant limits.

Further extensions are possible in order to speed up the convergence. This will be the focus of future work.

Appendix

Without loss of generality, the sampling time T_0 is expressed in sampling steps instead of seconds. This results in $T_0 = 1$. Then it can be omitted in the equations, meaning that for rotational axes all quantities are dimensionless. For example the acceleration $\mathbf{a}$ is expressed in radians per squared sampling steps instead of rad/s^2.

In addition, as in [19], all derivatives are computed by backward differences, e.g. $\mathbf{a}(k) = \mathbf{v}(k) - \mathbf{v}(k-1) = \mathbf{q}(k) - 2\mathbf{q}(k-1) + \mathbf{q}(k-2)$. With $\mathbf{a}(k+1) = \cdots = \mathbf{a}(k+i)$ this results in

$$\mathbf{q}(k+i) = \mathbf{q}(k) + i\mathbf{v}(k) + i(i+1)/2\ \mathbf{a}(k+i) \tag{33}$$

and

$$\mathbf{v}(k+i) = \mathbf{v}(k) + i\mathbf{a}(k+i). \tag{34}$$

For small i, this representation differs with respect to the common equation $\mathbf{q}(k+i) = \mathbf{q}(k) + i\mathbf{v}(k) + i^2/2\ \mathbf{a}$. For $i = 1$ the difference is a factor of 2 for the acceleration. But this representation is consistent with the backward differences.

Similar to the acceleration, the jerk is defined by $\mathbf{j}(k) = \mathbf{a}(k) - \mathbf{a}(k-1) = \mathbf{q}(k) - 3\mathbf{q}(k-1) + 3\mathbf{q}(k-2) - \mathbf{q}(k-3)$. With $\mathbf{j}(k+1) = \cdots = \mathbf{j}(k+i)$ this results in

$$\begin{aligned}\mathbf{q}(k+i) &= \mathbf{q}(k) + i\mathbf{v}(k) + i(i+1)/2\ \mathbf{a}(k) \\ &\quad + i(i+1)(i+2)/6\ \mathbf{j}(k+i)\end{aligned} \tag{35}$$

and

$$\mathbf{v}(k+i) = \mathbf{v}(k) + i\mathbf{a}(k) + i(i+1)/2\ \mathbf{j}(k+i), \tag{36}$$

thus differing up to a factor of 6 from $\mathbf{q}(k+i) = \mathbf{q}(k) + i\mathbf{v}(k) + i^2/2\ \mathbf{a} + i^3/6\ \mathbf{j}$. This discrepancy is solved by adapting the limits $\bar{\mathbf{a}}$ and $\bar{\mathbf{j}}$.

References

[1] K. Erkorkmaz and Y. Altintas. High speed CNC system design. Part I: jerk limited trajectory generation and quintic spline interpolation. *Int. J. of Machine Tools and Manufacture*, 41(9):1323–1345, July 2001.

[2] F. Debrouwere, W. Van Loock, G. Pipeleers, Q. Tran Dinh, M. Diehl, J. De Schutter, and J. Swevers. Time-optimal path following for robots with trajectory jerk constraints using sequential convex programming. In *Proc. 2013 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1908–1913, Karlsruhe, Germany, May 2013.

[3] M. Lawitzky, M. Kimmel, P. Ritzer, and S. Hirche. Trajectory generation under the least action principle for physical human-robot cooperation. In *Proc. 2013 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 4270–4275, Karlsruhe, Germany, May 2013.

[4] T. Kunz and M. Stilman. Probalistically complete kinodynamic planning for robot manipulators with acceleration limits. In *Proc. 2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3713–3819, Chicago, IL, USA, Sep 2014.

[5] J. Schultz and T. D. Murphey. Real-time trajectory generation for a planar crane using discrete mechanics. In *Workshop on "Real-time Motion Generation & Control - Constraint-based Robot Programming" at the 2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Chicago, IL, USA, Sep 2014. "http://cs.stanford.edu/people/tkr/iros2014/proceedings.html".

[6] Y. Bestaoui. On-line motion generation with velocity and acceleration constraints. *Robotics and Autonomous Systems*, 5:279–288, 3 1989.

[7] O. Dahl and L. Nielsen. Torque limited path following by on-line trajectory time scaling. *IEEE Trans. on Robotics and Automation*, 6(5):554–561, Oct 1990.

[8] Z. Shiller and H.-H Lu. Computation of path constrained time optimal motions with dynamic singularities. *ASME Journal of Dynamic Systems, Measurements, and Control*, 114:34–40, 1 1992.

[9] R. Haschke, E. Weitnauer, and H. Ritter. On-line planning of time-optimal, jerk-limited trajectories. In *Proc. 2008 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3248–3253, Nice, France, Sep 2008.

[10] X. Broquère, D. Sidobre, and I. Herrera-Aguilar. Soft motion trajectory planner for service manipulator robot. In *Proc. 2008 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2808–2813, Nice, France, Sep 2008.

[11] J. Mattmüller and D. Gisler. Calculating a near time-optimal jerk-constrained trajectory along a specified smooth path. *Int. J. Adv. Manuf. Technol.*, 45:1007–1016, 2009.

[12] T. Kröger and F. M. Wahl. Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events. *IEEE Trans. on Robotics*, 26(1):94–111, Feb 2010.

[13] M. H. Ghasemi, N. Kashiri, and M. Dardel. Near time-optimal control of redundant manipulatros along a specified path with jerk constraint. *Advanced Robotics*, 25:2319–2339, 2011.

[14] Q. Pham. A general, fast, and robust implementation of the time-optimal path parameterization algorithm. *IEEE Trans. on Robotics*, 30(6):1533–1540, Dec. 2014.

[15] C. Guarino Lo Bianco and F. Ghilardelli. A discrete-time filter for the generation of signals with asymmetric and variable bounds on velocity, acceleration, and jerks. *IEEE Trans. on Industrial Electronics*, 61(8):4115–4125, Aug 2014.

[16] C. Guarino Lo Bianco and O. Gerelli. Online trajectory scaling for manipulators subject to high-order kinematic and dynamic constraints. *IEEE Trans. on Robotics*, 27(6):1144–1152, Dec 2011.

[17] S. Pchelkin, A. Shiriaev, A. Robertsson, and L. Freidovich. Integrated time-optimal trajectory planning and control design for industrial robot manipulator. In *Proc. 2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2521–2526, Tokyo, Japan, Nov. 2013.

[18] R. Gill, D. Kulić, and C. Nielsen. Robust path following for robot manipulators. In *Proc. 2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3412–3418, Tokyo, Japan, Nov. 2013.

[19] F. Lange and M. Suppa. Predictive path-accurate scaling of a sensor-based defined trajectory. In *Proc. 2014 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 754–759, Hong Kong, China, May/June 2014.

[20] A. Amthor, S. Zschäk, C. Ament, A. Lorenz, and J. Werner. Method and device for real-time-capable forth-order path planning for generating continuous target trajectories free of jerk jumps. International patent WO002010136586A1, 2010.

[21] C. Guarino Lo Bianco and F. Ghilardelli. Real-time planner in the operational space for the automatic handling of kinematic constraints. *IEEE Trans. on Automation Science and Engineering*, 11(3):730–739, July 2014.

[22] F. Lange and M. Suppa. Trajectory generation for immediate path-accurate stopping of industrial robots. In *Proc. 2015 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2021–2026, Seattle, WA, USA, May 2015.

[23] F. Lange, W. Bertleff, and M. Suppa. Force and trajectory control of industrial robots in stiff contact. In *Proc. 2013 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2912–2919, Karlsruhe, Germany, May 2013.

[24] Reflexxes. http://www.reflexxes.ws/, last visited 2015.