Jerk-limited Real-time Trajectory Generation with Arbitrary Target States

Lars Berscheid and Torsten Kröger
Karlsruhe Institute of Technology (KIT), Germany
{lars.berscheid, torsten}@kit.edu

Abstract—We present Ruckig, an algorithm for online trajectory generation (OTG) respecting third-order constraints and complete kinematic target states. Given any initial state of a system with multiple degrees of freedom (DoFs), Ruckig calculates a time-optimal trajectory to an arbitrary target state defined by its position, velocity, and acceleration limited by velocity, acceleration, and jerk constraints. The proposed algorithm and implementation allows three contributions: (1) To the best of our knowledge, we derive the first time-optimal OTG algorithm for arbitrary, multi-dimensional target states, in particular including non-zero target acceleration. (2) This is the first open-source\textsuperscript{1} prototype of time-optimal OTG with limited jerk and complete time synchronization for multiple DoFs. (3) Ruckig allows for directional velocity and acceleration limits, enabling robots to better use their dynamical resources. We evaluate the robustness and real-time capability of the proposed algorithm on a test suite with over 1 000 000 000 random trajectories as well as in real-world applications.

I. INTRODUCTION

Modern robots are supposed to operate in and manipulate their unknown and non-deterministic environments. It is therefore not sufficient to plan actions beforehand; instead robots need to be able to react to novel sensor input on the fly. Then, a new trajectory needs to be generated in a real-time manner, allowing the robot to adapt the task execution within the scope of its dynamical resources. The trajectory representation is particularly important, as it serves as an interface between the (more abstract) task planning and (lower level) motion planning. A common representation is waypoint-based: The task execution is given as a single or list of waypoints with defined kinematic state. Then, online trajectory generation (OTG) will calculate a trajectory to the new waypoint target considering the robots constraints. Commonly, second-order (namely velocity and acceleration) constraints take the dynamical resources into account. However, third-order constraints (an additional jerk limit) are desirable to reduce mechanical stress, wear and tear, and the robots overall cost over lifetime. In fact, we find that modern industrial robots, e.g. by Franka Emika, monitor the jerk in the internal controller and terminate in case of acceleration discontinuities.

\textsuperscript{1}Ruckig is a C++ library published under the permissive MIT license at https://github.com/patior/ruckig

In this work, we propose a novel algorithm for OTG named Ruckig that is significantly simpler than related approaches. While guaranteeing a solution even for systems with multiple DoFs, Ruckig enables three contributions to the field of OTG: First, target waypoints can be defined not only by their position and velocity, but by their complete kinematic state including acceleration. This improves the practical usability of OTG in dynamic tasks. Second, Ruckig is the first open-source and freely available OTG implementation with constrained jerk. Third, this work introduces directional velocity and acceleration limits. This makes it easier to exploit the full dynamic capabilities of the robot. Moreover, we argue that this is useful for human robot interaction: To ensure human safety, a velocity limit towards the human needs to be met, while simultaneously moving away from the human to avoid possible contact.

In the following paper, we formalize the problem of waypoint-based OTG, derive the proposed algorithm and share details about the implementation. We evaluate the robustness and real-time performance on a set of randomly generated trajectories. Finally, we show real-world applications highlighting the proposed contributions.
II. RELATED WORK

As trajectory generation is an essential part within robotics, it has already seen decades of research of both offline as well as online (real-time) approaches.

a) Offline Trajectory Generation: More than 50 years ago, Kahn et Roth were the first to use optimal linear control to generate near time-optimal trajectories \[6\]. Modern approaches commonly split the motion planning into: (1) a calculation of a geometric path, e.g. as a result of a task-specific (e.g. goal-finding or collision-avoidance) algorithm, and (2) a general time-parametrization of this path regarding the dynamic resources. Regarding this second step, Kunz et al. proposed a time-optimal time-parametrization with velocity and acceleration constraints \[9\]. In recent years, Pham et al. have improved the calculation time for second-order constraints significantly \[12\]. Regarding jerk constraints, Lange et al. introduced an approach for path-following with constrained jerk \[10\].

b) Online Trajectory Generation: For OTG, the path and time-parametrization are oftentimes combined into a single step. This simplification is commonly done alongside a waypoint-based formulation of the problem. In contrast to most offline time-parametrization algorithms, OTG only considers kinematic constraints of each independent DoF. This difference is also known as kinematic vs. dynamic time-optimality.

Macfarlane et al. introduced time-optimal and jerk-constrained OTG for a single DoF \[11\]. For multiple DoFs, Broquere et al. presented a jerk-limited OTG however with zero initial acceleration \[3\]. Haschke et al. derived an algorithm to reach a given position with jerk-constraints from an arbitrary initial state with non-zero acceleration \[5\]. Kröger et al. classified OTG algorithms depending on the highest derivative order of both the non-zero target state and constraint \[8\]. Subsequently, Kröger introduced Reflexxes, an open-source Type II and proprietary Type IV implementation \[7\]. The latter allows to reach waypoints with non-zero velocity from any arbitrary state. In this context, the proposed Ruckig algorithm and library is - to the best of our knowledge - the first Type V and the first open-source Type IV implementation.

While considering complete third-order states, Ahn et al. presented a non-time-optimal algorithm that does not consider velocity, acceleration, or jerk limits directly \[11\]. In the field of quadrotor flight, Beul et al. introduced opt_control, an open-source jerk-constrained OTG algorithm however without considering complete time synchronization for arbitrary input states \[2\]. Moreover, there is significant interest in the field of safety and human robot collaboration (HRC). Most prominent, Haddadin et al. investigated reaction motions in the context of OTG for safe human-robot interaction \[4\].

III. PROBLEM DEFINITION

Let \(\vec{x}\) be the state of a kinematic particle with \(N\) DoFs. The state \(x_i(t)\) of DoF \(i \in \{1, \ldots, N\}\) at time \(t\) is defined by the position \(p_i\) and its partial derivatives of up to third order

\[
x_i = \left(p_i, v_i := \frac{\partial p_i}{\partial t}, a_i := \frac{\partial^2 p_i}{\partial t^2}, j_i := \frac{\partial^3 p_i}{\partial t^3}\right)
\]

named velocity \(v_i\), acceleration \(a_i\), and jerk \(j_i\). We consider the kinematic time-optimality, otherwise the total instead of partial derivatives would be required. Given an initial state \(\vec{x}_0\) and a target (final) state \(\vec{x}_f\), we seek the time-optimal trajectory \(\vec{x}^*(t)\) defined by

\[
\vec{x}^*(t) = \arg \min_{\vec{x}(t)} T_f, \quad \vec{x}(0) = \vec{x}_0, \quad \vec{x}(T_f) = \vec{x}_f
\]

satisfying the velocity, acceleration and jerk constraints

\[
v_{i,min} \leq v_i(t) \leq v_{i,max} \quad \text{a}_{i,min} \leq a_i(t) \leq a_{i,max} \quad j_{i,min} \leq j_i(t) \leq j_{i,max}
\]

for all times \(t \in [0, T_f]\) and DoFs \(i\). \(T_f\) is called the trajectory duration. If no vector-notation is given furthermore, we calculate each DoF independently. Note that we consider both the initial as well as target state to be complete with possibly all derivatives non-zero. For simplicity, we assume \(j_{min} = -j_{max}\) furthermore, but keep the directional acceleration and velocity limits. Moreover, not every kinematic target state is physically possible. We define an upper bound of the allowed target acceleration by

\[
a_f \leq \sqrt{2j_{max} \max(|v_{max} - v_f|, |v_{min} - v_f|)},
\]

because an acceleration target requires a minimum velocity interval to be reached with constrained jerk.

IV. ALGORITHM

We divide the OTG problem into six subsequent steps. Following a short overview, each step is explained in detail in its own subsection.

A. An optional brake pre-trajectory is calculated if the initial state \(x_0\) exceeds or will inevitably exceed the kinematic limits \(v_{min}, v_{max}, a_{max}\) or \(a_{min}\). In this case, recovering to a safe kinematic state is the most urgent task.

B. In Step 1: Extremal times, all possible profiles that utilize the full dynamic resources of the robot are calculated for each DoF \(i\) independently. We call this the set of valid extremal profiles. The duration of the fastest profile is called \(T_{i,min}\).

C. The target \(\vec{x}_f\) should be reached at the same time point \(T_f\) by each DoF \(i\). Therefore, some DoFs might need to slow down. In general, not every trajectory duration \(T > \max_i(T_{i,min})\) is possible,
as there might be a limited number of blocked intervals for the duration. We derive these intervals based on the set of valid extremal profiles.

D. The minimum trajectory duration $T_f$ is the fastest duration that is not blocked by any DoF. This duration corresponds to a limiting profile as well as a limiting DoF and is included in the set of valid profiles.

E. In Step 2: Time synchronization, we calculate a profile for every DoF that reaches its target $x_f$ at the given trajectory duration $T_f$.

F. Finally, the new state at a given time $t$ on the trajectory can be calculated.

A time-optimal trajectory will be limited by a single DoF $l$ which uses its entire dynamical resources at all times $t$. Therefore, this DoF will use a bang-bang-like jerk profile with $j_l(t) \in \{-j_{max}, 0, j_{max}\}$. We call such profiles extremal. In particular, the total duration $T$ of a profile can be changed by its underlying jerk profile in infinitesimal steps. If and only if the profile is extremal, the duration is bounded on one side.

More generally, we formulate each trajectory as a sequence of constant jerk values $j_k = s_k j_f$ with corresponding non-negative time steps $t_k \geq 0$. Let $s_k \in \{-1, 0, 1\}$ be the jerk sign and $j_f > 0$ the jerk value constant throughout the profile. An extremal profile with third-order constraints results in a linear acceleration, quadratic velocity, and in a cubic polynomial for the position. Fig. 1 shows an illustrative example of an extremal profile with seven steps $k$.

A. Brake Pre-trajectory

If the initial state $x_0$ exceeds or will exceed the acceleration or velocity limits, a so-called pre-trajectory is introduced to brake the system below its respective limits. Due to limited jerk, cases exists that will inevitably brake the velocity constraints at a later point in time. Let $T_{br}$ be the duration of the brake pre-trajectory or zero if none is required. In comparison to the rest of the algorithm, this step works in the velocity domain ignoring any position values. We introduce a decision tree (Fig. 2) depending on $a_0, v_0, a_{max}, a_{min}, v_{max},$ and $v_{min}$ that calculates the fastest profile to reach the limits. It can be seen by distinction of cases that a resulting profile includes up to two time-steps $t_{lb0}$ and $t_{lb1}$ with corresponding jerk $j_{lb0} \in \{-j_{max}, j_{max}\}$ and $j_{lb1} = 0$. The second step with zero jerk might be necessary, as no new constraints should get broken.

B. Step 1: Extremal Times

We want to find all extremal profiles that reach the target state $x_f$ for all DoFs independently.

Lemma IV.1. A velocity limit might only be reached once in an extremal profile.

At a velocity limit, the profile has zero acceleration and zero jerk. The profile can always be decelerated by reducing the velocity plateau. We show by contradiction: If two velocity limits would be in the same direction, the profile could be accelerated by removing the intermediate deceleration and extending the maximal velocity. If the velocity limits would be in opposite directions, the profile could be accelerated by removing the distance traveled from the shorter direction from the other one. As the duration can be shortened and extended, it cannot be an extremal profile.

Lemma IV.2. There are only up to two acceleration limits in an extremal profile.

Otherwise, a third acceleration peak exists resulting in one direction reached at least two times. Then, the profile could be speed up by shifting the acceleration from the later reached peak to the prior one. The profile could be decelerated with the inverse approach.

Therefore, only up to three limits can be reached in total (as shown in Fig. 1). First, an acceleration limit called ACC0, second a velocity limit called VEL, and third an acceleration limit called ACC1. Introducing optional steps of constant jerk before, after, and between the limits lead to a maximal number of seven steps $k$ with jerk $j_k = s_k j_f$ and corresponding duration $t_k$ (Table I). We denote the sign of the non-zero jerk $s_k$ as either $\uparrow = +1$ or $\downarrow = -1$. A redundant step is encoded with zero duration $t_k = 0$.

Four non-zero jerk steps result in 16 possible com-

---

**Fig. 2.** A part of the decision tree (given $a_0 > a_{max}$) for calculating an optional brake pre-trajectory. If required, we calculate the time-optimal profile to transfer the system to a safe kinematic state. The pre-trajectory is determined by up to two steps of constant jerk $j_0$ and $j_1$ and their respective duration $t_0$ and $t_1$.

---

**TABLE I**

<table>
<thead>
<tr>
<th>Step $k$</th>
<th>Jerk Sign $s_k$</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>$\uparrow \ \ \ \downarrow$</td>
<td>-</td>
</tr>
<tr>
<td>$t_2$</td>
<td>0</td>
<td>ACC0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>$\uparrow \ \ \ \downarrow$</td>
<td>-</td>
</tr>
<tr>
<td>$t_4$</td>
<td>0</td>
<td>VEL</td>
</tr>
<tr>
<td>$t_5$</td>
<td>$\uparrow \ \ \ \downarrow$</td>
<td>-</td>
</tr>
<tr>
<td>$t_6$</td>
<td>0</td>
<td>ACC1</td>
</tr>
<tr>
<td>$t_7$</td>
<td>$\uparrow \ \ \ \downarrow$</td>
<td>-</td>
</tr>
</tbody>
</table>
bimations. However, only four unique profiles meet the kinematic constraints and are non-redundant: ↑↓↓, ↑↓↑, ↓↑↓, and ↓↓↓. As the overall problem is invariant to a sign change in \( f_m \) and exchanging \( v_{min} \leftrightarrow v_{max} \) and \( a_{min} \leftrightarrow a_{max} \), the set of distinct jerk profiles can be simplified further to ↑↓↑ and ↑↓↓ profiles. Then, the first jerk sign corresponds to the UP or DOWN direction.

The final list of profile types include every combination of the above three limits and final two jerk profile types. Table II lists all 16 distinct profile types for a single direction. In step 1, only 12 profiles are possible as a ↑↓↓ profile with a positive acceleration after the velocity limit is not valid.

Mathematically, each profile type maps the initial state \( x_0 \), the target state \( x_f \), and the given limits

\[
S_1 : (p_0, p_f; v_0, v_f, a_0, a_f; v_{max}, a_{max}, a_{min}, f_{max}) \rightarrow \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}
\]

to corresponding times \( t_1 \) to \( t_7 \). Given 3 equations for position, velocity and acceleration and 7 variables, 4 additional conditions need to be introduced. Three conditions are set by their limits or a zero stop duration. The final condition is either set to \( a_3 = 0 \) for constant velocity, \( t_5 = 0 \) for fusing the centering steps ↑↓↓↑ if \( t_4 = 0 \), or \( t_7 = 0 \) to reach time-optimality for the ↑↓↓↓ profile. Note that profiles might have multiple solutions. Most profiles are analytically solvable. For some profiles however, roots of up to sixth-order polynomials need to be found. Here, we make use of a safe Newton root-finding algorithm: Given an isolated root of a polynomial in an interval, a Newton method ensures quadratic convergence on average. Using a bisection method as a fallback strategy, an upper bound of the number of iterations for a given tolerance can be specified. This is required to ensure real-time capability.

The initial interval is found by solving the second derivative of the sixth-order polynomial analytically and check if a root exists between two extrema. This step is repeated for the first derivative, leading to intervals with isolated roots for the polynomial itself.

We calculate the numeric times \( t_k \) for all 24 possible profile types, as given in Table II per direction. However, not all solutions are physically reasonable or within the kinematic limits of the system. First, we check that every time step \( 1 \leq k \leq 7 \) is non-negative

\[
t_k \geq 0.
\]

Then, we integrate position, velocity and acceleration

\[
a_{k+1} = a_k + s_k j k t_k, \quad v_{k+1} = v_k + a_k t_k + \frac{s_k j k^2}{2}, \quad p_{k+1} = p_k + v_k t_k + \frac{a_k t_k^2}{2} + \frac{s_k j k^3 t_k^3}{6}
\]

for each time step. We check the acceleration limits via

\[
a_{min} < \{a_1, a_3, a_5\} < a_{max}
\]

and the velocity limits via

\[
v_{min} < v_k < v_{max} \quad \text{if} \quad a_k \cdot a_{k+1} \leq 0,
\]

\[
v_{min} < v_k < v_{max} \quad \text{if} \quad a_k = 0.
\]

If the profile passes all checks, it is added to the set of valid extremal profiles. The total duration is given by

\[
T = \sum_{k=1}^{7} t_k.
\]

We find the fastest profile and its duration \( T_{i,min} \) for each DoF \( i \) easily by comparing.

### C. Blocked Duration Intervals

Given the set of valid extremal profiles, we want to find all possible duration \( T_i > T_{i,min} \). In general, a number of blocked intervals \((T_{i,\text{start}}, T_{i,\text{end}})\) might exist, in which a DoF cannot reach the target with a duration within the interval. Fig. illustrates an example for a single blocked interval.

**Lemma IV.3.** For a third-order target state, up to two blocked intervals might exist.
Here, we refer to the work of Kröger et al. [3]. In particular, the maximal number of blocked intervals depend on the target velocity and acceleration being non-zero. In our case, we denote the two possible blocked intervals as $\alpha$ and $\beta$.

$$
\begin{array}{|c|c|c|}
\hline
\text{Velocity} & \text{Acceleration} & \text{Max. Number of Blocked Intervals} \\
\hline
v_f \neq 0 & a_f = 0 & 0 \\
v_f = 0 & a_f = 0 & 1 \\
- & a_f \neq 0 & 2 \\
\hline
\end{array}
$$

Furthermore, we want to clarify the relationship between blocked intervals and extremal profiles.

**Lemma IV.4.** A blocked interval is between two extremal profiles and each valid extremal profile corresponds to an interval boundary.

For all but extremal profiles, the duration can be adapted infinitesimally by changing the jerk $j_f$ or introducing a velocity or acceleration plateau. As the duration is constrained (to one side) for a boundary profile, it must be extremal and vice versa.

**Lemma IV.5.** A blocked interval can only exist between two neighboring profiles (regarding their duration).

Otherwise there would be a valid profile within a blocked interval.

Given up to two intervals, the set of valid extremal profiles must include exactly 1, 3, or 5 profiles. Given a sorted list of the profile duration, the profiles are mapped to blocked intervals as follows:

**One profile** results in no blocked intervals.

**Three profiles** lead to a single blocked interval $\alpha$. The interval is between the second and third profile.

**Five profiles** correspond to two blocked intervals $\alpha$ and $\beta$. The first interval is between the second and third profile, the second one between the fourth and fifth profile.

**Else** a failure of the algorithm would have occurred. In particular, the implementation has to deal with edge cases where different profile types merge.

**D. Minimum Duration**

Given the blocked duration intervals for each DoF, we want to find the minimum duration that is possible for all DoFs. Fig. 4 shows an exemplary illustration of this problem.

The minimum duration $T_f$ needs to be either time-optimal for a single DoF or correspond to the right boundary of a blocked interval:

$$
T_f \in \{ \forall i \in \{0, \ldots, N\} : \\
T_{ib} + T_{i,\alpha} \text{, if } \alpha \text{ exists} , \\
T_{ib} + T_{i,\beta} \text{, if } \beta \text{ exists} \}
$$

The duration of a possible braking pre-trajectory needs to be added. Then, up to $3N$ possible duration are sorted and evaluated in ascending order. The first duration that is not blocked in any DoF is the final trajectory duration $T_f$. The DoF $l$ that corresponds to the resulting duration is called the limiting DoF.

**E. Step 2: Time Synchronization**

Given the trajectory duration $T_f$, let $T_{ip} = T_f - T_{ib}$ be the duration of the profile without possible braking. Expect for the limiting DoF $l$ (where we can reuse the calculated profile from the prior step), we need to find trajectories of corresponding duration $T_{ip}$. Therefore, step 2 maps the duration $T_f$, the initial state $x_0$, the target state $x_f$ and the given limits

$$
S2 : (T_p, p_0, p_f, v_0, v_f, a_0, a_f, v_{max}, a_{max}, a_{min}, j_{max})
\rightarrow (t_1, t_2, t_3, t_4, t_5, t_6, t_7, j_f)
$$

to corresponding times $t_k$ and the final jerk constant $j_f$. In comparison the extremal profiles, we adapt the duration by changing the velocity plateau $\text{VEL}$ to $v_{plat}$ with $v_{min} < v_{plat} < v_{max}$ or by reducing the jerk $|j_f| < j_{max}$. With a velocity plateau below its limit, the ↑↓↓↑↑↑ profile gets possible for all profile types with $\text{VEL}$ limit, leading to the full 16 possible profiles for each direction (Table I). Here, we check all 32 profiles similarly to step 1, but return after the first valid profile is found.

---

Fig. 3. Example of a single blocked interval: Given $p_0 = -0.75$, $p_f = 0.75$, and $v_0 = v_f = 1.0$. No trajectory is physically possible with a duration below $t_{min} = 1.35$ and between $t \in (2.0, 2.6)$.

Fig. 4. Example of finding the minimum non-blocked duration of multiple DoFs. Shown are the blocked intervals (gray), possible minimum duration (red), possible braking pre-trajectories (blue), and the final minimum duration (black).
F. New State

So far, Ruckig has calculated the step duration $t_k$ and corresponding jerk signs $s_k$, the jerk value $j_f$, and a possible two-step brake pre-trajectory $t_{ib}$ and $j_{ib}$ for each DoF $i$. For each step $k \leq 7$, we integrate the acceleration $a_{k+1}$, velocity $v_{k+1}$ and position $p_{k+1}$ of the final profile according to (2), (3), and (4). Then, we can calculate the state at a given time $t$ by finding the last index $s$ that fulfills
\[
\sum_{i=1}^{s} t_i \leq t
\]
and integrating from index $s$ for time $t-t_s$ starting from the kinematic state $p_s$, $v_s$, and $a_s$. If a brake trajectory exists, we apply the same principle to this pre-trajectory. Usually, only this final integration of the new state will be repeated every control cycle of the robotic system. If and only if the input parameters change, the whole trajectory needs to be recalculated.

V. EXPERIMENTAL RESULTS

Ruckig is available under the permissive MIT license at https://github.com/pantor/ruckig. It is implemented as a C++17 library without further dependencies. Symbolic equations were solved ahead of time using Wolfram Mathematica; the corresponding notebooks are included in the repository. The generated equations were exported as C/C++. To keep the implementation simple, we preferred to export a polynomial which roots correspond to the profile solution. This form is then solved by our own C++ polynomial root solver. A Python wrapper using pybind11 for offline trajectory generation is available. Moreover, we’ve implemented a velocity-control interface which calculates time-optimal trajectories ignoring the current position, the target position and velocity limits. Due to its simplicity, we focus only on the complete position-control interface in this paper.

A. Robustness

To evaluate the robustness and numerical stability of the proposed algorithm, we generated a test suite of over 1 000 000 000 random trajectories with up to 7 DoFs each. The input parameters are drawn from
\[
\begin{align*}
p_0, p_f & \sim \mathcal{N}(\sigma = 4.0) \\
v_0, v_f, a_0, a_f & \sim \mathcal{N}(\sigma = 0.8) \\
v_{\text{max}}, a_{\text{max}}, j_{\text{max}} & \sim \mathcal{G}(2.0, 2.0) + 0.05
\end{align*}
\]
with the Normal distribution $\mathcal{N}$, the gamma distribution $\mathcal{G}$ and a minimum limit of $5 \times 10^{-2}$. We skip cases that violate the target acceleration requirement (1). Then, we define a successful calculation if the maximal deviations $\Delta$ between the result and the target state of
\[
\begin{align*}
|\Delta_p| & < 10^{-8}, \\
|\Delta_v| & < 10^{-8}, \\
|\Delta_a| & < 10^{-12}
\end{align*}
\]
are met. Here, we achieve a robustness of 100%. However, Ruckig is quite sensitive to long trajectory duration. During integration, the numerical error will propagate with $\Delta_p = T^2 \Delta_a$. With the precision of a double type for $a$ and the required position accuracy of $p$, this results in a maximal trajectory duration of
\[
T = \sqrt{\frac{\Delta_p}{\Delta_a}} \approx \sqrt{\frac{1 \times 10^{-8}}{2 \times 10^{-18}}} = 7.1 \times 10^3
\]
that fulfills the above numerical error. In SI-units, this corresponds to an upper limit of around 16 min. If cases above this maximum valid trajectory duration are ignored, Ruckig achieves a robustness of 100% even without any minimum limit value. Note that the input parameters are invariant to the unit of distance, so the input can be scaled without loss of generality.

We compare the duration of trajectories generated by Ruckig and Reflexxes Type IV for the above input distribution with $a_f = 0$. We find that both duration of every trajectory within our test suite are within a deviation of $|\Delta_t| < 10^{-3}$, supporting the claim of time-optimality of each other.

B. Calculation Duration

As an online trajectory generator, Ruckig is real-time critical and must output the next state within one control cycle of the robot. Typical control cycles range between 0.5 ms and 5 ms. Following measurements were done on an Intel i7-8700K CPU 3.70GHz 6-core CPU using a single thread with PREEMPT-Linux. The test suite for benchmarking reuses the above input distribution.

<table>
<thead>
<tr>
<th>Mean [µs]</th>
<th>Worst [µs]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ruckig (ours)</td>
<td>$a_f \neq 0$</td>
</tr>
<tr>
<td>Reflexxes Type IV</td>
<td>$a_f = 0$</td>
</tr>
<tr>
<td>opt_control</td>
<td>$a_f \neq 0$</td>
</tr>
</tbody>
</table>

Table III shows the mean and worst calculation performance for a robotic system with 7 DoFs. Fig. 5 shows the calculation duration depending on the number of DoFs.

As expected, we find a near linear relationship $O(N)$ between the average performance and the number of
DoFs. Furthermore, we find that Ruckig is well suited for control cycles as low as half a millisecond. In fact, $N = 82$ is the smallest number of DoFs that misses the control cycle of 1 ms in the worst case on our hardware.

C. Real-world Experiments

We have integrated Ruckig into our controller library franks for the Franka Emika Panda robot arm with 7 DoFs. Franks is available under the MIT license at [https://github.com/pantor/frankx](https://github.com/pantor/frankx) and allows for high-level motion generation. As the Franka robot checks for acceleration discontinuities in its real-time control, a constrained jerk is a hard requirement for franks. The robot has a control cycle time of 1 ms.

Ruckig is agnostic towards the used parametrization; commonly either the joint space or the Cartesian (task) space formulations are used. Furthermore, we use Cartesian space control with 3 translational, 3 rotational DoFs and a single elbow parameter. Furthermore, we highlight two possible applications.

1) Online Reaction to Sensor Input: A robot requires OTG to react to unforeseen sensor input. Fig. 6 shows a concrete example within the field of HRC. Here, the position of a human worker is monitored. After a detected safety violation, the velocity towards the human should be reduced to a minimum to weaken possible collisions. Moreover, the robot should move away from the human as fast as possible to avoid bruises or potential dangerous contact. Ruckig offers a high-level interface for this scenario: The robot moves as fast as possible to a safe target position with zero velocity, limited by a near-zero velocity $v_{\text{max}}$ towards the human, and a robot-limited velocity $v_{\text{min}}$ in the opposite direction.

2) Offline Trajectory Planning: Ruckig allows for trajectory planning by following a list of successive waypoints. In particular, waypoints with zero target velocity and acceleration correspond the a piecewise path without blending. Given following waypoints, we highlight the use of a non-zero acceleration target in a dynamic task: An object (without any jerk constraint) should be accelerated as fast as possible by a jerk-limited robot. Therefore, no impact between robot and object should occur, as this would lead to an acceleration violation. Then, the robot should only be in contact with the object at its maximum acceleration. After reaching the target velocity of the object, the robot smoothly decelerates and moves back to its initial position. Fig. 7 illustrates the resulting trajectory.
VI. DISCUSSION AND OUTLOOK

We presented Ruckig, an online trajectory generation (OTG) algorithm that is able to handle non-zero target accelerations. As Ruckig considers third-order constraints (for velocity, acceleration, and jerk), the complete kinematic state can be specified for waypoint-based trajectories. The proposed algorithm is real-time capable and of high performance regarding control cycles as low as 1 ms. Our implementation is available as an open-source C++ library.

In comparison to related work, Ruckig expands the capabilities of the proprietary Reflexxes Type IV library [7]. Reflexxes uses decision trees to find matching profiles as well as the blocked intervals, and calculates the numerical profile afterwards. With over 4760 unique nodes, these decision trees cause great complexity [8]. In contrast, Ruckig calculates all valid extremal profiles first and derives matching profiles and blocked intervals afterwards. The proposed algorithm is significantly simpler without decision trees: The relevant code-base of Ruckig has around 2800 lines of code in comparison to 23 000 lines for Reflexxes Type IV. To our own surprise, we find that the mean calculation performance of Ruckig is around twice as fast as Reflexxes. This is probably due to implementational details and better optimizations. In comparison to opt_control [2], Ruckig is able to handle blocked intervals for time synchronization and therefore guarantees a solution for arbitrary input states. Moreover, Ruckig is an order of magnitude faster and real-time capable. In contrast to other related work within the field of OTG, Ruckig supports a complete initial and target state [3, 5], time-optimality [1, 13], and multiple DoFs [11].

Furthermore, Kröger et al. showed that OTG can be applied to motion planning using intermediate waypoints [8]. We argue that Ruckig is much more suitable for this application: By defining waypoints with non-zero target accelerations, curves, circles, or splines can be better approximated, resulting in smoother motions. In contrast, waypoints with zero velocity result in plateaus of constant velocity. In the future, we want to further investigate this application to bridge the gap between OTG and path-following time-parametrization.

REFERENCES


